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

Robust Shape Collection Matching and Correspondence from Shape Differences

Aharon Cohen

Robust Shape Collection Matching and Correspondence from Shape Differences

Research Thesis

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

Aharon Cohen

Submitted to the Senate
of the Technion — Israel Institute of Technology
Shvat 5780 Haifa February 2020

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 collections are widely used in many geometry processing and computer graphics applications. Such collections can be obtained by deforming a given 3D model or by sampling a 3D animation. Given two shape collections, e.g. two characters in similar poses, often rises the need to match the semantically corresponding shapes. This *matching* can assist in transferring information between the two collections. For instance, transferring shape annotations in order to allow pose labeling. A more common challenge is to automatically find the pointwise map, also known as *correspondence*, between non-isometric shapes, e.g. the two different non-isometric shapes from the two collections. In general, it is a very difficult problem, that has been tackled by many different approaches and often requires additional input such as landmarks or descriptors. This pointwise mapping can assist in transferring pointwise data, allowing for example texture transfer between non-isometric shapes.

We propose a method to automatically *match* two shape collections with a similar shape space structure, and compute the inter-maps between the collections. Given the intra-maps in each collection, which are often easier to compute since the shapes within the collection are isometric, we extract the corresponding *shape difference operators*, and use them to construct an embedding of the shape space of each collection. We then align the two shape spaces, and use the knowledge gained from the alignment to compute the inter-maps by formulating an appropriate optimization problem.

Unlike existing approaches for collection alignment, our method is applicable to small and large collections alike, and requires no parameter tuning. Furthermore, unlike most approaches for non-isometric correspondence, our method uses solely the *variation* within the collection to extract the inter-maps, and therefore does not require landmarks, descriptors or any additional input. We demonstrate that we achieve high matching accuracy rates, and compute high quality maps on non-isometric shapes, which compare favorably with automatic state-of-the-art methods for non-isometric shape correspondence. Furthermore, we show that in some cases, it is possible to automatically obtain a high quality map using our method even without requiring a collection, i.e. for two shapes only, by using collection composition.

Chapter 1

Introduction

A shape space contains variations of a given 3D model, for example, a sampling of an animation, or a character in different poses. Such spaces can arise when animators generate blend shapes for standard poses (smile, frown, A-pose, T-pose, etc.) or use rigged models for generating walking or running cycles. In many applications, a few such shape spaces are given (e.g. a walking cycle of a man and a woman), and it is required to transfer information between them. For example, transferring shape annotations such as pose labeling, or transferring point-wise data such as texture.

Often, it is possible to automatically obtain a high quality correspondence between variations of a single model, i.e., maps within the shape space, or shape collection, which we denote by *intra-maps*. On the other hand, correspondences between shapes in different collections, which we denote as *inter-maps*, are harder to compute, as these models will often have large non-isometric deformations.

Automatic computation of correspondences between non-rigid and non-isometric shapes is an active research problem. Most existing methods for computing a correspondence between non-isometric shapes require some additional semantic input, such as corresponding landmark points, and are therefore not fully automatic. We, on the other hand, leverage the cues of the *variations* within the collection as the semantic information, and use them to design a *completely automatic* method.

We draw our inspiration from a recent approach to this problem [SBC14], which similarly leverages shape variations as the semantic cues. Unfortunately, many aspects of that approach introduce technical difficulties which make the method not robust and impractical in many cases, derailing the hope for a completely automatic method. Specifically, a large sampling of shapes is required in both collections, as well as hand tuning of multiple dataset-dependent parameters. Practically, the existing approach yields low quality maps which are not on-par with state of the art correspondence methods.

While working in the same general setting, we propose very different design choices, leading to a considerably more robust system. Our method is simple to implement, completely automatic, and applicable to any mesh topology, yielding significantly better results. Additionally, our method extracts high quality pointwise maps between non-isometric shapes, generating maps

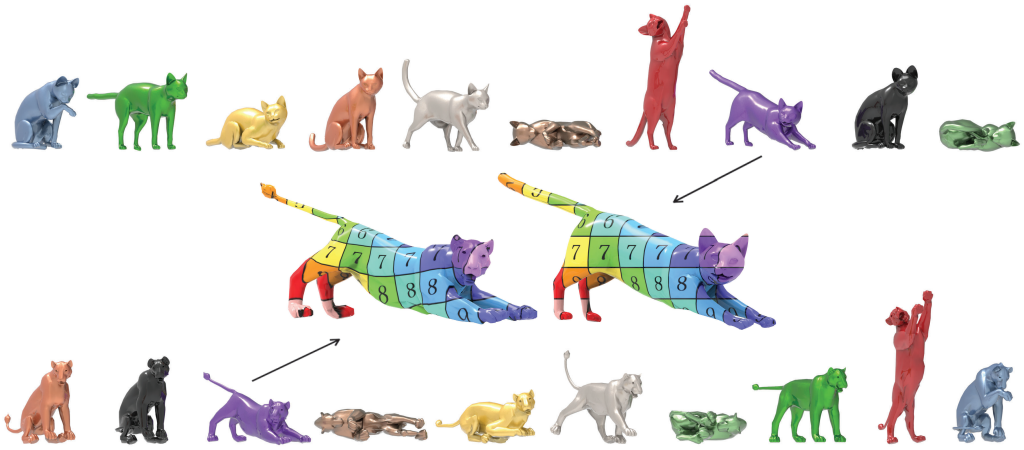


Figure 1.1: Given two shape collections with different shapes (cat and lion) in various poses and the intra-collection maps (top and bottom), our algorithm automatically finds a *matching* between the poses (the poses are color coded, such that matching poses have the same color). Using this matching, we further find an inter-collection correspondence between two automatically-chosen shapes, one from each collection (center). We can therefore *automatically* extract a high-quality non-isometric correspondence solely from the two collections, without any user input.

which surpass state-of-the-art automatic methods for non-isometric shape correspondence. Our approach is applicable to both benchmark data-sets, e.g. FAUST, as well as rigged models available for purchase from 3D modeling websites. Finally, it is important to note that the shape collections we can handle are not limited to the same character in different poses, but can rather be sets of shapes with any variations yielding similar enough shape spaces. For instance, a collection can include *different* people in the same pose, then matched to a collection of the same people in a different pose.

1.1 Related Work

Shape Collections. Collections of shapes are useful for a variety of applications. Some examples include deformation transfer, aiming to produce shapes with desired deformations that can be induced by given deformations in another collection. Some methods include learning models for 3D shape processing and shape reconstruction [GYQ+18, HRA+19]. Another example is *creating* a collection from one shape using modal analysis of the hessian of the mesh as described in [HWAG09]. Other aspects of shape collection analysis exploit the idea that the composition of maps along cycles should be identity maps, which led to the map synchronization problem, taking maps between pairs of objects and returning improved maps that are consistent along cycles [SLHH18]. This notion is further exploited in [NBCW+11, CRA+17] in order to obtain high quality intra-maps, that we require as an input, and which are easier to obtain than the inter-maps we compute. Other approaches use modular latent spaces, based on nonlinear embedding spaces, to find the correspondences within a collection [GSDG18].

In [ROA+13] some approaches to exploring shape collections are presented, including

browsing shapes by a user-defined region of interest, taking advantage of the localization property of the *shape difference operator*. Shape analogies are also used in [ROA+13] to match corresponding shapes from two collections. However, they use a brute-force search among all the *permutations* of the shapes that best aligns them with the other collection, limiting the applicability to very small collections. The shape difference operator has been further investigated and analyzed in terms of stability [HCO18] and optimal shape collection representation [HAGO19].

Our work generalizes the method proposed by [SBC14]. In that approach, given two shape collections, each representing a shape space, it is assumed that the intra-maps are given, and they are used to compute the shape variations, represented by *shape difference operators*. These variations are then considered as points on a high dimensional manifold, and a non-linear dimensionality reduction approach is used to generate a low-dimensional embedding of the shape spaces. The shape spaces are then aligned using affine point cloud alignment. Finally, an inter-map is computed using the alignment. This approach has several severe limitations. First, a large sampling of shapes is required to reliably represent the diffusion map, and hand tuning of parameters is required to align the shape spaces using an affine map. Furthermore, the existing approach fails to obtain high accuracy rates for aligning the shape spaces, and in addition the resulting inter-maps are of low quality and are not comparable to other correspondence methods.

We, on the other hand, have made critically different design choices, leading to an algorithm that is more robust and yields maps comparable to other automatic methods for shape correspondence. Specifically, we leverage the fact that in some cases the shape difference operators lie on a manifold which has a closed form expression for *geodesic distances*, therefore, we use these distances as a better representation of the shape space structure. Second, we use a *linear* dimensionality reduction approach, which is far less sensitive to the number of shapes in the collection. Furthermore, we use rigid alignment using a robust convex relaxation, which is parameter-free. Finally, we add a regularization to the shape correspondence formulation, and a post processing, which yield considerably better pointwise maps.

Automatic Correspondences. Computing shape correspondence using automatic algorithms, i.e. when no landmarks or user input are given, acts as a benchmark for the second part of our algorithm, aiming to obtain the correspondences between shapes in different collections. Blended intrinsic maps (BIM) [KLF11] is an automatic method to recover the point-to-point map between two given shapes. It often yields excellent results, however it is restricted to genus zero shapes. Bijective and continuous ICP (BCICP) [RPWO18] tackles this problem as well and is able to produce correspondences without landmarks. We compare our algorithm, which uses the shape differences within the collection as the additional information required for automatically extracting an inter-collection map, to these methods and demonstrate that we achieve a better performance than both.

Matching Problems and Procrustes Analysis. In order to solve the matching problem, i.e. finding the matching pairs from the two collections, one can make use of the distance matrix

between the shape differences of each collection. The distances represent the differences between the variations in the collection. One approach to solve the matching problem is to use the method presented in [KKBL15] in order to find the best permutation on the distance matrices' rows and columns such that they have the same structure. The retrieved permutation defines the pairing of shapes we wish to obtain. Unfortunately, this approach is not feasible for large data-sets where the distance matrices are large. An improved method has been proposed in [DML17] (see also references within for additional approaches to quadratic matching).

Instead of aligning the distance matrices, it is also possible to generate a low dimensional embedding of the point clouds, and use Procrustes matching (PM) for the registration. We chose to take this approach, first using linear dimensionality reduction to generate a point cloud representation of the shape collections, and then convex semidefinite programming (SDP) relaxation [MDK$^+$16] for the Procrustes matching of the point clouds. This approach has a few advantages. First, linear dimensionality reduction requires less parameters. Second, the SDP relaxation scheme is tight leading to close to optimal results, and finally, it is very efficient, enabling us to handle larger collection sizes.

Pointwise Map Extraction from Functional Maps. Functional maps allow us to transfer functions from one shape to another [OBCS$^+$12, OCB$^+$16]. When functional maps are represented using a reduced basis, such maps can provide only low frequency information about the correspondence. However, a high-quality pointwise map should in addition be able to transport high frequency data between the shapes. Once we retrieve the functional inter-map using our method, we would like to extract a high-quality point-to-point inter-map. The method described in [EBC17], proposes an efficient way to recover precise maps from functional maps. Another recent method tackling a similar problem is based on optimizing the harmonicity and reversibility of the forward and backward maps, known as the reversible harmonic maps (RHM) [ESBC19]. This approach, can handle diverse geometries, and can receive as an input functional maps or dense maps. Our method can take advantage of RHM as post-processing in order to refine the map obtained using [EBC17]. Both [EBC17] and [ESBC19] are publicly available.

1.2 Method Outline

Our algorithm follows the general structure suggested in [SBC14], yet with some critically different design choices. We first describe the general pipeline (see Figure 1.2), and then explain each component separately in Section 3.

Input: Two shape collections that represent two shape spaces of different models, not necessarily with the same number of shapes, and the intra-maps within each collection.

Output: (1) An injective function from the smaller collection to the larger one, which represents the *matching* pairs, and (2) a point-to-point *inter-map* for any shape in one collection to any shape in the other.

