הטכניון
מכון טכנולוגי לישראל
TECHNION
Israel Institute of Technology

ספריות הטכניון
The Technion Libraries

בית הספר ללימודי מוסמכים ע"ש ארווין וג'ואן ג'ייקובס
Irwin and Joan Jacobs Graduate School

A Genetic Algorithm for Fully Automatic Non-Isometric Shape Matching

Michal Edelstein

A Genetic Algorithm for Fully Automatic Non-Isometric Shape Matching

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering

Michal Edelstein

Submitted to the Senate
of the Technion — Israel Institute of Technology
Elul 5779 Haifa September 2019

This research was carried out under the supervision of Prof. Mirela Ben-Chen, in the Faculty of Electrical Engineering.

The generous financial help of the Technion is gratefully acknowledged.

Contents

List of Figures

Abstract

Shape correspondence is a challenging task in geometry processing and computer graphics with various applications including texture transfer, shape interpolation and statistical modeling. The goal is to find a meaningful relation between two or more shapes. The correspondence can be either sparse, where a small number of meaningful landmarks are matched, or dense, where a full map between the shapes is retrieved. Finding such a map between non-isometric shapes is difficult since the shapes can differ greatly from one another and the matching needs to be semantic.

We propose a fully automatic method for shape correspondence that is widely applicable, and especially effective for non-isometric shapes and shapes of different topology. We observe that fully automatic shape correspondence can be decomposed as a hybrid discrete-continuous optimization problem, and we find the best sparse landmark correspondence, whose sparse-to-dense extension minimizes a local metric distortion. To tackle the combinatorial task of landmark correspondence we use an evolutionary genetic algorithm, where the local distortion of the sparse-to-dense extension is used as the objective function.

We design novel geometrically guided genetic operators, which, when combined with our objective, are highly effective for non-isometric shape matching. Our method outperforms state of the art methods for automatic shape correspondence both quantitatively and qualitatively on challenging datasets.

Chapter 1

Introduction

Shape correspondence is a fundamental task in shape analysis: given two shapes, the goal is to compute a semantic correspondence between points on them. Shape correspondence is required when two shapes are analyzed jointly, which is common in many applications such as texture and deformation transfer [SP04], statistical shape analysis [MDW08] and shape classification [ESKBC17], to mention just a few examples.

The difficulty of the shape matching problem depends on the class of *deformations* that can be applied to one shape to align it with the second. For example, if only *rigid* transformations are allowed it is easier to find a correspondence than if non-rigid deformations are also possible, since the number of degrees of freedom is small and the space of allowed transformations is easy to parameterize. Similarly, if only *isometric* deformations are allowed, the matching is easier than if non-isometry is possible, since then there is a clear criteria of the quality of the map, namely the preservation of geodesic distances. The hardest case is when the two shapes belong to the same semantic class, but are not necessarily isometric. In this case, the correspondence algorithm should achieve two goals: (1) put in correspondence semantically meaningful points on both shapes, and (2) reduce the *local* metric distortion.

Hence, the non-isometric shape correspondence problem is often considered as a *two step* process. First, the *global semantics* of the matching is given by a sparse set of *corresponding landmarks* of salient points on both shapes. If this set is informative enough, then the full shapes can be matched by extending the landmark correspondence to a full map from the source to the target in a *consistent* and *smooth* way. The first problem is *combinatorial*, requiring the computation of a permutation of a subset of the landmarks, whereas the second problem is *continuous*, requiring the definition and computation of local differential properties of the map. Whereas the second problem has been tackled by multiple methods [AL16, MCSK$^+$17, ESBC19, EHA$^+$19] which yield excellent results for non-isometric shapes, methods that address the sparse landmark correspondence problem [KKBL15, MDK$^+$16, DML17, Sah18] have so far been limited either to the nearly isometric case, or to a very small set of landmarks.

We propose to leverage the efficient algorithms for solving the second problem to generate a framework for solving the first. Specifically, we suggest a combinatorial optimization for matching a sparse set of landmarks, such that the best obtainable local distortion of the corresponding

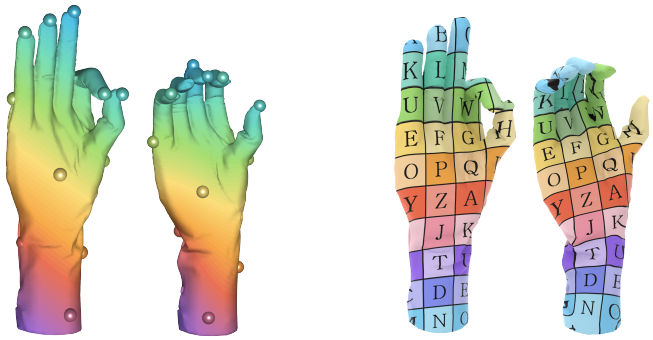


Figure 1.1: A map between shapes of different genus obtained by our approach. (left) Output landmark correspondence and functional map, visualized using color transfer. (right) Final pointwise map visualized using texture transfer.

sparse-to-dense extension is minimized. As the optimization tool, we propose to use a *genetic algorithm*, as these have been used for combinatorial optimization for a few decades [Hol92], and are quite general in the type of objectives they can optimize. Despite their success in other fields, though, to the best of our knowledge, their use in shape analysis has been limited so far to isometric matching [Sah18].

Using a genetic algorithm allows us to optimize a challenging objective function, that is given both in terms of the landmark permutation and the differential properties of the extended map computed from these landmarks. We use a non-linear non-convex objective, given by the *elastic energy* of the deformation of the shapes, an approach that has been recently used successfully for non-isometric matching [EHA+19] when the landmarks are known. Furthermore, we apply the *functional map* framework [OBCS+12] to allow for an efficient computation of the elastic energy. Finally, paramount to the success of a genetic algorithm are the the genetic operators, that *combine* two sparse correspondences to generate a new one, and *mutate* an existing correspondence. We design novel *geometric* genetic operators that are guaranteed to yield new valid correspondences. We show that our algorithm yields a landmark correspondence that, when extended to a functional map and a full vertex-to-point map, outperforms existing state-of-the-art techniques for automatic shape correspondence, both quantitatively and qualitatively.

1.1 Related Work

As the literature on shape correspondence is vast, we discuss here only methods which are directly relevant to our approach. For a more detailed survey on shape correspondence we refer the reader to the recent excellent reviews [VKZHCO11, TCL+13].

Fully automatic shape correspondence. Many fully automatic methods, like ours, compute a *sparse* correspondence between the shapes, to decrease the number of degrees of freedom and possible solutions. A sparse-to-dense method can be then used in a post processing step to obtain a dense map. For example, Kezurer et al. [KKBL15] formulated the shape correspondence

problem as a Quadratic Assignment Matching problem, and suggested a convex semi-definite programming (SDP) relaxation to solve it efficiently. While the convex relaxation was essential, their method is only suitable for small sets (of the same size) of corresponding landmarks. Dym et al. [DML17] suggested to combine doubly stochastic and spectral relaxations to optimize the Quadratic Assignment Matching problem, which is not as tight as the SDP relaxation, but much more efficient. Maron et al. [MDK$^+$16] suggested a convex relaxation to optimize a term that relates pointwise and functional maps, which promotes isometries by constraining the functional map to be orthogonal. Zhang et al. [ZSCO$^+$08] used a search tree traversal technique to optimize a deformation energy of sparse landmark correspondence.

Other methods for the automatic computation of a dense map include Blended Intrinsic Maps (BIM) by Kim et al. [KLF11], who used the most isometric combination of conformal maps. Their method works well for relatively similar shapes and generates locally smooth results, yet is restricted to mapping between genus-0 surfaces. Vestner et al. [VLR$^+$17] suggested a multi scale approach that is not restricted to isometries, but requires shapes with the same number of vertices and generates bijective vertex to vertex correspondence.

A different approach to tackle the correspondence problem is to compute a *fuzzy* map [OBCS$^+$12, SPKS16]. The first approach puts in correspondence functions instead of points, whereas the second is applied to probability functions. These generalizations allow much more general types of correspondences, e.g. between shapes of different genus, however, they also require an additional pointwise map extraction step.

The functional map approach was used and extended by many following methods, for example Nogneng et al. [NO17] introduced a pointwise multiplication preservation term, Huang et al. [HO17] used the adjoint operators of functional maps, and Ren et al. [RPWO18] recently suggested to incorporate an orientation preserving term and a pointwise extraction method that promotes bijectivity and continuity, that can be used for non isometric matching as well.

Our method differs from most existing methods by the quality measure that we optimize. Specifically, we optimize for the landmark correspondence by measuring the elastic energy of the *dense* correspondence implied by these landmarks. As we show in the results chapter, our approach outperforms existing automatic state-of-the-art techniques on challenging datasets.

Semi-automatic shape correspondence. Many shape correspondence methods use a non-trivial initialization, e.g. a sparse landmark correspondence, to warm start the optimization of a dense correspondence.

Panozzo et al. [PBDSH13] extended the landmark correspondence by computing the surface barycentric coordinates of each point w.r.t. the landmarks on the source shape, and matching it to the point on the target shape with the same barycentric coordinates w.r.t. the target landmarks. The landmarks could be chosen interactively, which allowed for intuitive user control. More recently, Gehre et al. [GBKS18], used curve constraints and functional maps, for computing correspondences in an interactive setting. Given landmark correspondences or an extrinsic alignment, Mandad et al. [MCSK$^+$17] used a soft correspondence, where each source point is matched to a point on the target shape with a certain probability, and minimized the variance

of this distribution. Parameterization based methods map the two shapes to a single, simple domain, such that the input landmark correspondence is preserved, and the composition of the maps from each shape to the common domain gives the final result [APL15, AL15, AL16]. Since minimizing the distortion of the map to the common domain does not guarantee minimal distortion of the final map, recently Ezuz et al. [ESBC19] directly optimized the Dirichlet energy and the reversibility of the forward and backward maps, which led to results with low conformal distortion. Finally, many automatic methods, for example all the functional map based approaches, can use landmarks as an auxiliary input to improve results in highly non isometric settings.

The output of our method is a sparse correspondence and a functional map, that can be used as input to semi-automatic methods to generate a dense map, thus our approach is complementary to semi-automatic methods. Specifically, we use [ESBC19] to extract a dense vertex-to-point map from the landmarks and functional map we compute with the genetic algorithm.

Genetic algorithms. Genetic algorithms were initially inspired by the process of evolution and natural selection [Hol92]. In the last few decades they were used in many areas, such as protein folding simulations [UM93], clustering [MB00] and image segmentation [BLM95]. In the context of graph and shape matching, genetic algorithms were used for registration of depth images [CTL04, SBB05], 2D shape recognition in images [OM97], rigid registration of 3-D curves and surfaces [YAF99], and inexact graph matching [CWH97, Auw07].

More recently, Sahillioğlu [Sah18] suggested a genetic algorithm for isometric shape matching. However, their approach is drastically different from ours. Their objective was the preservation of pairwise distances between the *sparse* landmarks. We, on the the other hand, use as our objective the energy of a *dense* correspondence, which is the output of a sparse-to-dense algorithm. The different focus of our approach is also evident in the other genetic operators that we define, which allow us to compute a highly non isometric matching.

1.2 Overview and Contribution

Our general pipeline is described in Chapter 3, and the following sections provide more details. We start by selecting geometrically meaningful landmarks on each shape (Chapter 4). Since we do not expect the landmark sets on the two shapes to match exactly, in the next step we compute a *partial landmark correspondence*, where each source landmark can be either matched to a *unique* target landmark or remained unmatched. We compute the landmark correspondence, as well as a functional map, using a genetic algorithm, as described in Chapter 5. Finally, we use a post processing step that extends the solution to a dense smooth correspondence. In Chapter 6 we demonstrate our results. We compare with other state of the art methods for automatic sparse correspondence [KKBL15, DML17, Sah18], and with the recent functional map based method [RPWO18], that automatically computes dense maps and does not have topological restrictions. We apply the same post-processing for computing a sparse-to-dense map to all

methods, and show that we outperform previous methods, as demonstrated by both quantitative and qualitative evaluation.

Chapter 2

Background

2.1 Notation

We use the following notation throughout the paper. We compute a correspondence between two manifold triangle meshes, denoted by M_1 and M_2, with n_1, n_2 vertices respectively. The vertex, edge and face sets are denoted by $\mathcal{V}_i,\ \mathcal{E}_i,\ \mathcal{F}_i$ respectively, where the subscript $i \in \{1,2\}$ indicates the corresponding mesh. The embedding in $\mathbb{R}^3$ of the mesh M_i is given by $X_i \in \mathbb{R}^{n_i \times 3}$. The area of a face f is denoted by a_f, and similarly a_v is the area of a vertex v (defined to be a third of the area of its adjacent faces).

A map that assigns a point on M_2 to each *vertex* of M_1 is denoted by $T_{12} : \mathcal{V}_1 \to M_2$, the corresponding matrix is $P_{12} \in \mathbb{R}^{n_1 \times n_2}$ (see section 2.2). Similarly, maps in the opposite direction are with opposite subscript, e.g. T_{21} is a pointwise map from M_2 to M_1.

The eigenfunctions of the Laplace-Beltrami operator that correspond to the smallest $k_1,\ k_2$ eigenvalues are used as reduced bases for scalar functions, and they are stacked as the columns of the matrices $\Psi_1 \in \mathbb{R}^{n_1 \times k_1},\ \Psi_2 \in \mathbb{R}^{n_2 \times k_2}$. For a vertex $i \in \mathcal{V}_1$, its corresponding row in the basis matrix is denoted by $\Psi_1\,[row\ i] \in \mathbb{R}^{1 \times k_1}$. The functional map in this reduced basis is denoted by $C_{12} \in \mathbb{R}^{k_1 \times k_2}$ (see equation (2.1)).

2.2 Functional Maps

Formulation. The term *functional maps*, as first introduced by Ovsjanikov et al. [OBCS$^+$12, OCB$^+$16], refers to a linear operator that maps scalar functions on one shape to the other. A pointwise map T_{12} from M_1 to M_2 can be fully represented by a matrix $P_{12} \in \mathbb{R}^{n_1,n_2}$, where at each row i there are at most 3 non zero entries, at entries that correspond to vertices of the face on M_2 that i is mapped to, and the values are the barycentric coordinates. A piecewise linear function on M_2, represented by a vector of vertex values $f_2 \in \mathbb{R}^{n_2}$, can be transported to M_1 by $P_{12} f_2 \in \mathbb{R}^{n_1}$. It assigns to each vertex v of M_1 the value of f_2 at the point it is mapped to, $T_{12}(v)$. By projecting P_{12} on reduced bases for scalar functions, Ψ_1 of size k_1, Ψ_2 of size k_2

respectively, we get a compact *functional map*:

$$C(P_{12}) = C_{12} = \Psi_1^\dagger P_{12} \Psi_2 \in \mathbb{R}^{k_1 \times k_2}. \tag{2.1}$$

Similarly, given a functional map C_{12}, one option to convert it to a pointwise map is to use:

$$P(C_{12}) = P_{12} = \Psi_1 C_{12} \Psi_2^\dagger \in \mathbb{R}^{n_1 \times n_2}. \tag{2.2}$$

Note, that there are multiple ways to extract a pointwise map from a functional map, see e.g. the discussion in [EBC17].

Basis. A common choice of basis for these subspaces is given by the first k_1, k_2 eigenfunctions of the Laplace-Beltrami (LB) operator of each shape, such that smooth functions can be well approximated using a small number of coefficients and C_{12} is compact.

It is often valuable to use a larger basis size for the *target functions*, so that mapped functions are well represented. Hence, since C_{ij} maps functions on M_j to functions on M_i, Ψ_i should contain more basis functions than Ψ_j. Thus, we denote the number of source eigenfunctions by k_s, the number of target eigenfunctions by k_t, and the functional maps in both directions $C_{12},\ C_{21}$ are of size $k_t \times k_s$. Slightly abusing notations, and to avoid clutter, we use Ψ_i to denote the eigenfunctions corresponding to M_i, in both directions, namely both when $k_i = k_s$ and $k_i = k_t$, as the meaning is often clear from the context. Where required, we will explicitly denote by $\Psi_{i,s}, \Psi_{i,t}$ the eigenfunctions with dimensions k_s, k_t, respectively.

Objectives. Many cost functions have been suggested for functional map computation, e.g., [OBCS$^+$12, NO17, CMS18, RPWO18], among others. In our approach we use the following terms.

Landmark correspondence. Given a set Π of pairs of corresponding landmarks, $\Pi = \{(i,j) \,|\, i \in \mathcal{V}_1, j \in \mathcal{V}_2\}$, we use the term:

$$\mathcal{E}^m\left(C_{12}, \Pi\right) = \sum_{(i,j)\in\Pi} \|\Psi_1\left[row\ i\right] C_{12} - \Psi_2\left[row\ j\right]\|^2. \tag{2.3}$$

While some methods use landmark-based descriptors, we prefer to avoid it due to possible bias towards isometry that might be inherent in the descriptors. The formulation in Equation (2.3) has been used successfully by Gehre et al. [GBKS18] for functional map computation between highly non isometric shapes, as well as in the context of pointwise map recovery [EBC17].

Commutativity with Laplace-Beltrami. We use:

$$\mathcal{E}^\Delta\left(C_{12}\right) = \|\Delta_1 C_{12} - C_{12}\Delta_2\|_F^2, \tag{2.4}$$

where Δ_i is a diagonal matrix with the first k_i eigenvalues of the Laplace-Beltrami operator of M_i on the diagonal. While initially this term was derived to promote isometries, in practice it

has proven to be useful for highly non isometric shape matching as well [GBKS18].

2.3 Elastic Energy

Elastic energies are commonly used for shape deformation [BPGK06, SA07, HRWW12, HRS+14], but were used for shape matching as well [LDRS05, WSSC11, dBDFN16, IRS18, EHA+19]. In this paper we use a recent formulation that achieved state of the art results for non isometric shape matching [EHA+19]. The full details are described there, and we mention here only the main equations for completeness.

The elastic energy is defined for a source, or undeformed mesh M, and a deformed mesh with the same triangulation but different geometry $\tilde{M}$. It consists of two terms, a *membrane* term and a *bending* term. The membrane energy penalizes area distortion:

$$\mathcal{E}^{\text{mem}}\left(M,\tilde{M}\right)=\sum_{t\in\mathcal{F}}a_t\left(\frac{1}{2}\text{tr}\mathcal{G}_t+\frac{1}{4}\det\mathcal{G}_t-\frac{3}{4}\log\det\mathcal{G}_t-\frac{5}{4}\right), \tag{2.5}$$

where a_t denotes the area of face t, and $\mathcal{G}_t$ denotes the geometric distortion tensor of the face t. In addition, in order to have a finite expression in case some triangles are deformed into zero area triangles, the negative $\log$ function is linearly extended below a small threshold.

The bending energy, on the other hand, penalizes misalignment of curvature features:

$$\mathcal{E}^{\text{bnd}}\left(M,\tilde{M}\right)=\sum_{e\in\mathcal{E}}\frac{(\tilde{\theta}_e-\theta_e)^2}{\tilde{d}_e}\left(\tilde{l}_e\right)^2, \tag{2.6}$$

where $\theta_e,\tilde{\theta}_e$ denote the dihedral angle at edge e in the undeformed and deformed surfaces respectively; if t,t' are the adjacent triangles to e then $\tilde{d}_e=\frac{1}{3}(\tilde{a}_t+\tilde{a}_{t'})$, and $\tilde{l}_e$ is the length of e in the deformed surface. Again, to handle degeneracies, we sum only over edges where both adjacent triangles are not degenerate.

The total elastic energy is:

$$\mathcal{E}^{\text{elstc}}\left(M,\tilde{M}\right)=\mu\,\mathcal{E}^{\text{mem}}\left(M,\tilde{M}\right)+\eta\,\mathcal{E}^{\text{bnd}}\left(M,\tilde{M}\right), \tag{2.7}$$

where we always use $\mu=1,\eta=10^{-3}$.

In the context of shape matching evaluation, the undeformed mesh is M_1, and the geometry of the deformed mesh is given by the embedding of the points on M_2 that correspond to the vertices of M_1. Specifically, these are given by $T_{12}(\mathcal{V}_1)$, or equivalently, $P_{12}X_2$, where X_2 is the embedding of the vertices of M_2.

2.4 Energies of Functional Maps

Elastic energy. The elastic energy can also be used to evaluate functional maps directly, by setting the geometry of the deformed mesh to $P(C_{12})X_2=\Psi_1C_{12}\Psi_2^{\dagger}X_2$. For brevity, we

denote this energy by $\mathcal{E}^{\text{elstc}}(C_{12})$.

Reversibility energy. In [EHA+19] the elastic energy was symmetrized and combined with a *reversibility* term that evaluates bijectivity. The reversibility term requires computing both C_{12} and C_{21}. Here we define the reversibility energy for functional maps, similarly to [ERGB16]:

$$\mathcal{E}^{\text{rev}}(C_{12}, C_{21}) = \|C_{12}\Psi_2^\dagger P(C_{21})X_1 - \Psi_1^\dagger X_1\|_F^2 + \|C_{21}\Psi_1^\dagger P(C_{12})X_2 - \Psi_2^\dagger X_2\|_F^2 \,. \quad (2.8)$$

The reversibility energy measures the distance between vertex coordinates (projected on the reduced basis), and their mapping to the other shape and back. The smaller this distance is, the more bijectivity is promoted [ESBC19].

Chapter 3

Method

Our goal is to automatically compute a semantic correspondence between two shapes, denoted by M_1 and M_2. The shapes are the only input to our method. We do not assume that the input shapes are isometric, but we do assume that both shapes belong to the same semantic class, so that a semantic correspondence exists.

Our pipeline consists of three main steps, see Figure 3.1:

(a) Compute two sets of geometrically meaningful landmarks on M_i, denoted by $\mathcal{S}_i$ (Chapter 4).

(b) Compute a sparse correspondence, i.e. a permutation, between $\mathcal{S}_i^+ \subseteq \mathcal{S}_i$, as well as corresponding functional maps C_{ij}, using a genetic algorithm (Chapter 5).

(c) Generate a dense pointwise map using an existing semi-automatic correspondence method [ESBC19].

We use standard techniques for the first and last steps, and thus our main technical contribution lies in the design of the genetic algorithm.

The most challenging component of our method is determining the objective function. Unlike the isometric correspondence case, where it is known that pairwise distances between landmarks should be preserved, in the general (not necessarily isometric) case there is no known criterion that the landmarks should satisfy. However, there exist well studied *differential* quality measures that proved to be useful in practice for *dense* non isometric correspondence. Therefore

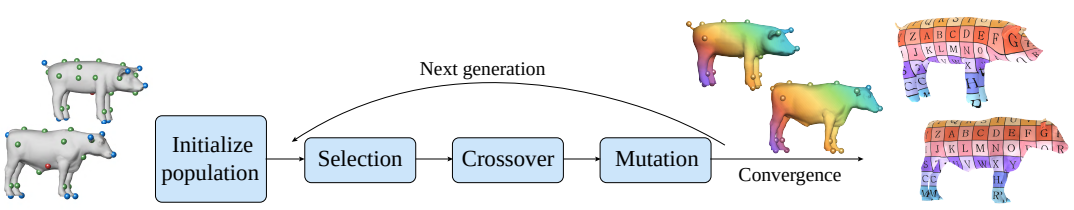


Figure 3.1: Our pipeline consists of three stages: (a) landmark computation (Chapter 4), (b) genetic algorithm (Chapter 5), (c) sparse to dense post processing using [ESBC19].

we use an existing method to extend the landmark correspondence to a dense correspondence, and define our objective function $\mathcal{E}^{\text{fit}}$ by the quality of the computed dense map.