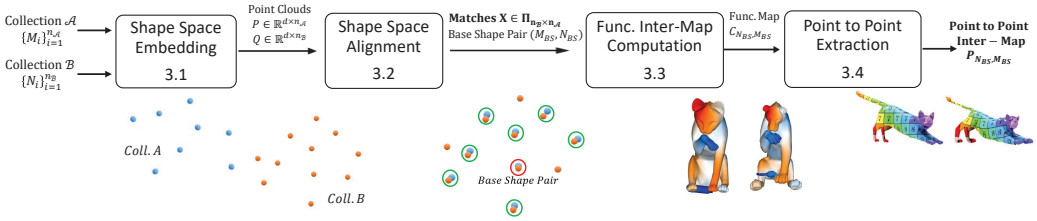


Figure 1.2: Block diagram of our algorithm, see the outline in Section 1.2.

Algorithm:

1. Use the input intra-maps to construct the shape differences between the input shapes in the same collection (Section 2.2).
2. Use the shape differences to construct a low-dimensional shape space embedding for both collections (Section 3.1).
3. Align the two shape spaces using Procrustes analysis to obtain the matching pairs and automatically determine an *optimal base shape* in each collection (Section 3.2).
4. Use the matches and the base shapes to compute a functional inter-map for the base shape pair (Section 3.3).
5. Recover a point-to-point inter-map from the functional map (Section 3.4).

1.3 Contributions

Our main contributions are:

- Matching corresponding pairs from two shape collections with a high rate of accuracy.
- Automatically obtaining a high-quality non-isometric inter-map between any shape in one collection to any shape in the other.

Chapter 2

Background

Our approach relies on recent techniques for shape correspondence and shape variability analysis which we briefly review here.

2.1 Functional Maps

Functional maps [OBCS$^+$12] have been widely used in many geometry processing applications, especially for shape correspondence. The main observation in this framework is that we can look at how *functions* on one shape are transformed to the other, rather than finding a point-to-point map between the shapes. The space of functions can often be well represented using a compact, yet informative, *functional basis*, allowing us to represent a map in this space as a *change of basis* linear operator, i.e., as a compact matrix.

Given two surfaces M and N, with a map $T : N \to M$ between them, a map between function spaces $F : L^2(M) \to L^2(N)$ is induced. Here, $L^2(\cdot)$ represents the set of square integrable real valued functions defined on the surface. F is called the functional map, mapping functions defined on M to functions defined on N, i.e. $g = F(f) = f \circ T$ where $f : M \to \mathbb{R}$ and $g : N \to \mathbb{R}$. F is a linear transformation between function spaces, and given reduced bases Ψ_M, Ψ_N of dimensions k_M, k_N, for M, N, respectively, is represented as a matrix $C \in \mathbb{R}^{k_N \times k_M}$.

2.2 Shape Difference Operators

Given two shapes and the map between them, the shape difference operator [ROA$^+$13] captures the variations between the shapes. It constitutes the main tool in our algorithm, allowing us to compare differences between shapes. This operator stores information about how a function on one of the shapes should be modified, such that the *inner product* of any two functions defined on the first surface equals to the inner product of the mapped functions on the other surface. In that way, it captures information about how and where one shape differs from the other. In addition, by modifying the inner product, we can quantify different types of differences, or distortions, between the shapes.

Given two shapes M and N, and a functional map $F : L^2(M) \to L^2(N)$, let $h_M : L^2(M) \times L^2(M) \to \mathbb{R}$ and $h_N : L^2(N) \times L^2(N) \to \mathbb{R}$ be inner products defined on M and N respectively, acting on two functions on the shape. Then there exists [ROA+13, Thm 2] a *unique* linear operator $D_{h_M,h_N} : L^2(M) \to L^2(M)$, denoted as the *shape difference operator*, satisfying $h_M(f, D_{h_M,h_N}(g)) = h_N(F(f), F(g))$ for any two functions $f, g : M \to \mathbb{R}$. Note that this operator depends only on the chosen inner products on M and N and the functional map F. Moreover, it is a linear self-map on the space of functions over M. Thus, the operators $D_{h_M,h_{N_1}}, D_{h_M,h_{N_2}}$ have the same domain and range $L^2(M)$, allowing us to compare the operators even if $N_1 \neq N_2$.

Two inner products that are of interest, are the *area-based* inner product $h^a(f, g) = \int_M f(x)g(x)d\mu(x)$ and the *conformal* inner product $h^c(f, g) = \int_M \langle \nabla f(x), \nabla g(x) \rangle_x d\mu(x)$. Intuitively, the first encodes variations in area due to the map, and the second encodes variations in angles.

Given a choice of reduced basis Ψ_M, Ψ_N, the shape difference operators are represented as matrices of dimensions $k_M \times k_M$. Often, the eigenvectors of the Laplace-Beltrami operator corresponding to the lowest eigenvalues are chosen for the basis Ψ, as this choice leads to a multi-scale basis, which can represent well smooth functions using only a small number of basis functions. In this case, the explicit expressions for the shape difference operators are[ROA+13, Eq. (4)]:

$$V_{M,N} \triangleq D_{h^a_M,h^a_N} = F^T F \tag{2.1}$$

$$R_{M,N} \triangleq D_{h^c_M,h^c_N} = (D^M)^{-1} F^T D^N F, \tag{2.2}$$

where F is the functional map represented in the bases Ψ_M, Ψ_N, and D^M is a diagonal matrix of the lowest non-zero k_M eigenvalues of the Laplace-Beltrami operator of shape M.

Chapter 3

Algorithm

We now give a detailed explanation of our algorithm and the motivation behind our design choices.

3.1 Shape Space Embedding

Given two shape collections $\mathcal{A} = \{M_i\}_1^{n_\mathcal{A}}$, $\mathcal{B} = \{N_i\}_1^{n_\mathcal{B}}$, we assume that each collection includes shapes sampled from a single shape space. We further assume, that the samplings are "compatible" in the sense that they include similar shape variations. Then, given a choice of two *base* shapes, one in each collection, $M_{BS} \in \mathcal{A}$, $N_{BS} \in \mathcal{B}$, we compute for each collection separately the shape difference operators with respect to the base shapes. These operators encode information about the variability of the collection, and the *distances* between them encode the structure of the shape space.

We first select a *random* shape in each collection, serving as the base shape for the following part. Later we show that the base shape can indeed be chosen randomly without affecting the performance of the algorithm.

3.1.1 Shape difference operators computation

Once we have chosen a random base shape for each collection, we compute the shape difference operators for all the shapes in the collection with respect to this base shape. As shown in the previous section, shape difference operators can be computed using either the area-based inner product (Eq. (2.1)) or the conformal inner product (Eq. (2.2)). We later show how the choice of inner product depends on the nature of the collection. However, for typical data-sets, we choose the conformal inner product since the variations within the collection are mostly non-conformal.

Given the intra-map between shapes within the collection, we compute the functional map between the base shape and any other shape in the set using k_1 eigenvectors of the Laplace-Beltrami operator of the base shape, and k_2 eigenvectors for the other shape. A collection of n shapes yields a set of n shape difference matrices of size $k_1 \times k_1$. Each of them represents the

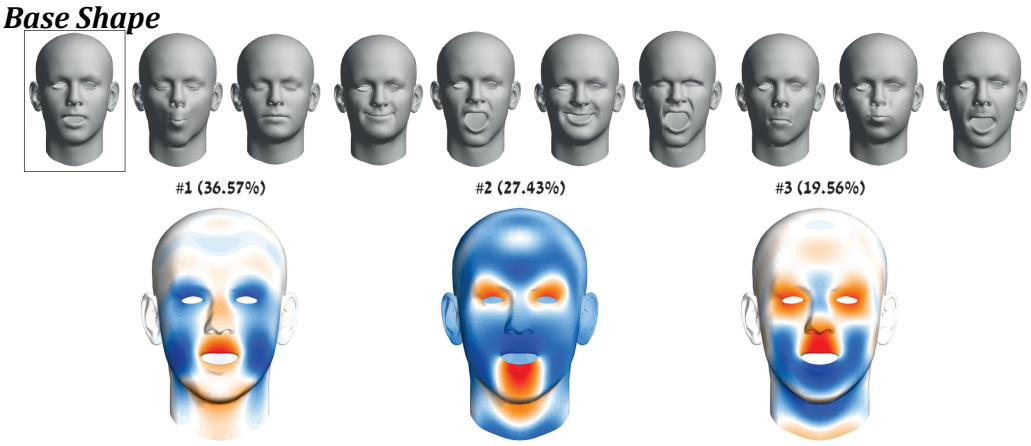


Figure 3.1: A collection of faces with varying expressions (top) with its shape differences' principal components visualization (bottom). Since they are computed with an arbitrary sign, both dark blue and dark orange indicate areas with high variation. The explained variance of each component appears above it. The principal components encode the main variations within the collection, mainly around the mouth and eyes area.

modification that needs to be done on functions on the base shape such that inner products on the base shape equal to inner products on the other shape.

Figure 3.1 shows the principal components of the shape difference operators for a given collection using the area-based inner product. The principal components are visualized by testing how they act on a constant function, thus visualizing the main variations between the shapes and their location. The percentage of the total variance explained by each principal component is shown as well, demonstrating the significance of each component in the collection.

Now that we have the set of shape difference operators, our goal is to embed them in a low dimensional space such that the information of the distances between the operators is best preserved.

3.1.2 Distance matrix construction

To construct a distance matrix for a given collection, we treat separately the area-based and conformal shape differences, as they have a different structure as operators.

Area based. The area-based shape difference operator, given by Eq. (2.1), is a *symmetric positive-definite* (SPD) matrix, as long as the functional map F is an invertible matrix. We therefore use the manifold of SPD matrices to compute the *geodesic distances* between the shape difference operators. The manifold of SPD matrices has a unique shortest geodesic curve between any two points, and its length has a closed form expression. Specifically, given two

SPD matrices, V_1, V_2, their geodesic distance on the SPD manifold is [Bha09, Eq. (6.14)]:

$$d^2_{SPD}(V_1, V_2) = \sum_{i=1}^{n} \log^2 \left(\lambda_i (V_1 V_2^{-1}) \right), \tag{3.1}$$

where $\lambda_i(V)$ is the i^{th} eigenvalue of the matrix V.

Conformal. The conformal shape difference matrices, given by Eq. (2.2), are not SPD, and we therefore use the Euclidean distances between the vectorized matrices.

Figure 3.2 shows the distance matrices obtained for the Sumner data-set (Figure 1.1), where the shapes are ordered correspondingly for better visualization. The distance matrix for the set of cats (collection $\mathcal{A}$) has a similar structure to the distance matrix computed for the set of lions (collection $\mathcal{B}$). In both sets we use the shape differences operators derived for the conformal inner product, since the variations within each collection are mainly non-conformal. Hence, in this case, the conformal shape differences capture more reliably the information about the variations compared to the area-based shape differences.

3.1.3 Dimensionality reduction

To solve the matching problem, we first apply multidimensional scaling (MDS) [Mea92] on the distance matrix obtained in order to embed the operators in a d-dimensional space, the *shape space embedding*. The dimension d is determined such that the energy accumulated in this d-dimensional space is at least β of the total energy, for both shape collections. The energy of each dimension is defined as the variance explained by this dimension using PCA. For instance, choosing $\beta = 0.95$ means that we use at least 95% of the total energy for both collections. In this case, we lose only up to 5% of the information regarding the variations, but store it in a space of dimension in the range of 3-8 for typical data-sets.

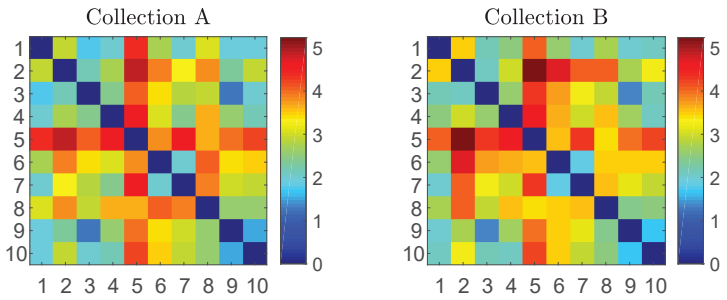


Figure 3.2: Distance matrices of the collections shown in Figure 1.1, computed for the conformal shape differences. For better visualization, in both collections the shapes are ordered correspondingly. Note that the collection of cats (collection $\mathcal{A}$) and the collection of lions (collection $\mathcal{B}$) have similar distance matrices, which we use for matching the collections.