To extend the landmark correspondence to a full dense correspondence, we chose a functional map approach, as it performs most computations in a reduced basis and is thus significantly faster than pointwise approaches. In our setting speed is of highest importance, as we compute a dense map for many different landmark correspondences, and indeed the sparse to dense approach we chose only takes 0.016 seconds on average.

The general formulation of the optimization problem we address is therefore:

$$\begin{aligned} &\underset{\Pi}{\text{minimize}} \quad \mathcal{E}^{\text{fit}}(C_{\text{opt}}(\Pi)) \\ &\text{subject to} \quad C_{\text{opt}}(\Pi) = \underset{C}{\text{argmin}}\ \mathcal{E}^{\text{map}}(C, \Pi), \end{aligned} \tag{3.1}$$

where Π is a permutation, which maps a subset of the landmarks $\mathcal{S}_1^+ \subseteq \mathcal{S}_1$, to a subset of the landmarks $\mathcal{S}_2^+ \subseteq \mathcal{S}_2$, and C is a functional map. Note that the objective that we *measure* for a given permutation Π, i.e. $\mathcal{E}^{\text{fit}}$, is different than the objective that we *optimize* to extend Π to a full map, i.e., $\mathcal{E}^{\text{map}}$. Due to the genetic algorithm, which can be applied to very general objective functions, the first can be considerably more complicated, i.e. non-linear and non-convex, than the second.

In the following we discuss the details of the algorithm, first addressing the landmark computation, and then the design of the genetic algorithm that we use to optimize Equation (3.1). Our algorithm contains a few parameters that are fixed in all our experiments (parameter tuning is not required). The parameters are listed and described in Appendix A.

Chapter 4

Automatic Landmark Computation

This chapter describes the first step of our pipeline, automatic computation of geometrically meaningful landmarks on each shape. As a pre-processing step, we normalize both meshes to have area 1. We classify the landmarks into three different categories, based on their computation method: *maxima*, *minima*, and *centers*.

Maxima and Minima. The first two categories are the local maxima and minima of the Average Geodesic Distance (AGD), that is frequently used in the context of landmark computation [KLF11, KKBL15]. The AGD of a vertex $v \in \mathcal{V}$ is defined as: $AGD(v) = \sum_{u \in \mathcal{V}} a_u d(v,u)$ where a_u is the vertex area and $d(v,u)$ is the geodesic distance between $v,\ u$. To efficiently compute approximate geodesic distances, we compute a high dimensional embedding as suggested by Panozzo et al. [PBDSH13], and use the Euclidean distances in the embedding space that approximate geodesic distances. The maxima of AGD are typically located at tips of sharp features, and the minima at centers of smooth areas, thus the maxima of the AGD provide the most salient features.

Centers. As the maxima and minima of the AGD are very sparse, we add additional landmarks using the local minima of the function:

$$f_N(v) = \sum_{k \leq N} \frac{1}{\sqrt{\lambda_k}} \frac{|\Psi_k(v)|}{\|\Psi_k\|_{L^\infty}} \qquad v \in \mathcal{V}\,, \tag{4.1}$$

defined by Cheng et al. [CMS18], who referred to these minima as *centers*. Here, Ψ_k and λ_k are the k^{th} eigenfunction and eigenvalue of the Laplace-Beltrami operator, and N is the number of eigenfunctions. We use the minima of Equation (4.1) rather than, e.g., farthest point sampling [KKBL15], as the centers tend to be more consistent between non isometric shapes.

Filtering. Landmarks that are too close provide no additional information, and add unnecessary degrees of freedom. Hence, we filter the computed landmarks so that the minimal distance between the remaining ones is above a small threshold d_ϵ. When filtering, we prioritize the landmarks according their salience, namely we prefer maxima of the AGD, then minima and

then centers. Finally, if the set of landmarks is larger than a maximal size of $m_{\max}$, we increase d_ϵ automatically to yield less landmarks.

Adjacent landmarks. We would like to define the genetic operators such that the geometry of the shapes is taken into consideration. Hence, we define two landmarks $l_i, r_i \in \mathcal{S}_i$ to be *adjacent* if their geodesic distance $d_i(l_i, r_i) < d_{\text{adj}}$ or if they are neighbors in the geodesic Voronoi diagram of M_i. The set of adjacent landmarks to l_i is denoted by $\mathcal{A}(l_i)$.

Landmark origins. The landmark categories are saved and used in the next step (the genetic algorithm) as well. Given a landmark $l_1 \in \mathcal{S}_1$, all the landmarks in $\mathcal{S}_2$ which are from the same category are denoted by $\mathcal{O}(l_1)$.

The maxima, minima and centers of each shape that remain after filtering form the landmark sets $\mathcal{S}_1, \mathcal{S}_2$. The number of landmarks is denoted by m_1, m_2, and is not necessarily the same for M_1, M_2.

Figure 3.1(a) shows example landmark sets, where the color of a landmark indicates its type: blue for maxima, red for minima and green for centers. As expected, the landmarks do not entirely match, however there exists a substantial *subset* of landmarks that do match. At this point the correspondence between the landmarks is not known, and it is automatically computed in the next step, using the genetic algorithm.

Chapter 5

Genetic Non Isometric Maps

Genetic algorithms are known to be effective for solving challenging combinatorial optimization problems with many local minima.

In a genetic algorithm [Hol92] solutions are denoted by *chromosomes*, which are composed of *genes*. In the initialization step, a collection of chromosomes, known as the *initial population* is created. The algorithm modifies, or *evolves*, this population by *selecting* a random subset for modification, and then combining two chromosomes to generate a new one (*crossover*), and modifying (*mutating*) existing chromosomes. The most important part of a genetic algorithm is the objective that is optimized, or the *fitness function*. The ultimate goal of the genetic algorithm is to find a chromosome, i.e. a solution, with the best fitness value. The general genetic algorithm is described in Algorithm 5.1.

Genetic algorithms are quite general, as they allow the fitness function to be any type of function of the input chromosome. We leverage this generality to define a fitness function that is itself the result of an optimization problem. We further define the genes, chromosomes, crossover and mutation operators and initialization and selection strategies, in a geometric manner.

Algorithm 5.1 Genetic Algorithm.

input : $M_1, M_2, \mathcal{S}_1, \mathcal{S}_2$
output : Π, C_{opt}
Initialize population ; `// Section 5.2`
Evaluate fitness ; `// Section 5.3`
while *population did not converge* **do**
    Select individuals for breeding ; `// Section 5.4`
    Perform crossover ; `// Section 5.5`
    Perform mutation ; `// Section 5.6`
    Evaluate offspring fitness and add to population
end
Compute output from fittest chromosome

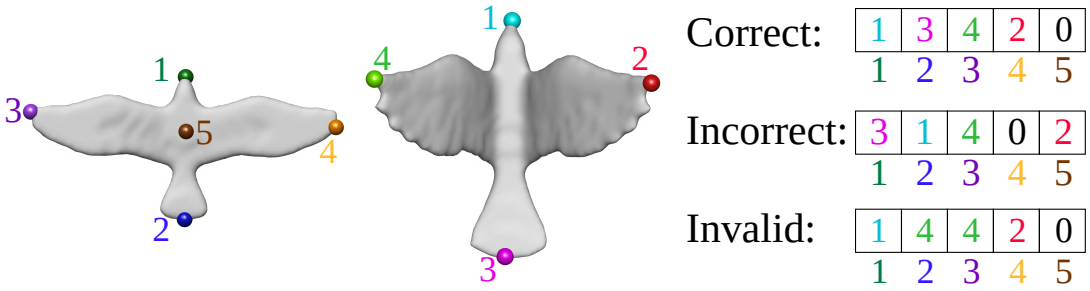


Figure 5.1: Illustration of chromosomes, which represent a sparse correspondence. The index of an array entry corresponds a landmark of M_1 (left), and the value corresponds to a landmark on M_2 (right shape). The first chromosome is the desired semantic correspondence, the second is valid but semantically incorrect, and the third is invalid because it is not injective (landmarks 2,3 on M_1 correspond to the same landmark with index 4 on M_2).

5.1 Genes and Chromosomes

Genes. A *gene* is given by a pair (l_1, l_2) such that $l_1 \in \mathcal{S}_1$ and $l_2 \in \mathcal{S}_2 \cup 0$, and encodes a single landmark correspondence. If $l_2 = 0$ we denote it as an *empty* gene, otherwise it is a *non-empty* gene.

Adjacency Preserving Genes. Two non-empty genes (l_1, l_2) and (r_1, r_2) are defined to be *adjacency preserving (AP) genes*, if l_i, r_i are adjacent landmarks on M_i, for $i \in \{1, 2\}$.

Chromosome. A *chromosome* is a collection of exactly m_1 genes, that includes a single gene for every landmark in $\mathcal{S}_1$. A chromosome is *valid* if it is injective, namely each landmark on $\mathcal{S}_2$ is assigned to at most a single landmark in $\mathcal{S}_1$. We represent a chromosome using an integer array of size m_1, and denote it by c, thus the gene (l_1, l_2) is encoded by $c[l_1] = l_2$ (see Figure 5.1).

Match. A *match* defined by a chromosome c is denoted by $\Pi(c)$ and includes all the non-empty genes in c. The sets $\mathcal{S}_i^+(\Pi) \subseteq \mathcal{S}_i$ are the landmarks that participate in the genes of Π, i.e., all the landmarks that have been assigned.

5.2 Initial Population

There are various methods for initialization of genetic algorithms, Paul et al. [PRB+13] discusses and compares different initialization methods of genetic algorithms for the Traveling Salesman Problem, where the chromosome definition is similar to ours. Based on their comparison and the properties of our problem, we chose to use a *gene bank* [WHG07], i.e. for each source landmark we compute a subset of target landmarks that are a potential match.

To compare between landmarks on the two shapes we use a descriptor based distance, where we chose the Wave Kernel Signature (WKS) [ASC11]. While this choice can induce some

isometric bias, as we generate *multiple* matches for each source landmark, this bias does not affect our results. Let $\mathcal{W}(l_1, l_2)$ denote the normalized WKS distance between two landmarks $l_1 \in \mathcal{S}_1, l_2 \in \mathcal{S}_2$, such that the distance range is $[0, 1]$, and is normalized separately for each landmark.

Gene bank. The *gene bank* of a landmark $l_1 \in \mathcal{S}_1$, denoted by $\mathcal{G}(l_1)$ is the set of genes that match l_1 to a landmark $l_2 \in \mathcal{S}_2$ which is close to it in WKS distance, and is of the same origin. Specifically:

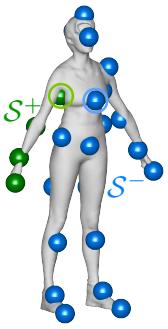
$$\mathcal{G}(l_1) = \{(l_1, l_2) \mid l_2 \in \mathcal{O}(l_1) \text{ and } \quad \mathcal{W}(l_1, l_2) < \epsilon_{\text{wks}} \text{ and } \mathcal{W}(l_2, l_1) < \epsilon_{\text{wks}}\}, \tag{5.1}$$

The gene bank defines an initial set of possibly similar landmarks. Since it is used for the initialization, the set does not need to be accurate and the WKS distance provides a reasonable initialization even for non-isometric shapes.