3.2 Shape Space Alignment

After constructing the shape space embedding of each collection, our goal is to *align* them such that we can match shapes from the two collections. We treat the two shape space embeddings as two point clouds, and assume that the distance matrices of the two collections are similar, so that we can use *rigid alignment*. In this setting, the alignment problem is known as the *Procrustes matching problem*, formulated as follows.

We are given two point clouds P and Q of dimension d and n_1, n_2 points respectively, $P \in \mathbb{R}^{d \times n_1}, Q \in \mathbb{R}^{d \times n_2}$, where we assume, without loss of generality, that $n_1 \leq n_2$. Our goal is to find a linear isometry (an orthogonal transformation) $R \in \mathcal{O}(d)$ and a permutation $X \in \Pi_{n_2 \times n_1}$, minimizing the distance between the point clouds:

$$\begin{aligned} & d(P,Q) = \min_{X,R} ||RP - QX||_F^2 \\ & \text{s.t.} \\ & X \in \Pi_{n_2 \times n_1} \ , \ R \in \mathcal{O}(d). \end{aligned} \tag{3.2}$$

Since $n_1 \leq n_2$, this formulation matches every point in P to exactly one point in Q, where some points in Q can remain unmatched.

This is in general a difficult, non-convex problem, however, recently, a very effective convex relaxation to the problem has been proposed [MDK+16], which we leverage to find the alignment.

Let X^*, R^* be the optimal solutions of the optimization problem (3.2), then the alignment error $d(P^*, Q^*)$ provides a quantitative measure of the success of the alignment process. We define the *normalized alignment error*:

$$\hat{e} = \frac{||R^*P - QX^*||_F}{||P||_F + ||Q||_F}, \tag{3.3}$$

which is invariant to the dimensions of the point clouds. We will later use this error to evaluate the matching accuracy of our results.

Optimal Base Shape Selection. The point clouds $\tilde{P} = R^*P \in \mathbb{R}^{d \times n_1}$ and $\tilde{Q} = QX^* \in \mathbb{R}^{d \times n_1}$ include only pairs of matching shapes *after* the alignment. The closest pair of points in $\tilde{P}, \tilde{Q}$ are the two shapes which are most likely to have been matched correctly. Therefore, we select these as the new base shapes M_{BS}, N_{BS}, and the shape difference operators are recomputed with respect to them, yielding a better input for the next part of the algorithm.

3.3 Functional Inter-Map Computation

The next goal of the algorithm is to compute the functional map between the newly chosen base shapes of the two collections, thus retrieving the inter-map correspondence.

Shape analogies. We use the shape analogies constraint presented in [SBC14], taking advantage of the similarity between the shape difference matrices of matching pairs. More precisely, let $\{(M_i, N_i) \,|\, M_i \in \mathcal{A}, N_i \in \mathcal{B}\,, i = 1, \ldots, n\}$, be the $n = \min\{n_\mathcal{A}, n_\mathcal{B}\}$ matched pairs extracted by the shape space alignment. Recall that (M_{BS}, N_{BS}) are the pair of base shapes of the two collections, defined as the closest matching pair. Our goal is to find a matrix C, the functional map between the base shapes.

Our main assumption is that the shape difference operators of matching shapes in both collections, i.e., V_{M_{BS},M_i} and V_{N_{BS},N_i} act similarly on functions. Hence, first applying a map between the collections, and then applying the shape difference operator, should yield a similar function to first applying the shape difference and then mapping the function to the other collection.

Regularization. Differently from [SBC14], we observe that without a regularization term the optimization results are unstable, leading to poor functional maps. This is especially evident for areas on the shapes with none or few variations, where the shape differences do not hold any information. To handle this problem, we adopt the regularization term that forces the functional map to commute with the Laplace-Beltrami operator [OBCS$^+$12].

Optimization problem. Our optimization problem is therefore:

$$\min_{C \in \mathbb{R}^{k_\mathcal{B} \times k_\mathcal{A}}} \sum_{i=1}^{n} \left(\|C V^{\mathcal{A}}_{BS,i} - V^{\mathcal{B}}_{BS,i} C\|^2_F + \|C R^{\mathcal{A}}_{BS,i} - R^{\mathcal{B}}_{BS,i} C\|^2_F \right) + \alpha \|C \Delta^{\mathcal{A}}_{BS} - \Delta^{\mathcal{B}}_{BS} C\|^2_F, \tag{3.4}$$

where $k_\mathcal{A} = k_\mathcal{B} = k_1$, n is the number of matching pairs, $V^{\mathcal{A}}_{BS,i}$ is the area based shape difference from the base shape in $\mathcal{A}$ to $M_i \in \mathcal{A}$, $\Delta^{\mathcal{A}}_M$ is a diagonal matrix with the eigenvalues of the Laplace-Beltrami operator of shape $M \in \mathcal{A}$, and α is a parameter controlling the regularizer.

To solve this optimization problem we observe that the objective is linear in the elements of C, resulting in a homogeneous linear system that is solved using SVD.

As proposed in [OBCS$^+$12, SBC14], we apply iterative refinement as post-processing to the minimizer of Eq. (3.4) in order to refine the solution so that it better represents a point-to-point map. The refinement process is effectively ICP in eigenspace, where the solution of the optimization problem is used as initialization.

Finally, we compute the functional map, F_{N_j,M_i} between any two shapes in the two collections, $M_i \in \mathcal{A}$ and $N_j \in \mathcal{B}$ using the following composition:

$$F_{N_j,M_i} = F^{\mathcal{B}}_{j,BS} C_{N_{BS},M_{BS}} F^{\mathcal{A}}_{BS,i}, \tag{3.5}$$

where the functional map $C_{N_{BS},M_{BS}} : L^2(M_{BS}) \to L^2(N_{BS})$ is the computed inter-map, i.e. the minimizer of Eq. (3.4), and the functional maps $F^{\mathcal{A}}_{BS,i} : L^2(M_i) \to L^2(M_{BS})$ and $F^{\mathcal{B}}_{j,BS} : L^2(N_{BS}) \to L^2(N_j)$ are the input intra-maps within the collections.

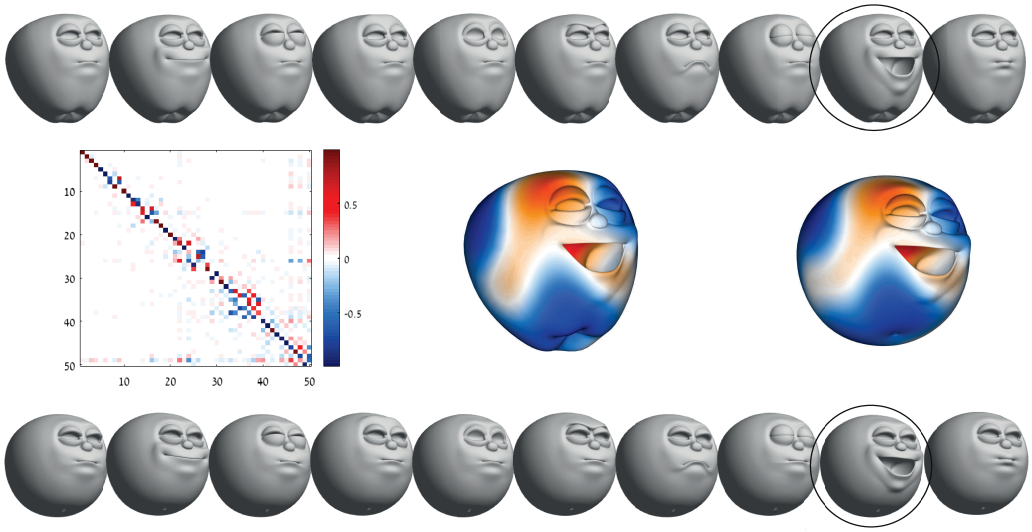


Figure 3.3: A functional map computed using our algorithm and its visualization using function transfer. Note that the map is of high quality in the areas where the shape differences are informative, i.e. areas with more variations (the ”face” of the apple and the orange). On the other hand, the functional map is distorted in areas where there is no information in the shape differences (the side of the apple and orange) since there are no variations within the set. When recovering the point-to-point map these distortions are fixed (see Figure 5.5).

In Figure 3.3 we visualize the computed functional map and demonstrate it with a specific function transferred from the apple (center) to the orange (right) from a data-set of rigged shapes (top and bottom). Note that a high quality map can be derived for the areas where the shape differences hold the information about the shapes, i.e. areas with more variations such as the ”face” of the apple and the orange. However, the map might be distorted in areas where there is no information in the shape differences, such as the side of the apple and orange, since there are no variations within the set in these regions. Our last step is to recover the point-to-point map, which additionally alleviates theses issues.

3.4 Recovering a Point-to-Point Inter-Map

The final step of our algorithm is to produce a point-to-point map using the functional map we computed. As we aim for a high quality map that can be used to transport textures, we use a recent map reconstruction approach [EBC17] to obtain a vertex-to-point map.

Some data-sets require post-processing of the point-to-point map using Reversible Harmonic Maps (RHM) [ESBC19]. This post-processing scheme is especially effective in areas on the shape where there are few or no variations within the collection, such that the shape difference operators contain no information for these areas. As a result, in these regions, the shape analogy constraints are not effective, resulting, locally, in a poor map. Post-processing using a map smoothing algorithm allows us to smoothly interpolate the map in such regions. To make a fair comparison, we also apply post-processing using RHM to all the methods that we compare to.

Chapter 4

Analysis

In this section we evaluate the different design choices that we made, and investigate the properties of our approach. We show that the initial choice of base shapes can indeed be random, without affecting the algorithm's performance. We also discuss the considerations of choosing shape difference operators derived from area-based inner product and conformal inner product. We explain and demonstrate the advantage of using the Riemannian distance for the area-based shape differences rather than the Euclidean approximation. Finally we show our algorithm's ability to handle collections of different size.

4.1 Resilience to Base Shape Choice

We show that our method is resilient to the choice of base shape, thus we can select it randomly, without requiring additional input.

In [SBC14] the result strongly depended on the choice of base shape in each collection, i.e. a bad base shape choice led to poor matching and poor functional maps. Our algorithm, on the other hand, is able to match corresponding pairs successfully, even when the choice of base shape is random. With a random choice of base shape in each collection, the chosen shapes (in most cases) do not represent a matching pair. Yet, the first part of our algorithm, aiming to match pairs of the two collections, is barely affected. The second part of the algorithm, producing the functional map, is affected by the choice of base shape, however it benefits from the result of

Table 4.1: Resilience to Base Shape Choice: corresponding base shape pair. Results are averaged over all corresponding base shape choices.

Result\Data-Set	Blend Shapes	Sumner	Rigged	FAUST
Average matching accuracy	99.88%	100%	100%	87.9%
Average normalized alignment error	0.1542	0.1950	0.0566	0.3383
Rate of corresponding base shapes after alignment	40/40	10/10	10/10	86/100

Table 4.2: Resilience to Base Shape Choice: **non**-corresponding base shape pair. Results are averaged over **non**-corresponding base shape choices.

Result\Data-Set	Blend Shapes	Sumner	Rigged	FAUST
Average matching accuracy	99.25%	100%	100%	85.5%
Average normalized alignment error	0.2012	0.3119	0.1481	0.3555
Rate of corresponding base shapes after alignment	39/40	10/10	10/10	85/100

the first part by using the computed matched pairs.