Prominent landmark. A landmark $l_1 \in \mathcal{S}_1$ is denoted as a *prominent landmark* if its gene bank $\mathcal{G}(l_1)$ is not empty, and has at most 4 genes. This indicates that our confidence in this landmark is relatively high, and we will prefer to start the chromosome building process from such landmarks.

Finally, to add more genes to an existing chromosome, we will need the following definition.

Closest matched/unmatched pair. Given two disjoint subsets $\mathcal{S}_1^+, \mathcal{S}_1^- \subseteq \mathcal{S}_1$, we define the *closest matched/unmatched pair* as the closest adjacent pair, where each landmark belongs to a different subset. In the inset figure, $\mathcal{S}_1^+$ is displayed in green and $\mathcal{S}_1^-$ is displayed in blue. The *closest matched/unmatched pair* $[l_1^+, l_1^-]$ is circled, l_1^+ in green and l_1^- in blue. Explicitly:



$$[l_1^+, l_1^-] = \underset{l_1 \in \mathcal{S}_1^+, r_1 \in \mathcal{S}_1^-}{\operatorname{argmin}} \; d_1(l_1, r_1) \quad \text{s.t.} \quad r_1 \in \mathcal{A}(l_1). \tag{5.2}$$

5.2.1 Chromosome construction

Using these definitions, we can now address the construction of a new chromosome c as seen in Algorithm 5.2 and demonstrated in Figure 5.2.

First, we randomly select a *prominent landmark* l_1, and add a random gene from its gene bank $\mathcal{G}(l_1)$ to c. We maintain two subsets $\mathcal{S}_1^+, \mathcal{S}_1^- \subseteq \mathcal{S}_1$, that denote the landmarks that have non-empty genes in c, and the unprocessed landmarks, respectively. Hence, initially, $\mathcal{S}_1^+ = \{l_1\}$, and $\mathcal{S}_1^- = \mathcal{S}_1 \setminus \{l_1\}$.

Then, we repeatedly find a *closest matching pair* $[l_1^+, l_1^-]$, and try to add an *adjacency preserving gene* for l_1^- which keeps the chromosome *valid*. First, we look for an AP gene in the gene bank $\mathcal{G}(l_1^-)$. If none is found, we try to construct an AP gene (l_1^-, l_2), where $l_2 \in \mathcal{S}_2$, and is of the same *type* (i.e., maxima, minima or center) as l_1^-.

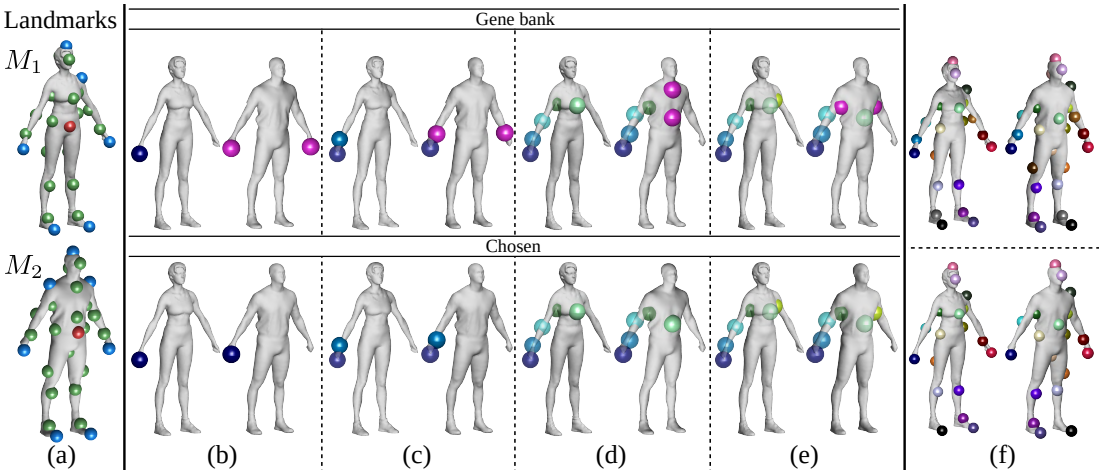


Figure 5.2: Chromosome Construction demonstration.

If no AP gene can be constructed which maintains the validity of c, an empty gene for l_1^- is added to c, and we look for the next closest matching pair. If $\mathcal{A}(\mathcal{S}_1^+) \cap \mathcal{S}_1^-$ is empty, namely, no more adjacent landmarks remain unmatched, empty genes are added to c for all the remaining landmarks in $\mathcal{S}_1^-$.

Figure 5.2 demonstrates the chromosome construction. Both meshes and their computed landmarks (color indicates landmark type) are displayed in (a). In the first stage, a random *prominent landmark* is selected, its gene bank (GB) is displayed in magenta on M_2 (b - top). Then, one random feature is selected from the GB (b - bottom). In the next stage, the *closest unmatched pair* is selected, and its matching options from the GB are displayed in magenta (c - top). Since only the option from the first chromosome is *adjacency preserving (AP)*, this option is selected (c - bottom). After adding two more landmarks in a similar fashion, the landmark on the chest has a few *AP* and unselected matching options in the GB (d - top), therefore a random landmark is selected (d - bottom). The next *closest unmatched landmark* has two matching options (e - top), both are *AP*, yet the one under the right armpit was previously selected. Therefore the landmark under the left armpit is chosen (e - bottom). After going through all the landmarks, the resulting chromosome is displayed (f - top). The final stage is changing the *match size* (as explained in 5.2.2), the resulting initial chromosome is displayed in (f - bottom).

5.2.2 Match size

The number of non-empty genes in a chromosome, i.e., $|\Pi(c)|$, implies how many landmarks are used for computing the dense map. The maximal number of non-empty genes is $m_{\max} = \min(m_1, m_2)$.

On the one hand, if $|\Pi(c)|$ is too small, it is unlikely that the computed map would be useful. Thus, we restrict the minimal match size to $m_{\min}$, and discard the constructed chromosome if it does not have enough non-empty genes.

On the other hand, we would like to have a variety of possible maps, and thus we force the

Algorithm 5.2 Create a random chromosome.

input : Landmark adjacencies $\mathcal{A}$, Landmark origins $\mathcal{O}$
output : A chromosome c

```
Pick a random prominent landmark l_1 ∈ S_1 ;                    // seed landmark
Add a random gene from G(l_1) to c ;              // first gene from gene bank
S_1^+ = {l_1}, S_1^- = S_1 \ {l_1} ;                          // initialize sets
while A(S_1^+) ∩ S_1^- ≠ ∅ do          // Adjacent unmatched landmarks exist
    Find closest matched/unmatched pair [l_1^+, l_1^-]
     g^+ = (l_1^+, c[l_1^+]) ;                                  // adjacent gene
    g = pickGene(c, g^+, l_1^-, {G(l_1^-), O(l_1^-)}) ;          // gene to add
    add g to c
     remove l_1^- from S_1^- ;                       // landmark was processed
    if g is not empty then
        add l_1^- to S_1^+
    end
end
add empty genes for all l_1^- ∈ S_1^- ;          // remaining unmatched landmarks
Function pickGene(c, g^+, l_1^-, S_2^-) is
    foreach l_2^- ∈ S_2^- do
        g = (l_1^-, l_2^-)
         if g is AP to g^+ and c ∪ g is valid then
            return g
        end
    end
    return empty gene
end
```

maximal number of matches to vary in size. Specifically, before a chromosome is constructed, we randomly select a match size $m_{\min} \leq \tilde{m} \leq m_{\max}$. If the chromosome assigns more than $\tilde{m}$ landmarks, we remove landmarks from c until we reach the required size. The removed landmarks are chosen randomly from the centers class, since we do not want to lose the salient extremities landmarks.

5.2.3 Population construction

We construct chromosomes as described previously until completing the initial population or a maximal iteration is reached. During construction, repeated chromosomes are discarded.

5.3 Fitness

Functional map optimization. The fitness of a chromosome c is computed by first using its *match* $\Pi(c)$ to extract a functional map $C_{\text{opt}}(\Pi(c))$. In fact, since the non-empty genes define a permutation mapping the subset $\mathcal{S}_1^+$ to $\mathcal{S}_2^+$ and vice versa, we use $\Pi(c)$ to compute functional maps in *both* directions. Specifically, we define:

$$\mathcal{E}^{\text{map}}(C, \Pi) = \alpha\, \mathcal{E}^{\Delta}(C) + \beta\, \mathcal{E}^{m}(C, \Pi), \tag{5.3}$$

where $\mathcal{E}^m, \mathcal{E}^\Delta$ are defined in equations (2.3) and (2.4), respectively, and we take α, β are the corresponding weights.

Given a chromosome c, we first compute

$$\hat{C}_{ij}(\Pi) = \underset{C_{ij}}{\operatorname{argmin}}\ \mathcal{E}^{\text{map}}(C_{ij}, \Pi), \tag{5.4}$$

where $\Pi = \Pi(c)$, and $(i, j) \in \{(1, 2), (2, 1)\}$. Note that this computation is very efficient, as these are two unconstrained linear least squares problems.

Functional map refinement. The optimized functional maps, $\hat{C}_{ij}$, can be efficiently refined by converting them to pointwise maps and back, such that they better represent valid pointwise maps. Specifically, we solve:

$$P_{ij}(\hat{C}_{ij}) = \underset{P}{\operatorname{argmin}}\ \|PX_j - P(\hat{C}_{ij})X_j\|_F^2, \tag{5.5}$$

where P is a binary row-stochastic matrix, and $P(C_{ij})$ is given in Equation (2.2). This problem can be solved efficiently using a kd-tree, by finding nearest neighbors in $\mathbb{R}^3$. Finally, the optimal functional maps in both directions are given by:

$$C_{ij}^{\text{opt}} = C(P_{ij}), \tag{5.6}$$

where $C(P_{ij})$ is given in Equation (2.1).

Functional map fitness. Finally, we evaluate the fitness of the maps with the *reversible elastic energy*:

$$\mathcal{E}^{\text{fit}}(C_{12}^{\text{opt}}, C_{21}^{\text{opt}}) = \gamma \sum_{ij} \mathcal{E}^{\text{elstc}}(C_{ij}^{\text{opt}}) + (1-\gamma)\, \mathcal{E}^{\text{rev}}(C_{12}^{\text{opt}},\ C_{21}^{\text{opt}}), \tag{5.7}$$

where $\mathcal{E}^{\text{elstc}},\ \mathcal{E}^{\text{rev}}$ are defined in section 2.4. While this objective is highly non-linear and non-convex, the fitness is never optimized directly, but only evaluated during the genetic algorithm, which is well suited for such complex objective functions.

To be precise, the term *fitness* commonly describes a value that is higher for better chromosomes, while in our case the energy is *lower* for better chromosomes. Thus, when we write *fittest* we refer to the chromosome with the lowest energy $\mathcal{E}^{\text{fit}}$.

5.4 Selection

In this process, individuals from the population are selected in order to pass their genes to the next generation. At each stage half of the population is selected to mate and create offspring. In order to select the individuals for mating we use a fitness proportionate selection (further discussed in [Bac96]). In our case the probability to select an individual for mating is proportional to 1 over its fitness so that the fitter individuals have a better chance of being selected.

5.5 Crossover

The chromosomes selected for the next generation undergo a *crossover* operation with probability p_{cross}. The crossover operator merges two different input chromosomes, c_A, c_B into two new chromosomes, $\bar{c}_A, \bar{c}_B$. To combine the input chromosomes in a geometrically consistent way, we again use adjacency preserving (AP) genes, as defined in section 5.1.