Shape space alignment. To demonstrate the invariance of our approach to the choice of base shape, we test all possible base shape choices on a few datasets, and compare with the ground truth. We test both the case that the base shapes correspond, i.e., represent the same pose (Table 4.1), and the case where the base shapes do not correspond (Table 4.2). For each possible choice of base shapes, we run the algorithm and measure a few performance indicators. *Matching accuracy* represents the percent of shapes that were matched correctly to the corresponding shape in the second collection, and it is averaged over all choices of base shape pair. The *normalized alignment error*, defined in Eq. (3.3), indicates how well the two point clouds are aligned after applying the rigid transformation. It is also averaged over all experiments. Finally, as discussed previously, after alignment we choose the pair with the smallest distance as the new base shape pair. The *rate of corresponding base shapes after alignment* indicates in how many cases the chosen base shape pair after alignment indeed represents a corresponding pair. For the FAUST data set we averaged the result over 10 subset pairs, each subset containing 10 shapes referring to the same person.

Note that for tested data-sets, the choice of base shapes, and whether or not the base shapes correspond, does not significantly affect the average matching accuracy nor the rate of corresponding base shapes chosen after alignment. The reason is that the principal components derived from PCA do not significantly depend on the base shape choice, causing only small variations in the shape space, that our alignment procedure can handle. Although the average normalized alignment error is larger for non-corresponding base shapes, the matching accuracy is hardly affected, meaning that we can still align successfully the two point clouds. These results also demonstrate that in most cases the new chosen base shape pair is a corresponding pair.

Inter-map computation. In the second part of the algorithm, it is more important to have corresponding base shapes. Figure 4.1 shows that the functional map computation has preferable results for corresponding base shapes, resulting in a less noisy functional map matrix, which is more similar to the ground truth. This motivates our choice to leverage the alignment results from the first part of the algorithm, and choose a new pair of base shapes which are more likely

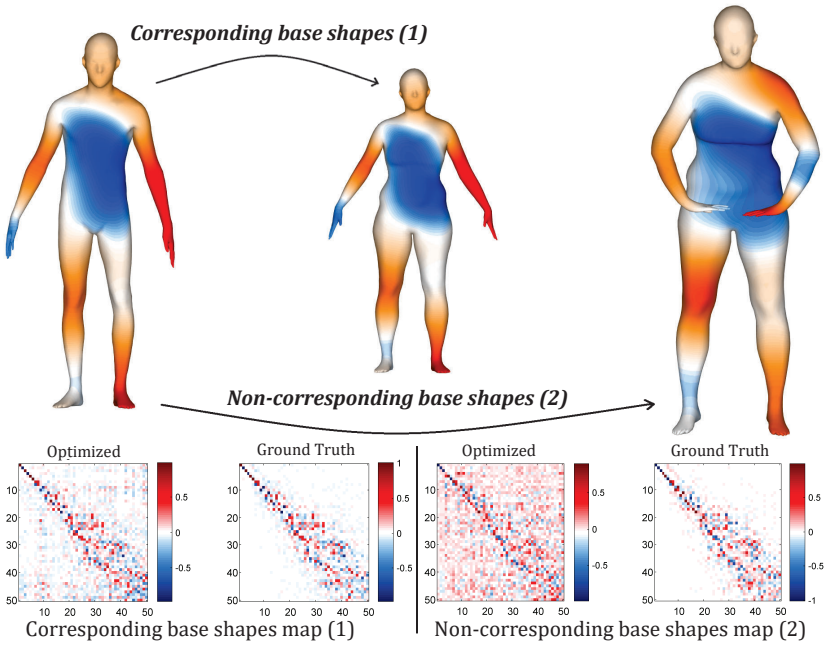


Figure 4.1: Functional maps and their visualization for corresponding and non-corresponding base shapes. Corresponding base shapes lead to a higher quality functional map (e.g. arms area).

to match for computation of the functional inter-map.

4.2 Area-based vs. Conformal Shape Differences

The choice of using area-based or conformal shape difference operators depends on the nature of the data-set. For collections in which the variations between the shapes are area preserving but not conformal (e.g. the same person in different poses, Figure 4.2), it is beneficial to use the conformal shape difference operator, as it better captures the variations in such a collection. Similarly, for collections with conformal variations but not area preserving (e.g. different people in the same pose, Figure 4.3) we would rather use the area-based shape differences.

Figure 4.4 demonstrates how the choice of shape difference type affects the performance of the algorithm, in the case of data-sets in which each collection has non-conformal variations and in the case of non-area-preserving variations. We show the *matching accuracy* (top) and the *normalized alignment error* (bottom left), defined in Eq. (3.3), as a function of the dimension of the shape space. Note that for non-conformal variations, using the conformal shape difference operators yields better results, using a lower shape space dimension, and vice versa for the non-area-preserving case.

Figure 4.4 (bottom, right) shows the *cumulative energy*, defined as the sum of variances explained up to a certain dimension by using PCA. We demonstrate that for a collection with mostly area variations (same pose), the area based shape difference explains the data using less principal components than the conformal shape difference. For the data-set with the non-conformal variations, the number of principal components required to explain the data is similar

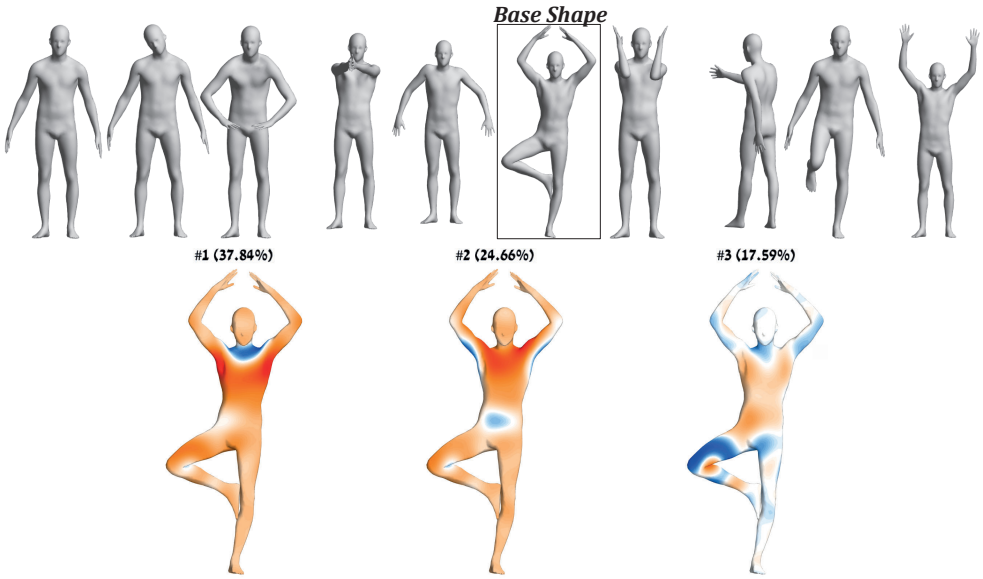


Figure 4.2: An example of a shape collection with non-conformal variations (top) and their first three conformal shape differences' principal components (bottom). The explained variance of each component is shown above it. Note that the variations are located around the joints area, where motion occurs, mainly in the shoulders area (first and second principal components) and the right knee area (third principal component).

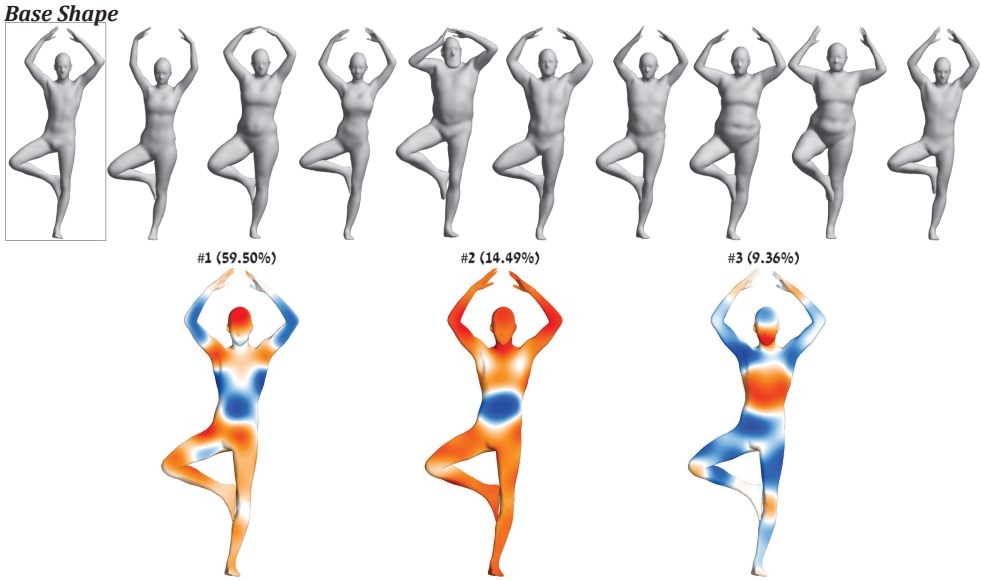


Figure 4.3: An example of a shape collection with non-area-preserving variations (top) and their first three area based shape differences' principal components (bottom). The explained variance of each component is shown above it. In this collection, the variations are located around the areas where people differ, mostly hips, torso and abdominal area. For example, the second principal component corresponds to greater abdominal area at the expense of thicker arms (or vice versa).

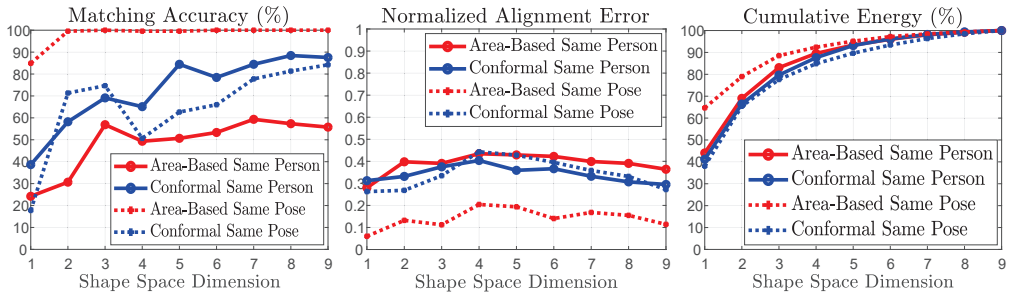


Figure 4.4: Comparison of our results for two collection types, with shape difference operators based on two different inner products. Note that for a collection with the same person but in different poses (Figure 4.2), the conformal shape differences show better performance. On the other hand, for a collection with the same pose of different people (Figure 4.3) the area-based shape differences perform better. See the text for details.

for conformal and area-based shape differences.

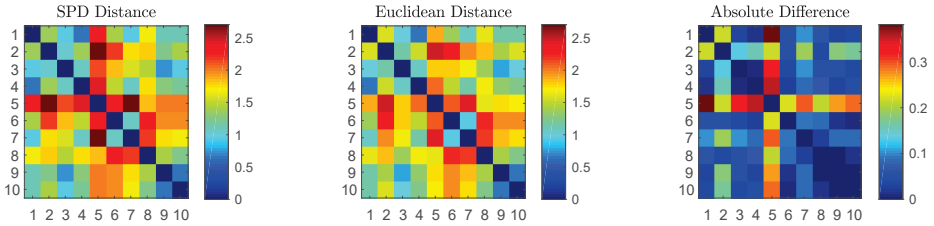
Note that even if the distances between the shape differences are similar for the two collections, the low-dimensional embedding can be unstable, in the sense that the principal components are only defined up to sign. Furthermore, even if the shape differences are *exactly* the same, if there are two principal components that have a similar explained variance, their order can be arbitrary. However, as we seek a rigid transformation that matches the two sets (including orientation-reversing transformations), our method is robust to these instabilities. If, however, we take a considerably smaller number of dimensions, some principal components that appear in one collection might not be present in the other. This leads to the non-monotonic behavior of the performance graphs in Figure 4.4 with respect to the embedding dimension.

4.3 SPD Riemannian Distance vs. Euclidean Distance

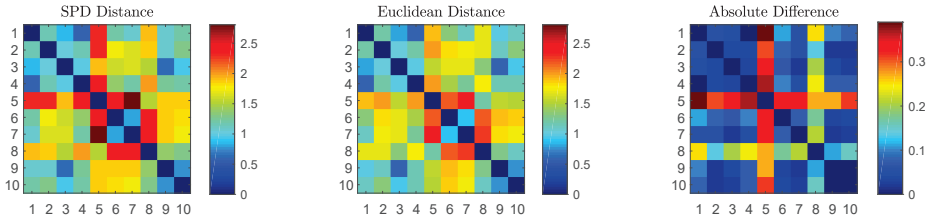
Now we explain why it is beneficiary to take advantage of the fact that the area-based shape difference operators are Symmetric Positive-Definite (SPD) matrices, by using the expression for the Riemannian distance on this manifold (Eq. (3.1)).

Figure 4.5 shows the distance matrices computed using SPD distance vs. Euclidean distance (using the same color-bar) for various collections. The difference matrix (absolute value) between the two distance matrices is shown as well. It is clear that we get different results for the two distance types. Note that the largest difference between the two matrices is around 15% of the distance. Therefore, we conclude that Euclidean distance is an *approximation* since it does not take into account the manifold curvature. Thus, we would rather use the SPD distance in order to avoid approximations. It is important to mention that it is easy to compute and does not increase our algorithm's timing. Also note that we get different distance matrices for the cat and lion from Sumner dataset (Figure 4.5a, 4.5b) compared to those in the main paper (Figure 4) since here we use the area-based shape differences rather than the conformal.

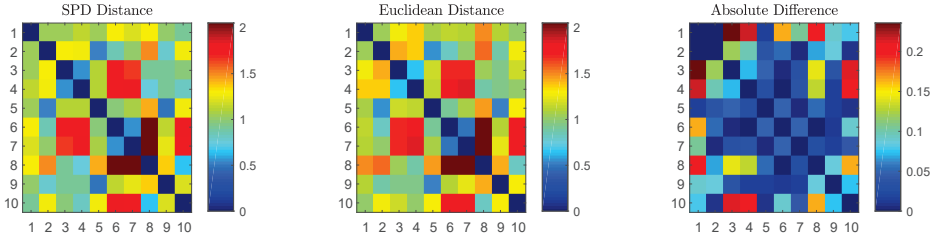
Due to the reason that a collection of different people in the same pose (Figure 4.3) has mainly non area preserving variations, we could test the effect of distance type choice on



(a) Distance matrices for the **cat** in Sumner dataset.



(b) Distance matrices for the **lion** in Sumner dataset.



(c) Distance matrices for **various people in the same pose** from FAUST dataset.

Figure 4.5: Distance matrices computed using SPD Riemannian distance, using Euclidean distance and their difference (absolute value).

the algorithm's performance. Figure 4.6 demonstrates the *matching accuracy* and *normalized alignment error* (as in Figure 4.4) for matching collections of various people in the same pose. The result was averaged over all possible collection combinations (45 in total) and plotted vs. the shape space dimension. Using SPD distance leads to higher matching accuracy and lower alignment error. Thus, we can indeed see that using SPD distance gives slightly, however consistent, better performance than the Euclidean distance approximation.

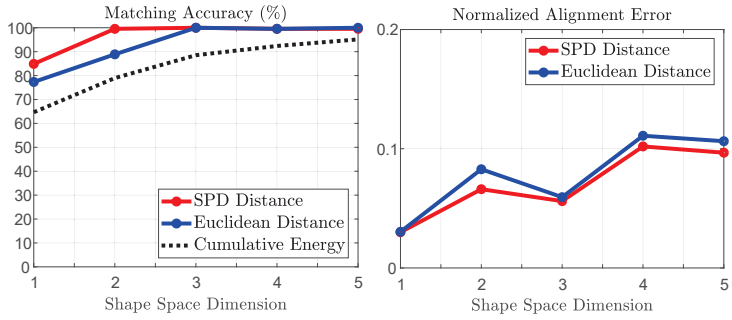


Figure 4.6: Matching accuracy and normalized alignment error vs. shape space dimension for SPD and Euclidean distance for various people in the same pose from FAUST dataset.

4.4 Collections of different size

Now we treat the case of collections of different size, where some of the shapes in the bigger collection do not have a match in the smaller collection. Our algorithm naturally handles this case, since we find a permutation matrix that selects points from the larger collection's point cloud such that they align better with the rotated smaller point cloud (see Eq. (3.2)). We analyze how the excess shapes in the bigger collection affect our results. We split the answer into two cases.

4.4.1 Small collections

For small collections, e.g. containing five shapes, adding an excess shape can significantly affect the shape space of the resulting extended collection. This happens since the variations of the added shape are not always explained by a linear combination of the existing principal components. Hence, the newly computed principal components are significantly different, leading to very different shape space embeddings for the two collections, yielding poorer matching accuracy after aligning the two point clouds. Figure 4.7 (left) shows the matching accuracy for various small data-sets. Starting from collections of five shapes each, we added one shape at a time to one of the collections and tested the matching accuracy averaged over all possible choices of base shapes for the two collections. Finally, the bigger collection had twice as many shapes as the original. It can indeed be seen that the matching accuracy is significantly affected and decreases quickly. The notation A, B and C for the Blend Shapes data-set refers to three different subsets from this data-set. Note that each collection in the FAUST data-set contains up to ten shapes, thus it is treated as a small collection.

4.4.2 Large collections

In the case of large collections, containing twenty shapes for instance, it is likely that the variations of the added shapes can be described using a linear combination of the existing principal components. In this case the shape space is not significantly affected, and adding shapes is equivalent to adding points to the point cloud without moving the original points. Figure 4.7 (right) demonstrates this, as the matching accuracy is less affected in this case. Here we start with original collections of size twenty and add up to twenty more shapes to one of the collections resulting in a collection which has twice as many shapes as the original. Despite this, the matching accuracy decreases only to $65 - 85\%$, depending on the data-set. Also note that due to the linear dimensionality reduction, we do not have a strict limit on the largest collection size. However, the point cloud alignment timing will be the performance bottle neck for very large data-sets.

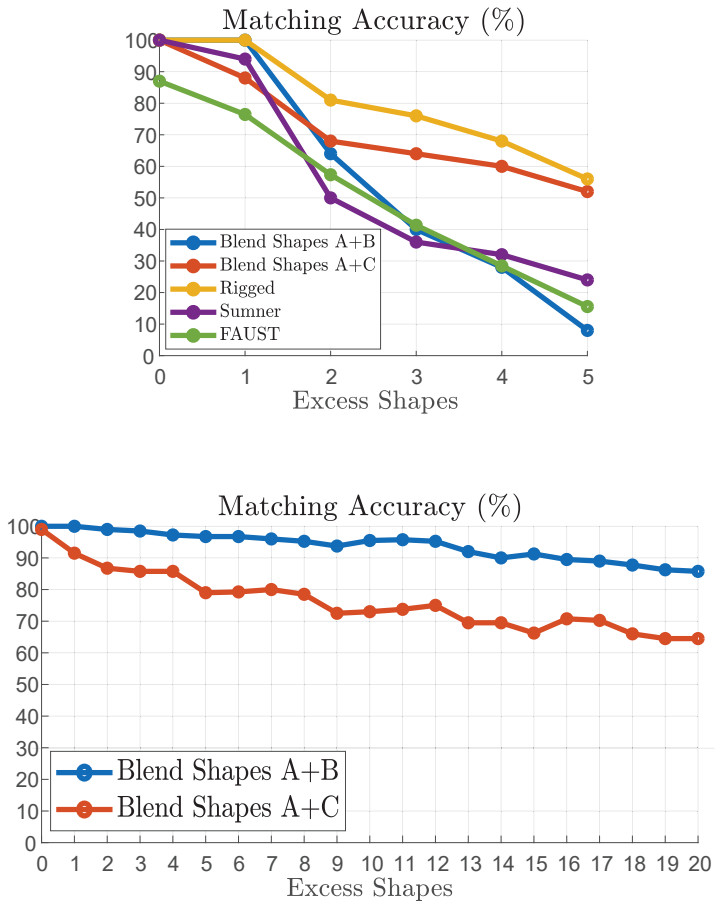


Figure 4.7: Matching accuracy for collections of different size. Note the considerable effect on the matching accuracy for small collections (left) in contrast to large collections (right).

Chapter 5

Experimental Results

In this section we show some results of our algorithm, compare it to the previous shape collection matching method, and to other automatic correspondence methods. We show that our method automatically computes high-quality maps, which compare favorably to existing automatic methods, given even two small collections or short animations, and even for less-than-perfect matching results.

5.1 Implementation details

Parameters. Our algorithm does not require data-dependent parameter tuning and we fixed the parameters to $k_1 = 50$ eigenfunctions on the base shape and $k_2 = 3k_1 = 150$ eigenfunctions on the other shape when computing the shape difference operators resulting in 50×50 matrices. The matching part has been done with conformal shape differences, unless stated otherwise. We set $\beta = 0.95$ to determine the shape space dimension and $\alpha = 0.1$ for the Laplacian commutativity in the energy we minimize (after normalizing both terms to have the same Frobenius norm). We run the ICP refinement of the functional map until the biggest change in the functional map matrix elements is smaller than 10^{-10}. We set the number of iterations in the RHM post-processing to 200, as recommended by the authors.

Timing. Our timings are comparable to other automatic methods for computing correspondences. Implemented with MATLAB, on a desktop machine with an Intel Core i7 @3.4GHz processor, the first part of the algorithm matches the pairs of the two collections of 10 shapes each, with shapes of $5K - 8K$ vertices in 45 seconds. The second part of the algorithm extracts the point-to-point inter-map in 130 seconds. The RHM post-processing takes 220 seconds on our machine. For most data-sets, it is possible to use smaller functional map matrices, e.g. $k_1 = 30$ and $k_2 = 3k_1 = 90$ without significantly decreasing the performance and obtaining both matches and an inter-map in approximately 75 seconds.

Limitations. We assume that the collections will lead to a similar point cloud structure of the shape space embedding, i.e., the variations within each collection should be similar. We also

require non-symmetric shape space embeddings to allow unambiguous rigid alignment. For shape spaces that require a very high dimensional embedding (12 and greater) alignment using PM-SDP [MDK$^+$16] can be slow.

5.2 Comparison with [SBC14]

The goal of this section is to compare our method and [SBC14] which we drew our inspiration from. Here we give comparisons of the matching accuracy and functional maps quality obtained using our algorithm compared to that of [SBC14]. In Appendix A we give additional comparison with [SBC14] to emphasize the differences in the methods and show the pointwise maps obtained by them.

5.2.1 Comparison: Matching Corresponding Shapes

In the previous Section, Table 4.1 and Table 4.2 showed our algorithm's matching accuracy on four different data-sets, regardless of the base shape choice and demonstrated the robustness of our algorithm on several data-sets. For the FAUST data-set, it was enough to mistake only two pairs to decrease the matching accuracy to 80%, since every collection contains ten shapes.

Now we compare our algorithm to the only method known to us having the same goal of matching corresponding pairs from two collections. We test our algorithm and the method in [SBC14] on the Blend Shapes data-set with varying collection size. The results appear in Table 5.1 and show that in contrast to [SBC14], our method does *not* require large data-sets for successful matching and we get perfect matching for any collection size. Our results were averaged over 100 random initial base shapes choices since our algorithm selects a base shape randomly.

Table 5.1: Blend Shapes data-set: Matching accuracy comparison to [SBC14]

Collection Size\Method	SBC14	Ours
20 shapes	35%	99.47%
25 shapes	36%	99.78%
35 shapes	63%	99.54%
40 shapes	90%	99.41%

5.2.2 Functional Map Comparison

Figure 5.1 shows the quality of our functional map compared to that of [SBC14], for the case of 40 shapes. Note that our map is better, even though the matching accuracy of [SBC14] is 90%. It is clear that using the regularization term of the Laplacian commutativity in Eq. (3.4) significantly improves the results, allowing us to obtain a functional map very similar to the ground truth.