The crossover algorithm (demonstrated in Figure 5.3 and specified in Algorithm 5.3), is very similar to the initial chromosome creation. Here, however, rather than matching landmarks based on the gene bank alone, matching is mainly based on the parent chromosomes. First, we randomly pick a non empty gene from the parent chromosomes. The correspondence of the selected gene as assigned by c_A is the seed of $\bar{c}_A$, and correspondence of the same gene as assigned by c_B is the seed of $\bar{c}_B$. Then, each chromosome is constructed by iteratively adding valid AP genes from the parents. If an AP gene invalidates the child chromosome, we consider the gene bank options instead, and if these are not valid too, we pick a random gene that does not invalidate the child chromosome.

Figure 5.3 demonstrates the creation of $\bar{c}_A$ using the crossover algorithm. Both parent chromosomes appear in (a). As can be seen, the first chromosome maps the legs correctly, yet the arms are switched. The second parent switches the legs but the arms are correctly assigned. None of the chromosomes maps the head correctly.

First, a random non empty gene is copied from the first chromosome to the child chromosome (b). Then, the *closest unmatched landmark* is selected, and its matching options from both parent chromosomes are displayed (c - top). Since only the option from the first chromosome

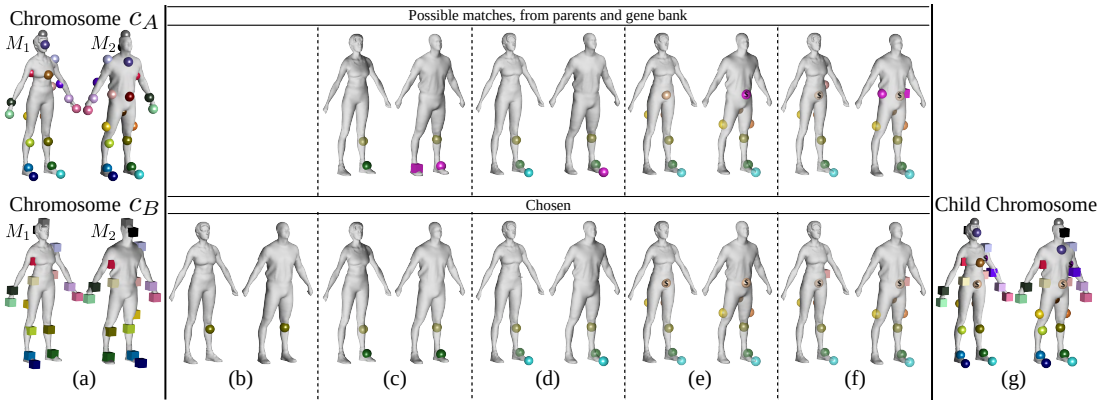


Figure 5.3: Crossover demonstration.

is *adjacency preserving (AP)*, this option is selected (c - bottom). In (d), the next *closest unmatched landmark* has only one matching option as only the first parent chromosome assigns correspondence to this landmark. Since this option *AP* and was not previously selected, it is selected. After matching a few more landmarks, a landmark on the stomach is chosen (e). This landmark has no matching options from either parent chromosomes, therefore, the possible matches are taken from the gene bank - marked with \$ sign (e - top). In this case there are two possible matching options, one is visible on the stomach, and another is on the back of M_2 and is not visible in the figure. Both options are *AP* and were not previously selected, so one is chosen randomly (e - bottom). The next *closest unmatched landmark* has two possible matches (f - top), both are *AP* and were not previously selected. Therefore, one is selected randomly (f - bottom). After repeating this process for all source landmarks, the resulting child chromosome is shown in (g). In the child chromosome the legs and hands are mapped correctly as it contains parts from both parent chromosomes. The head is still not mapped correctly. If in the next stage this chromosome is combined with one that maps the head correctly, their child could correctly match the entire shapes.

5.6 Mutation

After descendants are computed using the crossover operator, they can undergo different mutations, with varying probabilities ρ. Let c be the chromosome that undergoes mutation. We define three different types of mutation operators (inspired by [BM05]):

5.6.1 Growth.

With probability p_{growth}. Go over all the empty genes in random order and try to replace one, $(l_1, 0)$, by assigning it a corresponding landmark using the following two options.

FMap. Compute $P_{12}(\hat{C}_{12}(\Pi(c)))$, using equations (5.4) and (5.5) and set l_2 to the closet landmark on M_2 to $P_{12}(l_1)$. Add $g = (l_1, l_2)$ to c if $c \cup g$ is valid.

Algorithm 5.3 Crossover.

input : Landmark adjacencies $\mathcal{A}$, chromosomes c_A, c_B
output : Chromosomes $\bar{c}_A, \bar{c}_B$
$\mathcal{S}_{1A}^{+} = \mathcal{S}_1^{+}(\Pi(c_A))$, $\mathcal{S}_{1B}^{+} = \mathcal{S}_1^{+}(\Pi(c_B))$
$c_A = p(\bar{c}_A)$, $c_B = p(\bar{c}_B)$; // parents
Pick a random landmark $l_1 \in \mathcal{S}_{1A}^{+} \cap \mathcal{S}_{1B}^{+}$; // seed landmark
foreach $\bar{c} \in \{\bar{c}_A, \bar{c}_B\}$ **do**
  Copy the l_1 gene from $p(\bar{c})$ to $\bar{c}$; // first gene
  $\mathcal{S}_1^{+} = \{l_1\}, \mathcal{S}_1^{-} = \mathcal{S}_1 \setminus \{l_1\}$; // initialize sets
  while $\mathcal{S}_1^{-} \neq \emptyset$ **do** // unmatched landmarks exist
    if $\mathcal{A}(\mathcal{S}_1^{+}) \cap \mathcal{S}_1^{-} \neq \emptyset$ **then** // Adjacent landmarks exist
      Find closest matched/unmatched pair $[l_1^{+}, l_1^{-}]$
      g^{+} = $(l_1^{+}, \bar{c}[l_1^{+}])$; // adjacent gene
      $\mathcal{P}$ = $\{(l_1^{-}, c_A[l_1^{-}]), (l_1^{-}, c_B[l_1^{-}])\}$; // parent genes
      g = pickGene($\bar{c}, g^{+}, l_1^{-}, \{\mathcal{P}, \mathcal{G}(l_1^{-})\}$) ; // gene to add
    else
      g = random gene from $\Pi(p(\bar{c}))$ such that $\bar{c} \cup g$ is valid
      if $g = \emptyset$ **then** // no more parent genes
        break
      end
      l_1^{-} = source landmark of g
    end
    add g to $\bar{c}$
    remove l_1^{-} from $\mathcal{S}_1^{-}$; // landmark was processed
    if g *is not empty* **then**
      add l_1^{-} to $\mathcal{S}_1^{+}$
    end
  end
end
add empty genes for all $l_1^{-} \in \mathcal{S}_1^{-}$; // remaining unmatched landmarks

Gene Bank. If the previous attempt failed, set g to a random gene from the gene bank $\mathcal{G}(l_1)$, and add it to c if $c \cup g$ is valid.

5.6.2 Shrinkage.

With probability $p_{shrinkage}$. Randomly select n_{sh} *centers* landmarks from $\mathcal{S}_1^+(\Pi(c))$. Replace each one of the corresponding genes with an empty gene, such that n_{sh} new chromosomes are obtained. Each of these has a single new empty gene. The result is the fittest chromosome among them (including the original before shrinking).

5.6.3 Functional map guidance.

With probability $p_{FMguidance}$. Compute $P_{12}(\hat{C}_{12}(\Pi(c)))$, using equations (5.4) and (5.5). For each landmark $l_1 \in \mathcal{S}_1^+(\Pi(c))$ set l_2 to the closet landmark on M_2 to $P_{12}(l_1)$. If injectivity is violated, i.e. $c[i] = c[j]$, if i is a center and j is in another category, set $c[i] = 0$ and $c[j]$ as computed by the functional map guidance. Otherwise, randomly keep one of them and set the other one to 0. We prioritize maxima and minima as they are often more salient.

5.7 Convergence

We stop the iterations when one of two conditions is met. Either, the fittest chromosome remains unchanged for a set amount of iterations, meaning the population converged, or when a maximal iteration number is reached, in which case the best solution from the last generation is provided. The latter condition affects the algorithm's worst case running time as it bounds the maximal number of iterations performed when the algorithm doesn't converge.

5.8 Timing

The most computationally expensive part of the algorithm is computing the fitness of each chromosome. While computing the functional map and evaluating the elastic energy of one chromosome is fast, this computation is done for all the offspring in each iteration, and is therefore expensive. Our method is implemented in MATLAB. On a desktop machine with an Intel Core i7 processor, for meshes with 5K vertices, computation of a functional map for a single chromosome takes 0.035 seconds and the computation of the elastic energy takes 0.002 seconds. The average amount of iterations it took for the algorithm to converge is 225 and the average total computation time is 10.5 minutes.

Chapter 6

Results

6.1 Datasets and Evaluation

Our method computes a sparse correspondence and a functional map that can be used as input to existing semi-automatic methods such as [ESBC19], that we use in this paper. We demonstrate the results of our method on two datasets with different properties.

The SHREC'07 dataset [GBP07] contains a variety of non-isometric shapes as well as ground truth sparse correspondence between manually selected landmarks. This dataset is suitable to demonstrate the advantages of our method, since we address the highly non-isometric case. The recent dataset SHREC'19 [DSLR19] contains shapes of the same semantic class but different topologies, that we use to demonstrate our results in such challenging cases.

Quantitatively evaluating sparse correspondence on this dataset is challenging, since the given sparse ground truth does not necessarily coincide with the computed landmarks. We therefore quantitatively evaluate the results after post processing, where we use the same post processing for all methods (if a method produces sparse correspondence we compute functional maps using these landmarks and run RHM [ESBC19] to extract a pointwise map, and if a method produces a pointwise map we apply RHM directly). Since we initialize RHM with a functional map or a dense map, we use the Euclidean rather than the geodesic embedding that was used in their paper, that is only needed when the initialization is very coarse. We use the evaluation protocol suggested by Kim et al. [KLF11], where the x axis is a geodesic distance between a ground truth correspondence and a computed correspondence, and the y axis is the percentage of correspondences with less than x error. We also allow symmetries, as suggested by Kim et al. [KLF11], by computing the error w.r.t. both the ground truth correspondence and the symmetric map (that is given), and using the map with the lower error as ground truth for comparison.

We qualitatively evaluate our results using color and texture transfer. We visualize sparse correspondence by showing corresponding landmarks in the same color. To visualize functional maps we show a smooth function, visualized by color coding on the target mesh, and transfer it to the source using the functional map. We visualize pointwise maps by texture transfer.

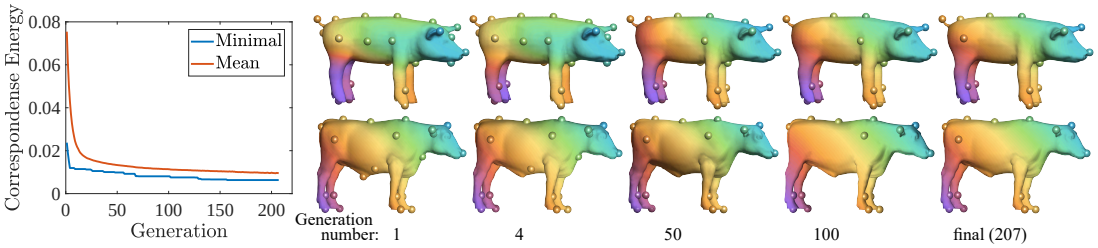


Figure 6.1: Convergence of the genetic algorithm. (left) The energy of the fittest offspring and the average energy of the population decrease until convergence indicating the improvement throughout the generation. (right) The fittest offspring at different generations, improving until reaching the right match between the shapes.