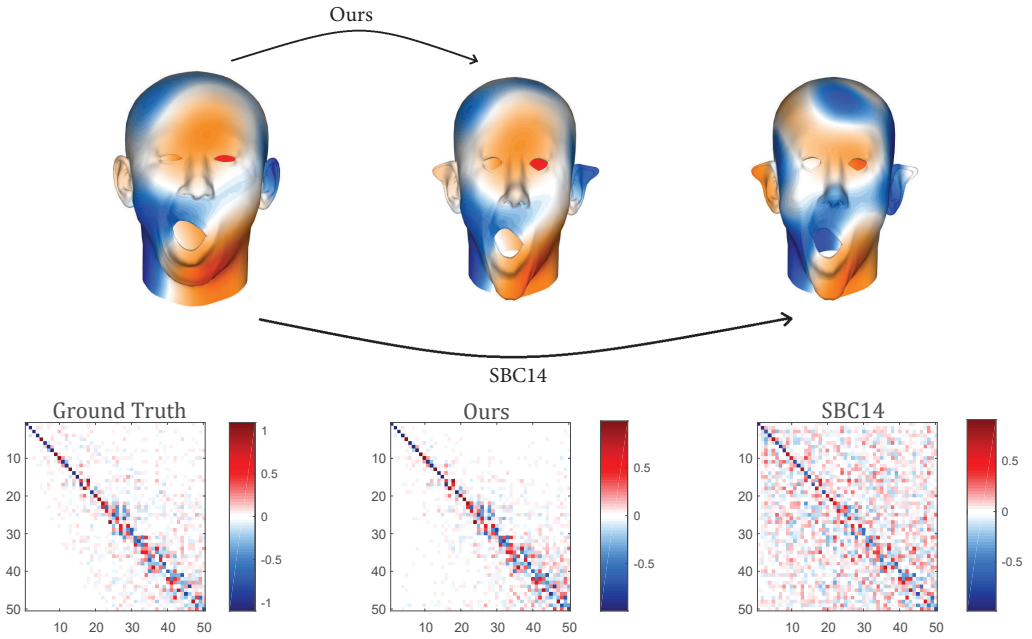


Figure 5.1: Comparing the functional maps obtained using our algorithm and [SBC14] using function transport (top), and comparison to the ground truth matrix (bottom). Our algorithm obtains a high-quality functional map, very similar to the ground truth.

5.3 Comparison: Point-to-Point Inter-Map Computation

Now we demonstrate and evaluate the computation of the point-to-point inter-map between the two base shapes, one from each collection. For the qualitative comparison, we show the texture on the target mesh we map to and the ground truth, if it exists, of the source mesh we map from. We compare our method to other automatic methods: BIM [KLF11] and BCICP [RPWO18]. For all methods, we show the results with and without post-processing using RHM for map refinement. For the quantitative evaluation we measure the map smoothness through its conformal and area distortions, and its semantic accuracy, using the distance to the ground truth correspondence, when given.

5.3.1 Quality Metrics

Conformal distortion. We use the definition given in [HG00, Eq. (3)] for the conformal distortion of a single triangle $f_{M_{BS}} \in \mathcal{F}_{M_{BS}} : \kappa(f) = \frac{\sigma_{M_{BS}}}{\sigma_{N_{BS}}} + \frac{\sigma_{N_{BS}}}{\sigma_{M_{BS}}}$ where $\sigma_{M_{BS}} \geq \sigma_{N_{BS}}$ are the singular values of the linear transformation which maps $f_{M_{BS}}$ from M_{BS} to N_{BS}. We subtract 2 such that the minimal conformal distortion is zero and visualize the result as a cumulative graph showing the percentage of triangles with less than a certain distortion value.

Area distortion. The area distortion is computed by measuring the distortion in the area of each triangle caused by the map from M_{BS} to N_{BS}. For each triangle we compute $\left|\log\left(\frac{A(P_{N_{BS},M_{BS}} f_{M_{BS}})}{A(f_{N_{BS}})}\right)\right|$ where $A(f_{N_{BS}})$ denotes the area of a triangle $f_{N_{BS}} \in \mathcal{F}_{N_{BS}}$ and

$A(P_{N_{BS},M_{BS}} f_{M_{BS}})$ the area of that triangle mapped by $P_{N_{BS},M_{BS}}$, the extracted point-to-point inter-map. We visualize the result as a cumulative graph showing the percentage of triangles with less than a certain area distortion value.

Distance from ground truth. When a ground truth map is given, we measure the distance from the ground truth using the protocol suggested by [KLF11, Section 8.2]. For every mapped vertex, we measure its geodesic distance from the ground truth location, relative to the square root of the total area of N_{BS}, and visualize the percent of vertices whose distortion is less than a given value.

5.3.2 Setup

Choice of pairs. We always compute the map between the automatically computed base shapes after alignment. Since the initial choice of base shapes is random, the pair with the smallest distance after alignment may vary, resulting in a different pair of shapes (one from each collection) that we map between. Therefore, We average the results obtained by running the algorithm n times, where n equals the (smaller) collection size. For n experiments, we compute the correspondence for the computed base shape pair, where each base shape pair can be considered at most once. This procedure gives a set of $m \leq n$ pairs, on which we run the other methods and average the results. As shown in Table 4.1 and Table 4.2, in most cases we indeed get a corresponding base shape pair.

Parameters for comparison methods. For BIM, we use the default parameters (no need to set them maually). When applying BCICP we use the parameters recommended by the authors [RPWO18]: $k_1 = k_2 = 50$ (the number of eigenfunctions used for each shape), the weight for the orientation-preserving term was set to 0.1 and we used 10 iterations for the BCICP refinement step. For the FAUST data-set we set the time-scale parameter for computing the WKS descriptors to 100 and the skip size for the computed descriptors to 10. For all the other data-sets, we use the parameters suggested for the non-isometric TOSCA data-set, i.e. the time-scale parameter for the WKS descriptors was set to 50.

Mesh normalization. While our method does not require all the meshes to have the same surface area, other methods do, hence we normalized all the meshes to the same area.

5.3.3 Blend Shapes data-set

We use our method to compute the correspondences for the Blend Shapes data-set (using 10 shapes) and the results appear in Figure 5.2. For this data-set, our algorithm did not benefit from RHM refinement while the other methods did. It is seen both qualitatively and quantitatively that our method achieves the best performance while the others fail to compute a high-quality

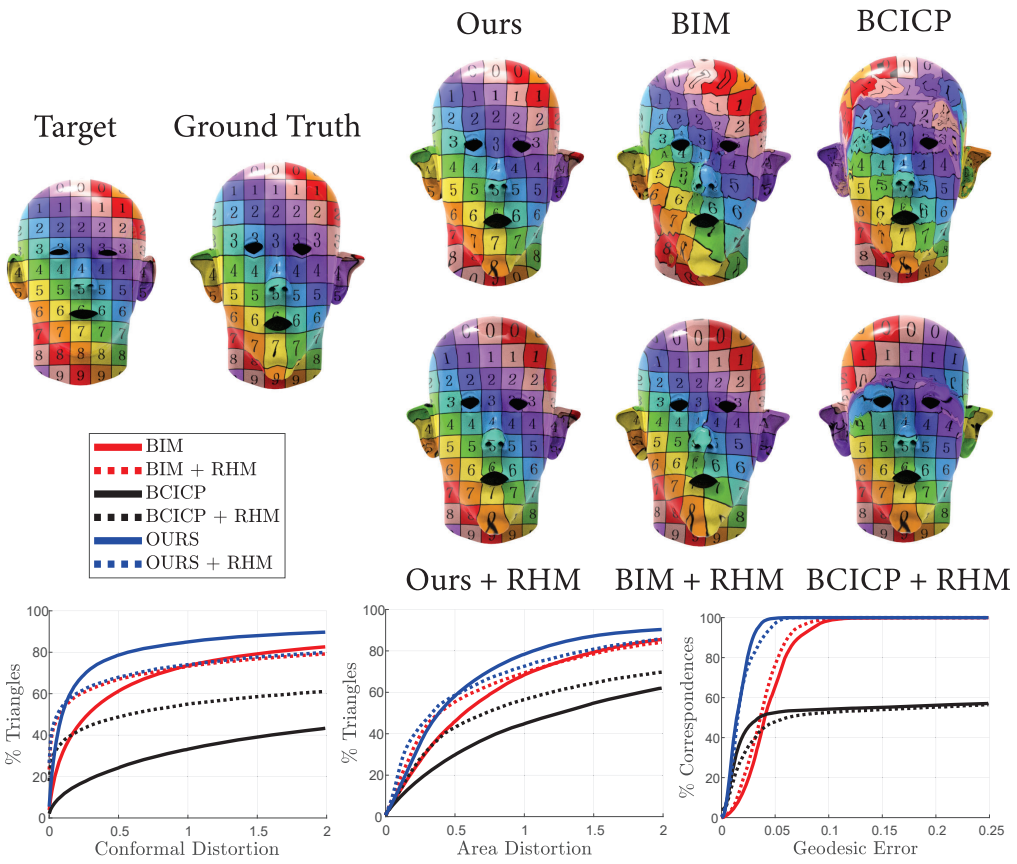


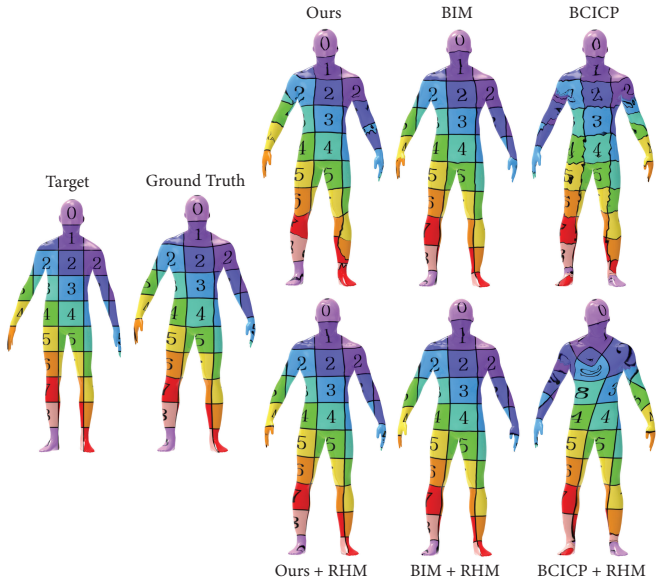
Figure 5.2: Blend Shapes data-set - qualitative (top) and quantitative (bottom) comparison. Our method compared to other automatic methods - BIM [KLF11] and BCICP [RPWO18], with and without final refinement using RHM. Our method yields high-quality maps, while others suffer from distortions or do not resolve symmetry ambiguities.

map or resolve symmetry ambiguities (BCICP flips the upper part of the head while maintaining the correct orientation of the bottom part).

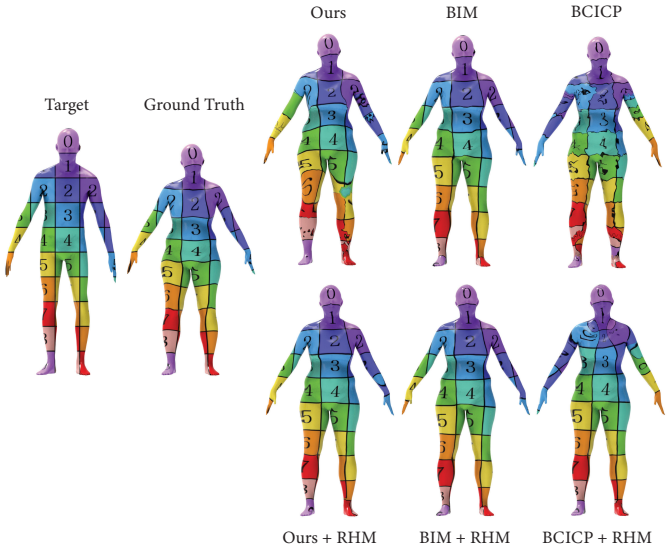
5.3.4 FAUST data-set

For the FAUST data-set, we use the ten subsets that represent ten different people as collections and use all the subset pairs to evaluate our algorithm (45 combinations in total). For the chosen base shapes in each subset pair we compute the resulting map using the comparison methods. When RHM is not used, BIM obtains preferable results. However, after the refinement our method achieves similar conformal and area distortion, yet the smallest distance from the ground truth (Figure 5.3 (c)). Note, that BIM is only applicable to genus zero shapes, whereas our method (including the RHM post-processing) is applicable to any mesh topology.

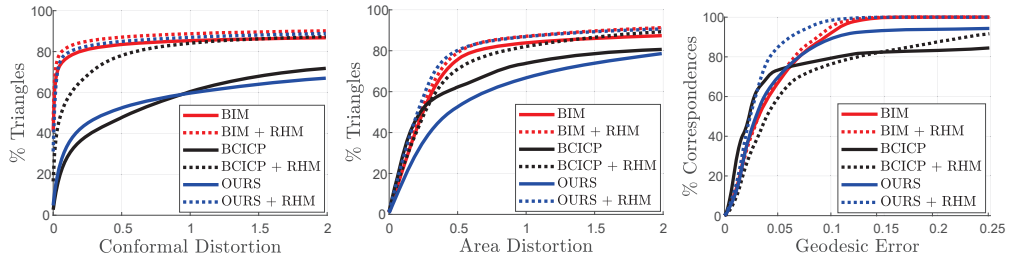
Qualitatively, Figure 5.3 (a) shows that our algorithm refined with RHM obtains the highest quality map. Note that the feet and hands areas are closer to the ground truth than BIM. On the



(a) Our algorithm refined with RHM obtains the highest quality map: Note that the feet and hands areas are closer to the ground truth than BIM. BCICP cannot completely resolve symmetry ambiguity, flipping the hands.



(b) Despite imperfections of our results on the hands and feet, our map refined with RHM leads to a smaller geodesic error.



(c) Quantitative results.

Figure 5.3: FAUST - qualitative and quantitative comparison: Our method and other automatic methods - BIM and BCICP, with and without final refinement using RHM.

other hand, BCICP cannot completely resolve the symmetry ambiguity, flipping the right arm with the left one, while maintaining the correct orientation for the other parts. In Figure 5.3 (b), despite imperfections of our method on the hands and feet, our map refined with RHM still obtains the smallest geodesic error. We conjecture that BCICP has difficulties resolving symmetry ambiguities. In our case, the base shape pairs computed by our algorithm are, in most cases, corresponding pairs, yielding more symmetric shapes for *both* the source and the target than in averaging over all random pairs in this data-set (most of the shapes in this data-set are extrinsically symmetric). Moreover, BCICP appears to be very sensitive to the parameters of the WKS descriptors, which need to be tuned for each data-set separately.

5.3.5 Sumner data-set

The Sumner data-set does not have ground truth correspondences, hence we show only conformal and area distortion as well as a qualitative visualization. For this data-set, after RHM refinement we get conformal distortion similar to that of BIM and a smaller area distortion. The qualitative results in Figure 5.4 show that we indeed get high-quality maps. Since every time we run the algorithm we might get two different shapes that we put in correspondence, we get a different choice of base shape pair than in Figure 1.1, demonstrating the results on another pair.

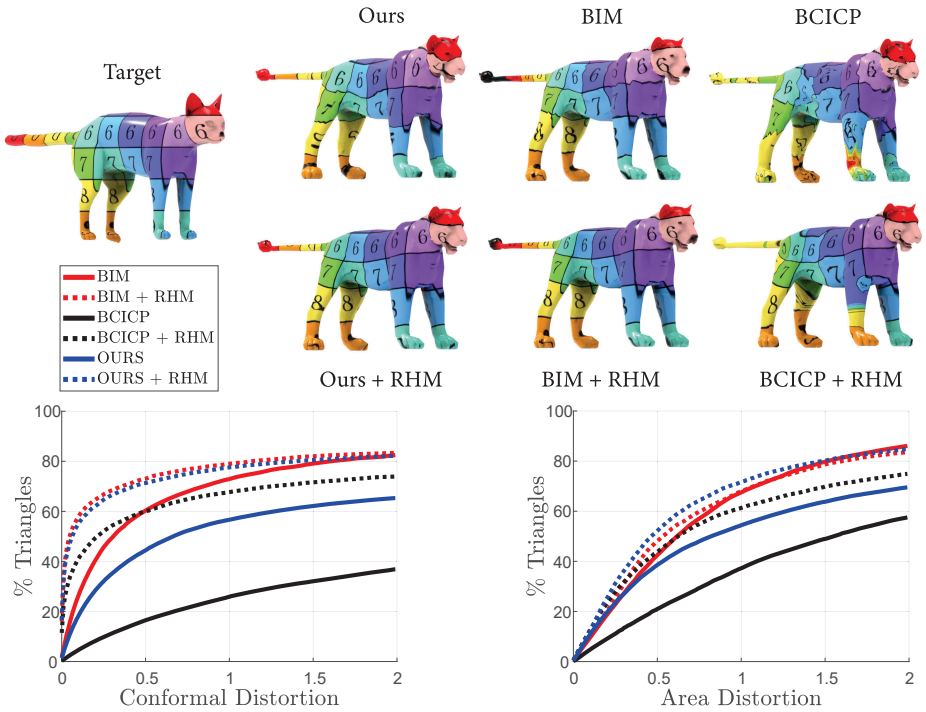
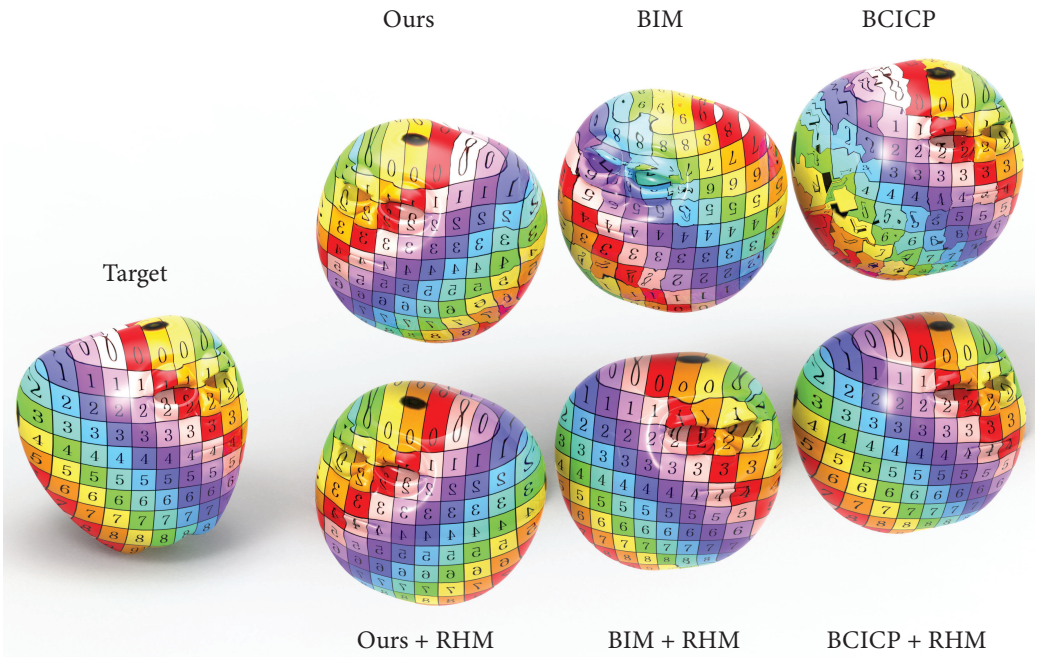


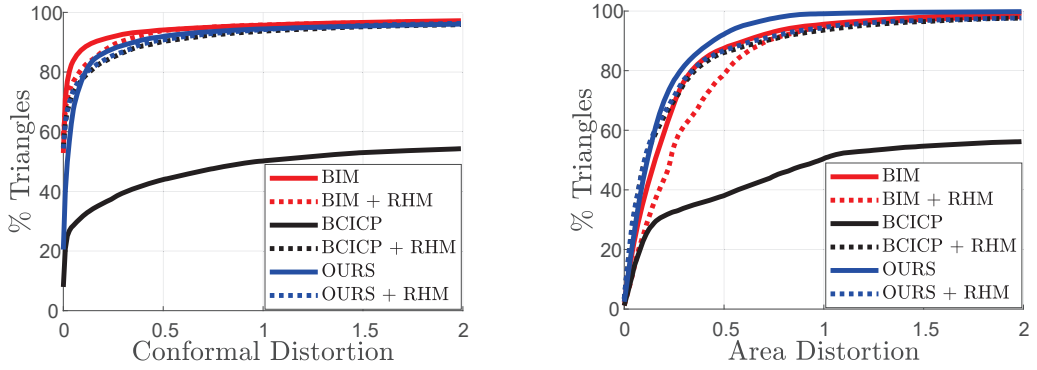
Figure 5.4: Sumner, qualitative and quantitative comparison: Our method and other automatic methods - BIM and BCICP, with and without final refinement using RHM. Note that we achieve conformal distortion similar to BIM, yet a better area distortion. The qualitative comparison indeed shows that our result is preferable.

5.3.6 Rigged fruit data-set

We acquired this dataset on TurboSquid, and the ground truth map is not available. In this data-set all the shapes are symmetric, hence, the maps obtained cannot distinguish between left and right. However, note that BIM also flips the map upside-down, which is fixed using RHM refinement. In this data-set, when no RHM refinement is applied, our method yields the best map since it is not as noisy as BCICP and does not flip up and down as BIM. After RHM refinement good results are obtained for all methods (see figure 5.5).



(a) Note that BIM flips the map upside-down, which is fixed using refinement with RHM.



(b) Quantitative results.

Figure 5.5: Rigged fruit - qualitative and quantitative comparison. Our method and other automatic methods - BIM and BCICP, with and without final refinement using RHM.

5.4 Inter-Map Computation using Composition of Maps

We now demonstrate that it is possible to obtain the inter-map from *any* shape in collection $\mathcal{A}$ to *any* shape in collection $\mathcal{B}$ using the map composition in Eq. 3.5, exploiting the inter-map for the *corresponding* base shapes and the given intra-maps. Figure 5.6 shows the case of an inter-map for *non*-corresponding shapes. First, we show the case of an inter-map computed *directly* for the two shapes (as if they were chosen as base shapes). As explained in section 4.1, this case of non-corresponding base shapes yields poorer maps as indeed can be seen. Second, we visualize the map obtained using Eq. 3.5 where the base shapes are corresponding (as in Figure 5.3). In this case, the computed maps are of high-quality. We also compare to the results obtained using BIM and BCICP. In all cases maps are visualized after RHM post-processing.

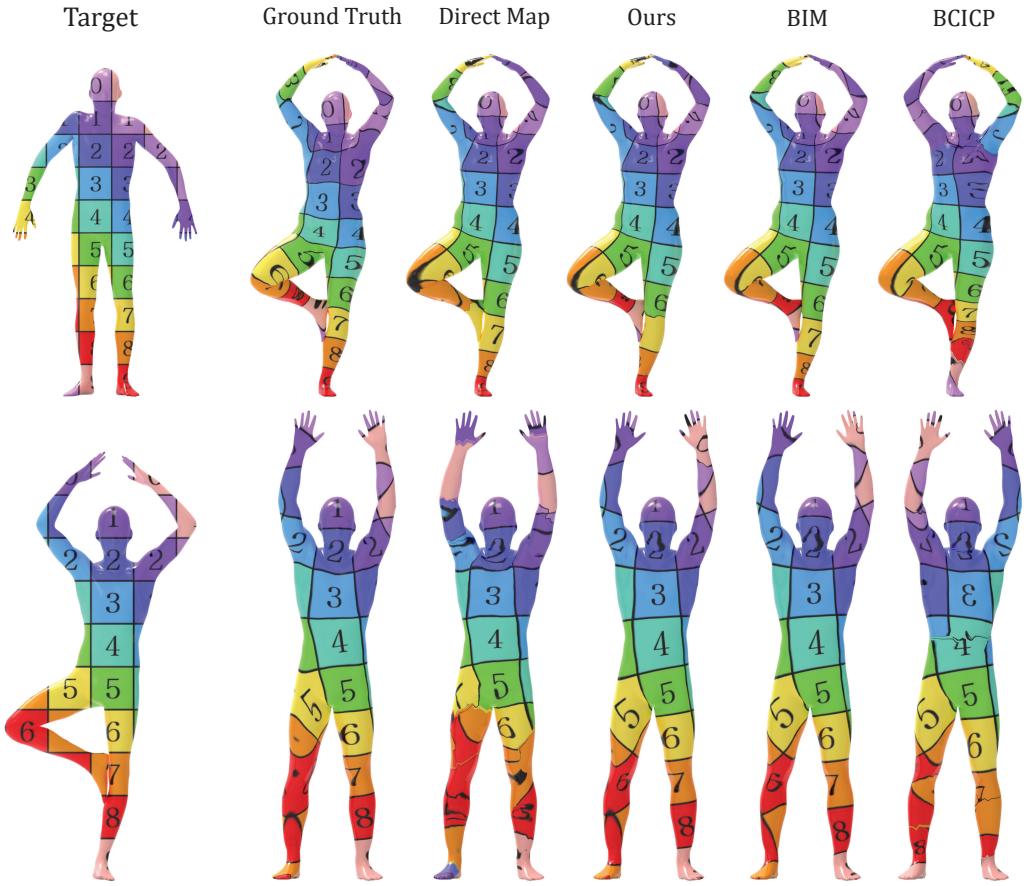


Figure 5.6: Composition of maps. Using map composition we can get a high-quality inter-map for non-corresponding shapes rather than computing the map directly for those shapes. All visualized maps include post-processing with RHM. See the text for details.