6.2 Population Evolution and Convergence

Figure 6.1 shows the evolution of the population in the pig-cow example. We plot the mean energy of a chromosome as a function of the generation number, as well as the minimal energy (of the fittest chromosome). We also visualize the functional map and the landmarks of the fittest chromosome throughout the algorithm, which is inaccurate at the first iteration (the back legs are mapped relatively correctly, but the front legs are reversed), and gets more and more accurate as the population evolves, until finally the map is semantic.

6.3 Quantitative and Qualitative Comparisons

Since our algorithm is designed to handle significant non isometric distortion, we demonstrate our results on shapes from SHREC'07 [GBP07], a dataset that contains a variety of highly non isometric shapes. Due to the geometric variability of the shapes in this dataset, dense ground truth is not available, but sparse ground truth is given.

We compare our method with a few fully automatic methods for sparse correspondence: "Tight Relaxation" by Kezurer et al. [KKBL15],

DS++ by Dym et al. [DML17], and the recent method by Sahillioglu et al. [Sah18] that also used a genetic algorithm (GA+AS, AS stands for Adaptive Sampling which they use for improved results). We only compare with fully automatic methods, since existing semi automatic methods require additional user input that is not always available. For comparisons that require dense correspondence we apply the same post processing we used for our method on the output sparse correspondence of the other methods, i.e. we compute a functional map as described in section 5.3 and use it as input to the sparse-to-dense post processing method [ESBC19].

Figure 6.3 visualizes the output sparse correspondence of each of these methods on shapes from SHREC'07 [GBP07], as well as the functional maps that are visualized by transferring smooth functions and color coding. Our method consistently generates semantic results on shapes from various classes.

After the post processing that extracts a dense pointwise map from the functional maps, we can quantitatively compare the results using the protocol suggested by Kim et al. [KLF11]

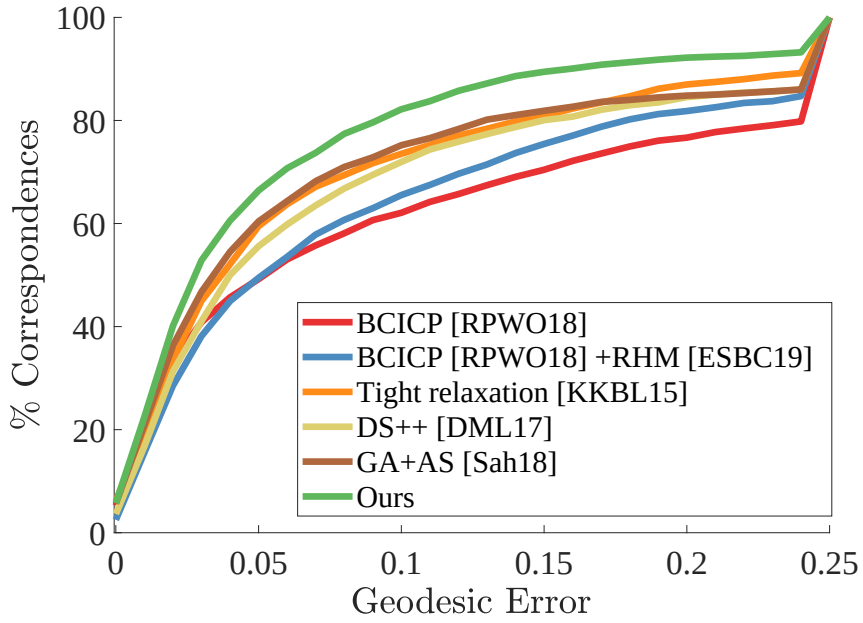


Figure 6.2: Quantitative comparison between our final results and fully automatic state of the art methods [KKBL15, DML17, Sah18, RPWO18] (after applying the same post processing [ESBC19] on all the methods to extract a dense map), using the evaluation protocol suggested by Kim et al. [KLF11]. The graph shows that our results are closer to the ground truth than other methods.

as described in section 6.1. We compare the dense pointwise maps to a recent method by Ren et al. [RPWO18] (BCICP), whose output is a dense map. Since Ren et al. [RPWO18] computes vertex-to-vertex maps, we apply the post processing [ESBC19] on their results as well (BCICP+RHM).

The results are shown in Figure 6.2. Our method outperforms all previous results. In addition we visualize the final dense maps using texture transfer in Figure 6.4.

6.4 Correspondence between Shapes of Different Genus

Our method is not restricted to genus zero, and works for shapes of different topology as well. Figure 1.1 shows our results for two shapes of hands from SHREC'19 [DSLR19]. The left hand shape is of genus 1 (the upper part of the middle and ring finger are connected), and the right hand shape is of genus 2 (the tips of the index finger and the thumb are connected, as well as the middle of the index and the middle finger). Figure 6.5 shows our sparse correspondence and functional map for two shapes of cups from SHREC'07 [GBP07], both genus 1, and two human shapes from SHREC'19 [DSLR19] with genus 1 and genus 0. Note that in all these cases, despite the difficult topological issues, our approach found a meaningful map.

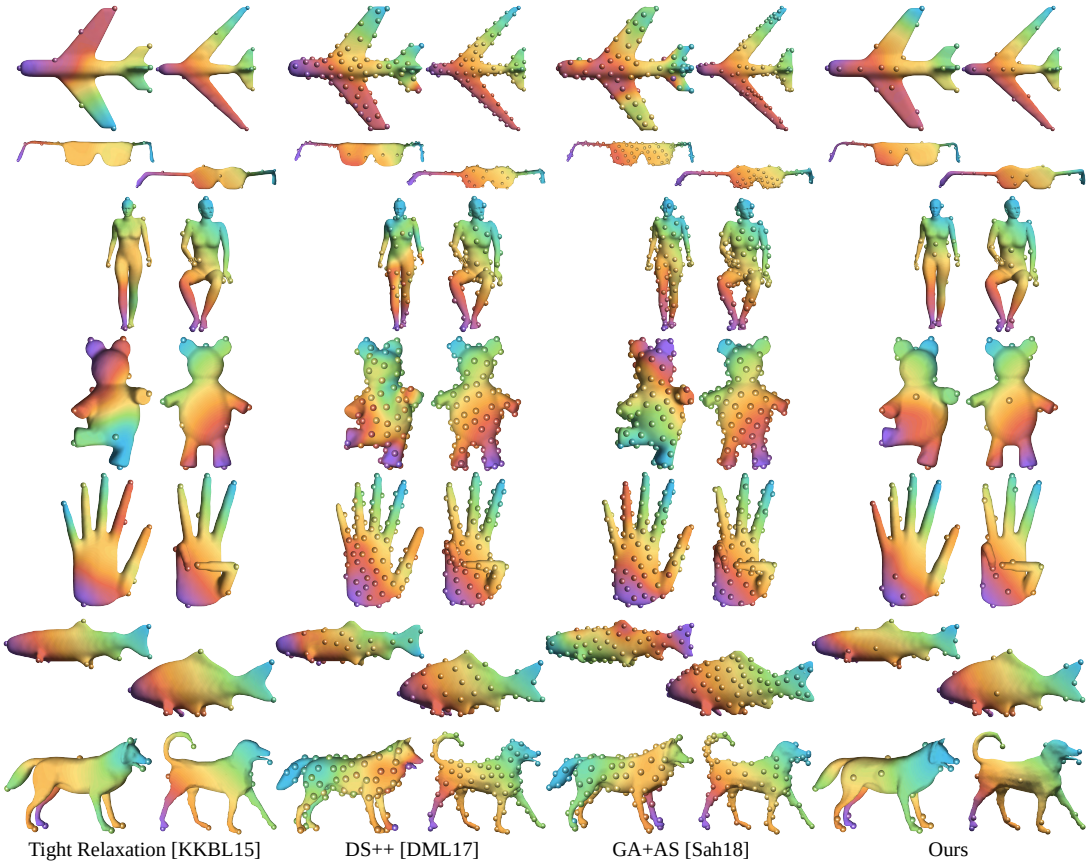


Figure 6.3: Qualitative comparison between the results of our genetic algorithm, i.e. a sparse correspondence and a functional map, and the results of automatic state of the art methods [KKBL15, DML17, Sah18] for sparse correspondence. The functional map is visualized by color transfer (see text for details), and the sparse correspondence is visualized by landmarks with corresponding colors.

6.5 Stability

Since our algorithm has random components, we tested the stability of our method. We ran our method 100 times on a pair of shapes from the SHREC'07 dataset [GBP07], and used the protocol suggested by Kim et al. [KLF11] to quantitatively evaluate the results (after applying post processing [ESBC19] to extract a dense map). The results are shown in Figure 6.6. The best and worst results are determined by the total geodesic error with respect to the given sparse ground truth. The best result has at most 0.07 maximal relative geodesic error, and the curve of all the results is very similar to the best result, indicating that the bad results were an outlier. In addition, the worst result is still comparable with other methods.

6.6 Limitations

We evaluate the quality of each sparse correspondence using the elastic energy of the functional map between the shapes. When representing a mesh in the functional space, in order to represent

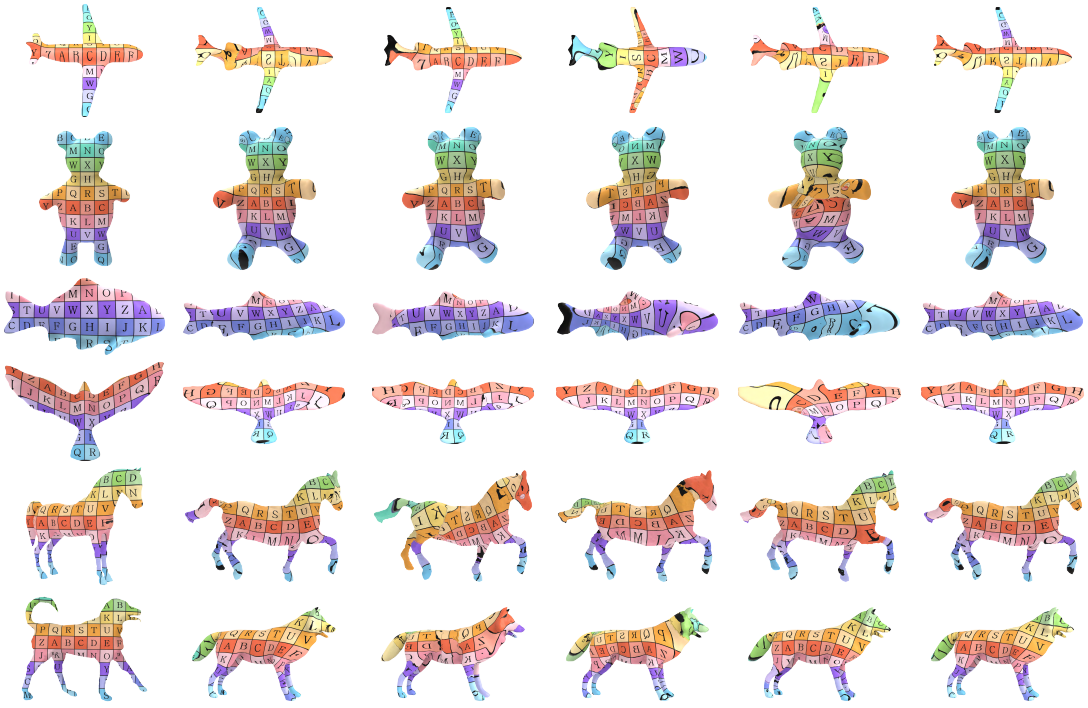


Figure 6.4: Qualitative comparison between our final results and fully automatic state of the art methods [KKBL15, DML17, Sah18, RPWO18] (after applying the same post processing [ESBC19] on all the methods to extract a dense map). The pointwise maps are visualized using texture that is computed on the target mesh (left) and transferred to the source using each method.

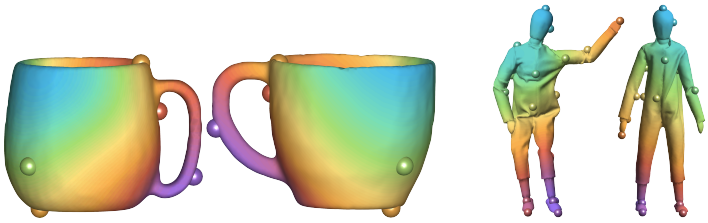


Figure 6.5: Our results on shapes of genus 1 (cups) and two shapes with different genus.

thin parts, a large number of eigenfunctions is required. When using a relatively small number of eigenfunctions, the thin parts become even thinner and can be reduced to a line or even a point. Therefore, when working with meshes with thin parts, wrong correspondence that distorts those parts can still have a low elastic energy relatively to correct correspondence (that also distorts these regions because of the reduces basis). Such a case is displayed in Figure 6.7. Both the body and the legs of the ants are very narrow, and the resulting best match switches between both the back legs of the ant and rotates the tail.

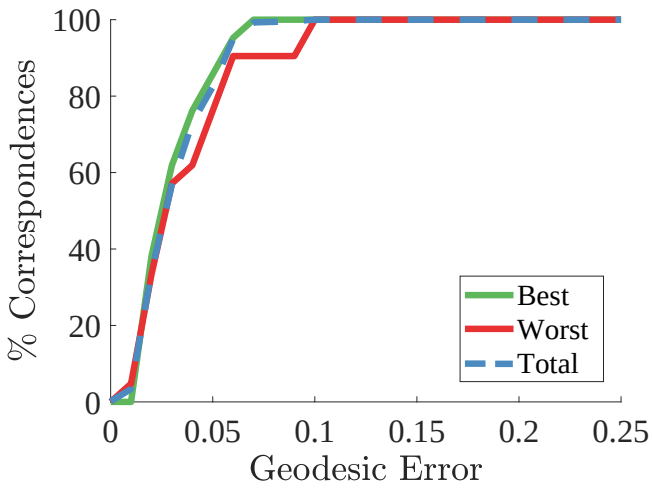


Figure 6.6: Cumulative geodesic error of the best, worst and total results of our algorithm, after we ran it 100 times on the same pair of shapes.

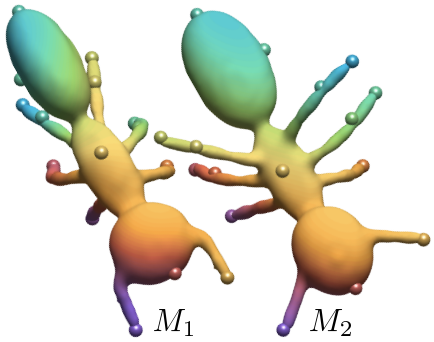


Figure 6.7: A failure case where the back legs are switched and the tail is rotated.

Chapter 7

Conclusion

We present an approach for computing an automatic correspondence between shapes of the same semantic class which are non-isometric. We leverage a genetic algorithm for the combinatorial optimization of a small set of automatically computed landmarks, and use a fitness energy that is based on extending the sparse landmarks to a functional map. As a result, we can achieve meaningful maps, that outperform existing state-of-the-art automatic methods, and can additionally handle difficult cases of different topology of the source and target.

We believe that our approach can be generalized in a few ways. First, the decomposition of the automatic mapping computation problem into combinatorial and continuous problems mirrors other tasks in geometry processing which were handled in a similar manner, such as quadrangular remeshing. It is interesting to investigate whether additional analogs exist between these seemingly unrelated problems. Furthermore, it is intriguing to consider what other problems in shape analysis can benefit from genetic algorithms. One potential example is map synchronization for map collections, where the choice of cycles to synchronize is also combinatorial.

Appendix A

Appendix

A.1 Parameters

All the parameters are fixed in all our experiments. In addition, they are all unitless. We chose the values specified here experimentally, yet our results are not sensitive to the parameters. Many of them stem from the fact that the genetic algorithm inherently contains many parameters.

A.1.1 Landmark Computation

Centers. To compute the landmarks classified as centers (eq. (4.1)), we use $N = 30$ eigenfunctions, as chosen in [CMS18].

Filtering. We filter close landmarks according to the initial distance of $d_\epsilon = 0.08$, where the distance is on the normalized shape. In addition, we set the maximal amount of landmarks to $m_{\text{max}} = 35$. We saw that for a variety of shapes, 35 landmarks with the specified minimal distance covered the shape features well and gave a large enough subset of matching landmarks for our fitness function.

Adjacent landmarks. We use the geodesic distance of $d_{\text{adj}} = 0.3$ to define adjacent landmarks. This distance is large enough to match highly non-isometric shapes but still allows to construct consistent chromosomes.

A.1.2 Genetic Non Isometric Maps

Gene bank. We set the WKS distance to $\epsilon_{\text{wks}} = 0.2$.

Match size The minimal match size is $m_{\min} = \frac{2}{3}m_{\max}$ and the maximal match size is $m_{\max} = \min(m_1, m_2)$ where m_i is the number of landmarks on M_i.

Population construction The initial population contains 400 chromosomes, based on various experiments with different population's sizes and according to [RG02].

Functional map optimization. To construct the functional map we use $k_s = 30, k_t = 60$ and $\alpha = 1, \beta = 100$.

Functional map fitness. For the elastic energy, we set $\gamma = 5 \cdot 10^{-4}$ similarly to [EHA+19].

Genetic operators. For the operators, we use the rates $p_{cross}= 0.75, p_{growth}= 0.05, p_{shrinkage}= 0.1, p_{FMguidance}= 0.05$. The crossover is the main operator, therefore its rate is much higher than the mutations (as common in genetic algorithms).

Convergence. We stop if the fittest chromosome remains unchanged for 70 iterations or when a maximal iteration number of 700 iterations is reached.

Bibliography

[AL15] Noam Aigerman and Yaron Lipman. Orbifold Tutte embeddings. *ACM Transactions on Graphics (TOG)*, 34, 2015.

[AL16] Noam Aigerman and Yaron Lipman. Hyperbolic orbifold Tutte embeddings. *ACM Transactions on Graphics (TOG)*, 35, 2016.

[APL15] Noam Aigerman, Roi Poranne, and Yaron Lipman. Seamless surface mappings. *ACM Transactions on Graphics (TOG)*, 34(4):72, 2015.

[ASC11] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 2011.

[Auw07] Surapong Auwatanamongkol. Inexact graph matching using a genetic algorithm for image recognition. *Pattern Recognition Letters*, 28(12):1428–1437, 2007.

[Bac96] Thomas Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[BLM95] Bir Bhanu, Sungkee Lee, and John Ming. Adaptive image segmentation using a genetic algorithm. *IEEE Transactions on systems, man, and cybernetics*, 25(12):1543–1567, 1995.

[BM05] Alexandru Horia Brie and Philippe Morignot. Genetic planning using variable length chromosomes. In *ICAPS*, pages 320–329, 2005.

[BPGK06] Mario Botsch, Mark Pauly, Markus H Gross, and Leif Kobbelt. PriMo: coupled prisms for intuitive surface modeling. In *Proc. Eurographics Symposium on Geometry Processing*, pages 11–20, 2006.

[CMS18] Xiuyuan Cheng, Gal Mishne, and Stefan Steinerberger. The geometry of nodal sets and outlier detection. *Journal of Number Theory*, 185:48–64, 2018.

[CTL04] Chi Kin Chow, Hung Tat Tsui, and Tong Lee. Surface registration using a dynamic genetic algorithm. *Pattern recognition*, 37(1):105–117, 2004.

[CWH97] Andrew DJ Cross, Richard C Wilson, and Edwin R Hancock. Inexact graph matching using genetic search. *Pattern Recognition*, 30(6):953–970, 1997.

[dBDFN16] Maya de Buhan, Charles Dapogny, Pascal Frey, and Chiara Nardoni. An optimization method for elastic shape matching. *C. R. Math. Acad. Sci. Paris*, 354(8):783–787, 2016.

[DML17] Nadav Dym, Haggai Maron, and Yaron Lipman. Ds++: A flexible, scalable and provably tight relaxation for matching problems. *ACM Transactions on Graphics (TOG)*, 36(6), 2017.

[DSLR19] Roberto Dyke, Caleb Stride, Yu-Kun Lai, and Paul L. Rosin. Shrec 2019. `https://shrec19.cs.cf.ac.uk/`, 2019.

[EBC17] Danielle Ezuz and Mirela Ben-Chen. Deblurring and denoising of maps between shapes. In *Computer Graphics Forum*, volume 36, pages 165–174. Wiley Online Library, 2017.

[EHA+19] Danielle Ezuz, Behrend Heeren, Omri Azencot, Martin Rumpf, and Mirela Ben-Chen. Elastic correspondence between triangle meshes. In *Computer Graphics Forum*, volume 38, 2019.

[ERGB16] Davide Eynard, Emanuele Rodola, Klaus Glashoff, and Michael M Bronstein. Coupled functional maps. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 399–407. IEEE, 2016.

[ESBC19] Danielle Ezuz, Justin Solomon, and Mirela Ben-Chen. Reversible harmonic maps between discrete surfaces. *ACM Transactions on Graphics*, 2019.

[ESKBC17] Danielle Ezuz, Justin Solomon, Vladimir G Kim, and Mirela Ben-Chen. Gwcnn: A metric alignment layer for deep shape analysis. In *Computer Graphics Forum*, volume 36, pages 49–57, 2017.

[GBKS18] Anne Gehre, M Bronstein, Leif Kobbelt, and Justin Solomon. Interactive curve constrained functional maps. In *Computer Graphics Forum*, volume 37, pages 1–12. Wiley Online Library, 2018.

[GBP07] Daniela Giorgi, Silvia Biasotti, and Laura Paraboschi. SHREC: Shape retrieval contest: Watertight models track, 2007.

[HO17] Ruqi Huang and Maks Ovsjanikov. Adjoint map representation for shape analysis and matching. In *Computer Graphics Forum*, volume 36, pages 151–163, 2017.

[Hol92] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[HRS+14] Behrend Heeren, Martin Rumpf, Peter Schröder, Max Wardetzky, and Benedikt Wirth. Exploring the geometry of the space of shells. *Comput. Graph. Forum*, 33(5):247–256, 2014.

[HRWW12] Behrend Heeren, Martin Rumpf, Max Wardetzky, and Benedikt Wirth. Time-discrete geodesics in the space of shells. *Comput. Graph. Forum*, 31(5):1755–1764, 2012.

[IRS18] José A. Iglesias, Martin Rumpf, and Otmar Scherzer. Shape-aware matching of implicit surfaces based on thin shell energies. *Found. Comput. Math.*, 18(4):891–927, 2018.

[KKBL15] Itay Kezurer, Shahar Z Kovalsky, Ronen Basri, and Yaron Lipman. Tight relaxation of quadratic matching. In *Computer Graphics Forum*, volume 34, pages 115–128, 2015.

[KLF11] Vladimir G Kim, Yaron Lipman, and Thomas Funkhouser. Blended Intrinsic Maps. *ACM Transactions on Graphics (TOG)*, 30, 2011.

[LDRS05] Nathan Litke, Mark Droske, Martin Rumpf, and Peter Schröder. An image processing approach to surface matching. In *Proc. Eurographics Symposium on Geometry Processing*, pages 207–216, 2005.

[MB00] Ujjwal Maulik and Sanghamitra Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern recognition*, 33(9):1455–1465, 2000.

[MCSK+17] Manish Mandad, David Cohen-Steiner, Leif Kobbelt, Pierre Alliez, and Mathieu Desbrun. Variance-minimizing transport plans for inter-surface mapping. *ACM Transactions on Graphics*, 36:14, 2017.

[MDK+16] Haggai Maron, Nadav Dym, Itay Kezurer, Shahar Kovalsky, and Yaron Lipman. Point registration via efficient convex relaxation. *ACM Transactions on Graphics (TOG)*, 35(4), 2016.

[MDW08] Brent C Munsell, Pahal Dalal, and Song Wang. Evaluating shape correspondence for statistical shape analysis: A benchmark study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):2023–2039, 2008.

[NO17] Dorian Nogneng and Maks Ovsjanikov. Informative descriptor preservation via commutativity for shape matching. In *Computer Graphics Forum*, volume 36, pages 259–267, 2017.

[OBCS+12] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)*, 31(4), 2012.

[OCB+16] Maks Ovsjanikov, Etienne Corman, Michael Bronstein, Emanuele Rodolà, Mirela Ben-Chen, Leonidas Guibas, Frederic Chazal, and Alex Bronstein. Computing and processing correspondences with functional maps. In *SIGGRAPH ASIA 2016 Courses*, page 9. ACM, 2016.

[OM97] Ender Ozcan and Chilukuri K Mohan. Partial shape matching using genetic algorithms. *Pattern Recognition Letters*, 18(10):987–992, 1997.

[PBDSH13] Daniele Panozzo, Ilya Baran, Olga Diamanti, and Olga Sorkine-Hornung. Weighted averages on surfaces. *ACM Transactions on Graphics (TOG)*, 32(4):60, 2013.

[PRB+13] P Victer Paul, A Ramalingam, R Baskaran, P Dhavachelvan, K Vivekanandan, R Subramanian, and VSK Venkatachalapathy. Performance analyses on population seeding techniques for genetic algorithms. *International Journal of Engineering and Technology (IJET)*, 5(3):2993–3000, 2013.

[RG02] STANLEY GOTSHALL BART Rylander and S Gotshall. Optimal population size and the genetic algorithm. *Population*, 100(400):900, 2002.

[RPWO18] Jing Ren, Adrien Poulenard, Peter Wonka, and Maks Ovsjanikov. Continuous and orientation-preserving correspondences via functional maps. *ACM Transactions on Graphics (TOG)*, 37(6), 2018.

[SA07] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proc. Eurographics Symposium on Geometry Processing*, pages 109–116, 2007.

[Sah18] Yusuf Sahillioğlu. A genetic isometric shape correspondence algorithm with adaptive sampling. *ACM Trans. Graph.*, 37(5), October 2018.

[SBB05] Luciano Silva, Olga Regina Pereira Bellon, and Kim L Boyer. Precision range image registration using a robust surface interpenetration measure and enhanced genetic algorithms. *IEEE transactions on pattern analysis and machine intelligence*, 27(5):762–776, 2005.

[SP04] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics (TOG)*, 23, 2004.

[SPKS16] Justin Solomon, Gabriel Peyré, Vladimir G Kim, and Suvrit Sra. Entropic metric alignment for correspondence problems. *ACM Transactions on Graphics (TOG)*, 35(4):72, 2016.

[TCL+13] Gary KL Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C Langbein, Yonghuai Liu, David Marshall, Ralph R Martin, Xian-Fang Sun, and Paul L Rosin. Registration of 3d point clouds and meshes: a survey from rigid to nonrigid. *IEEE Transactions on Visualization and Computer Graphics*, 19(7):1199–1217, 2013.

[UM93] Ron Unger and John Moult. Genetic algorithms for protein folding simulations. *Journal of molecular biology*, 231(1):75–81, 1993.

[VKZHCO11] Oliver Van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. In *Computer Graphics Forum*, volume 30, pages 1681–1707, 2011.

[VLR+17] Matthias Vestner, Roee Litman, Emanuele Rodola, Alex Bronstein, and Daniel Cremers. Product manifold filter: Non-rigid shape correspondence via kernel density estimation in the product space. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6681–6690. IEEE, 2017.

[WHG07] Yingzi Wei, Yulan Hu, and Kanfeng Gu. Parallel search strategies for tsps using a greedy genetic algorithm. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 3, pages 786–790. IEEE, 2007.

[WSSC11] Thomas Windheuser, Ulrich Schlickewei, Frank R Schmidt, and Daniel Cremers. Geometrically consistent elastic matching of 3d shapes: A linear programming solution. In *Proc. IEEE International Conference on Computer Vision*, pages 2134–2141, 2011.

[YAF99] Sameh M Yamany, Mohamed N Ahmed, and Aly A Farag. A new genetic-based technique for matching 3-d curves and surfaces. *Pattern Recognition*, 32(10):1817–1820, 1999.

[ZSCO+08] H. Zhang, A. Sheffer, D. Cohen-Or, Q. Zhou, O. van Kaick, and A. Tagliasacchi. Deformation-driven shape correspondence. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1431–1439. Eurographics Association, 2008.

קומבינטורית ויכולים לשמש לביצוע אופטימיזציה עבור פונקציות מטרה כלליות. למרות זאת, למיטב ידיעתנו, בתחומינו השתמשו בהם רק לביצוע התאמה בין צורות איזומטריות ולא עבור צורות לא איזומטריות כפי שאנו מראים בעבודה זו.

אלגוריתמים גנטיים מבוססים על תורת האבולוציה והברירה הטבעית. האלגוריתמים כוללים תחילה יצירה של אוכלוסייה התחלתית של פתרונות אפשריים. לאחר מכן, הערכת איכות כל הפתרונות ובחירות הטובים ביותר, אותם ממזגים ומשנים על מנת לייצר אוכלוסייה חדשה של פתרונות. כך, כל דור חדש כולל פתרונות טובים יותר עד להתכנסות לפתרון הטוב ביותר.

השיטה שלנו מקבלת כקלט שתי צורות המיוצגות כמשטחים דיסקרטיים משולשים. כשלב מקדים אנו מחשבים סט נקודות עניין על כל אחת מן הצורות. שתי קבוצות הנקודות פזורות על כל הצורות וגדולות מספיק כך שנוכל למצוא תת קבוצה של נקודות שתואמת על שתי הצורות. לאחר מכן, אנחנו משתמשים באלגוריתם הגנטי על מנת למצוא את הקבוצה התואמת וההתאמה עצמה. כל פתרון אפשרי הוא פרמוטציה חלקית ומתאר התאמה חד־חד ערכית בין תתי קבוצות של נקודות כנ"ל. שימוש באלגוריתם גנטי מאפשר לנו לבצע אופטימיזציה עבור פונקציית מטרה מסובכת שאינה לינארית ואינה קמורה ותלויה גם בפרמוטציה של סט נקודות העניין וגם בתכונות הדיפרנציאליות של המיפוי המלא שהורחב מנקודות העניין התואמות. אנחנו משתמשים בפונקציית מטרה שמבוססת על האנרגיה האלסטית של הצורות שעברו דפורמציה אחת לשניה. האנרגיה האלסטית שימשה בעבודה קודמת להתאמה בין צורות לא איזומטריות והשיגה תוצאות טובות כאשר הייתה ידועה התאמה בין סט נקודות עניין. בנוסף, פיתחנו אופרטורים לאלגוריתם הגנטי שמבוססים על הגיאומטריה של הצורות ומאפשרים למזג ולשנות התאמות קיימות בין נקודות עניין כך שנוצרות התאמות חדשות אפשריות טובות יותר.

אנחנו מראים שהאלגוריתם שלנו מתכנס להתאמה דלילה שניתנת להרחבה למיפוי פונקציונלי וכן למיפוי מלא בין הצורות, וכן שהשיטה שלנו משיגה ביצועים טובים בהשוואה לשיטות אחרות. בנוסף, אנחנו מראים תוצאות איכותיות של המיפוי הפונקציונלי וכן תמונות טקסטורה.

תקציר

התאמה בין צורות הינה משימה בסיסית וחשובה בניתוח צורות: בהינתן שתי צורות, המטרה היא לחשב התאמה סמנטית בין נקודות על הצורות. ההתאמה בין הצורות נחוצה לניתוח משותף שלהם, דבר שנפוץ באפליקציות רבות בגרפיקה ממוחשבת כגון העברת דפורמציות והעברת סגנון, ניתוח סטטיסטי, סיווג צורות ועוד.

ניתן לחלק את משימת ההתאמה בין צורות לשלושה סוגים עיקריים על פי מאפייני הצורות: התאמה בין צורות קשיחות, התאמה בין צורות איזומטריות והתאמה בין צורות לא איזומטריות ששייכות לאותה מחלקה סמנטית. מציאת התאמה בין צורות קשיחות, כלומר בין צורות הנבדלות זו מזו רק על ידי סיבוב והזזה, נחקרה לעומק בעבר ונחשבת פשוטה יותר מאשר המקרים אחרים כיוון שכמות דרגות החופש הינה קטנה ומרחב הטרנספורמציות האפשריות בין הצורות קל לייצוג. הסוג השני הוא התאמה בין צורות שאינן קשיחות אך כן איזומטריות, לדוגמה עבור אותו אוביקט הנמצא בפוזות שונות. מציאת ההתאמה במקרה זה קלה יותר מאשר במקרה הלא איזומטרי, כיוון שבמקרה האיזומטרי ישנו קריטריון איכות ברור עבור המיפוי והוא שימור מרחקים גאודזיים. הסוג השלישי והמאתגר ביותר הוא כאשר שתי הצורות שייכות לאותה מחלקה סמנטית אך אינן איזומטריות. האתגר העיקרי במקרה הזה הוא שאין הגדרה אחת עבור התאמה טובה ולכן, אפילו שלאדם משימת ההתאמה הינה קלה, ניסוח מתמטי מדויק של הבעיה הינו קשה ועל אלגוריתם שמוצא התאמה בין הצורות להתאים בין נקודות על הצורות בצורה סמנטית וכן להפחית מדד עיוות מקומי כלשהו.

ניתן לפרק את תהליך מציאת ההתאמה בין צורות לא איזומטריות לשני שלבים. תחילה, יש למצוא התאמה בין מספר מועט של נקודות עניין על הצורות. לאחר מכן, אם סט הנקודות מספיק אינפורמטיבי, ניתן להרחיב את ההתאמה להתאמה מלאה בין הצורות באופן עקבי וחלק. הבעיה הראשונה היא קומבינטורית מטבעה ודורשת חישוב פרמוטציה של סט חלקי של נקודות העניין, ואילו הבעיה השנייה היא רציפה ודורשת הגדרה וחישוב של תכונות דיפרנציאליות ומקומיות של המיפוי. בעוד שקיימות עבודות רבות שמתמקדות בבעיה השנייה ומפיקות תוצאות טובות גם עבור צורות לא איזומטריות, עבור החלק הראשון – מציאת התאמה דלילה בין קבוצת נקודות עניין – העבודות הקיימות מוגבלות לצורות איזומטריות או לכמות נקודות קטנה מאוד ולכן העבודה שלנו מתמקדת בפתרון חלק זה.

אנחנו מציעים שיטה למציאת התאמה דלילה בין סט נקודות עניין והרחבתה למיפוי מלא בין צורות באמצעות אלגוריתם גנטי. אלגוריתמים גנטיים משמשים לביצוע אופטימיזציה

המחקר בוצע בהנחייתה של פרופסור מירלה בן־חן, בפקולטה להנדסת חשמל.

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי.

אלגוריתם גנטי להתאמה אוטומטית בין משטחים לא איזומטריים

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים בהנדסת חשמל

מיכל אדלשטיין

הוגש לסנט הטכניון --- מכון טכנולוגי לישראל
אלול התשע"ט חיפה ספטמבר 2019

אלגוריתם גנטי להתאמה אוטומטית בין משטחים לא איזומטריים

מיכל אדלשטיין