5.5 Inter-Map Computation with Low Matching Accuracy

After we showed that the cross collection map computation using our method leads to good results both qualitatively and quantitatively, we test the resilience of this part of the algorithm when the matching accuracy, determined in the first part of the algorithm, is low. Using the ground truth of the matches, we set them manually to obtain the desired accuracy and test the results of the map computation. The matching accuracy affects the analogies constraints (Eq. (3.4)), since we use also wrong matches as constraints. We show that our approach can handle wrong matching in the first phase of the algorithm and still retrieve high-quality maps.

In Figure 5.7 we demonstrate the experiment for FAUST data-set, all the results and visualizations include RHM refinement. Note that starting from 80% accuracy it is possible to obtain high-quality maps. In other words, the algorithm can tolerate the wrong matching of two pairs, when we use collections with ten shapes.

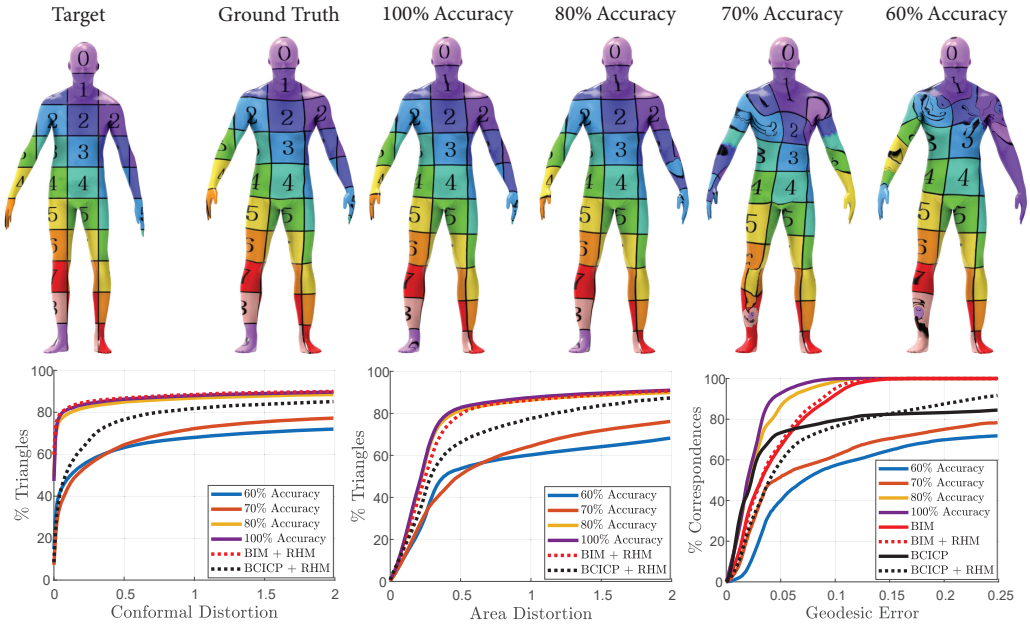


Figure 5.7: FAUST - qualitative (top) and quantitative (bottom) comparison for varying matching accuracy derived from the first part of our algorithm. We manually set the matches to allow varying accuracy and test the derived map from the second part of the algorithm. Note that starting from 80% we get high-quality maps. All results include RHM as post-processing

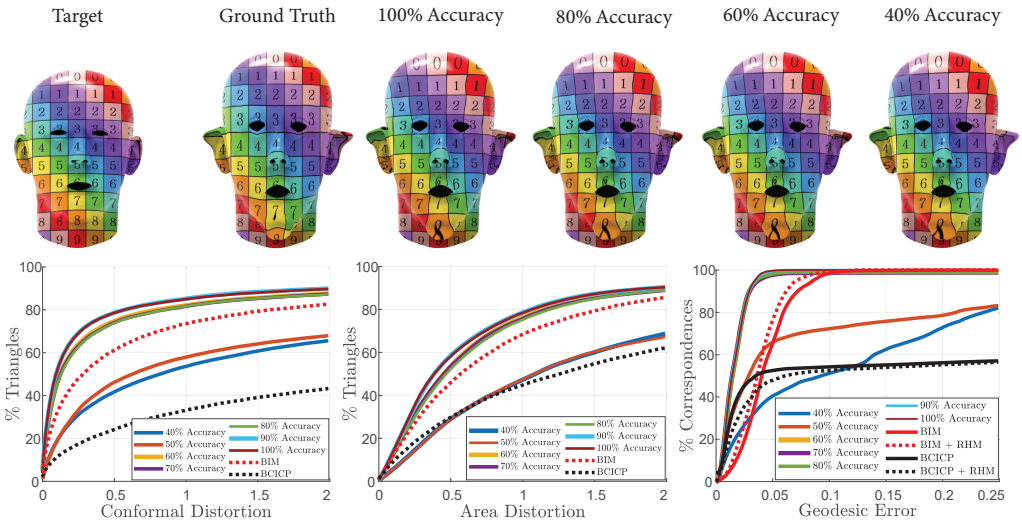


Figure 5.8: Blend shapes - qualitative (with RHM post-processing, top) and quantitative (without RHM post-processing, bottom) comparison for varying matching accuracy derived from the first part of our algorithm. We manually set the matches to allow varying accuracy and test the derived map from the second part of the algorithm. It is possible to obtain a high-quality refined map with matching accuracy as low as 40%.

Figure 5.8 shows the same experiment for the Blend Shapes data-set (using 40 shapes). Here we visualize the maps after RHM refinement, all achieving similar conformal and area distortion and high-quality maps Figure 5.8 (top). We conclude that in spite of the low matching accuracy, RHM refinement is able to recover a map very similar to the ground truth. Hence, to show the differences between different matching accuracies, we plotted the conformal and area distortion for the results *without* RHM refinement, Figure 5.8 (bottom). It is seen that for this data-set high-quality maps can be obtained even when the matching accuracy is as low as 40%. Since this data-set has more shapes than in FAUST (40 shapes in contrast to 10 in FAUST), we have more terms in the energy function (Eq. (3.4)) and since the variations within the shapes are localized to the face region, it is possible to recover the functional map with lower matching accuracy.

5.6 Inter-Map Computation for Varying Collection Size

We evaluate the computed map depending on the collection size, especially for small collections, i.e. with collection size as low as 2 or 3. We demonstrate that computing correspondences using our method is possible for such small collections, allowing us to retrieve high-quality maps even when only few shapes are given. We tested our algorithm for varying collection size and examined the map retrieved.

In Figure 5.9 we see that for the Blend Shapes data-set it is possible to get a high-quality map using any collection size varying from 2 to 40. For the case with 2 shapes, we set the matches manually since aligning point clouds with two points each has 50% success rate as

they are necessarily symmetric. Results are shown *without* RHM refinement. This result means that we can get the cross collection map when we have only three shapes in each collection, or two with manual matching, and still get higher-quality map than other methods, with very low timing, since for smaller collection size the time needed for our algorithm notably decreases. It is important to mention that matching accuracy remained as high as 100% in all cases of collection sizes, even for 3 shapes only (except the manual matching for the case with 2 shapes).

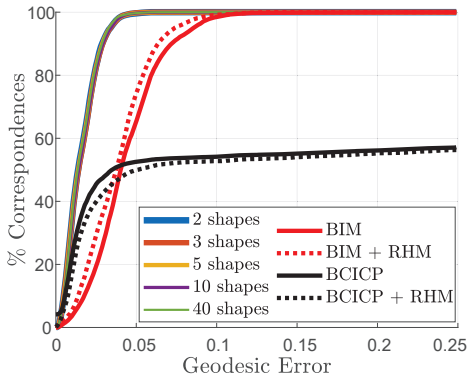


Figure 5.9: Blend Shapes - quantitative comparison for varying collection size without RHM post-processing. Note that we can obtain a high-quality map with collections containing 2 shapes only.

Figure 5.10 shows the same experiment for Sumner data-set. Note that the conformal and area distortion of the map are not affected when we decrease the number of shapes, neither is the matching accuracy. Our results include post-processing using RHM. As in the demonstration for the Blend Shapes data-set, we set the matches manually for the case with 2 shapes in a collection, to handle the case of symmetric shape space.

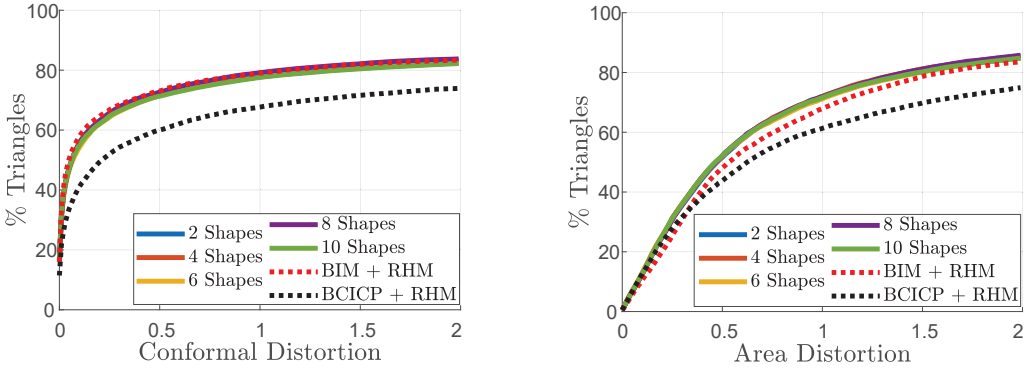


Figure 5.10: Sumner, quantitative comparison, varying collection size. All results include RHM post-processing. Note that we obtain a high-quality map even for collections with only two shapes.

5.7 Regularization Effect

In order to show that at least one analogy is needed, and the functional map using Eq. (3.4) for the case of 2 shapes in a collection is not recovered solely due to the regularization term, we demonstrate the quality of the functional map retrieved for varying α, the parameter controlling the weight of the regularization term, and for varying collection size from the Blend Shapes dataset. We measure the error of the optimized functional map $C_{N_{BS},M_{BS}}$ (after ICP refinement), and plot the error as a function of α (Figure 5.11). The error is given by the distance from the ground truth functional map C_{GT}, namely $\|C_{N_{BS},M_{BS}} - C_{GT}\|_F$.

We can conclude that:

- For $\alpha = 0$ (no regularization), the error is negatively correlated with the collection size: the larger the collection, the smaller the error. In other words, the more analogies we have, the higher-quality functional map we can infer (see the numbers on the vertical axis showing the decreasing error as the collection size increases).
- There exists a range of α, for any collection size, for which we can get the same best-optimized functional map with the same error, i.e., with good regularization we can decrease the dependency on the collection size (around $0.1 \leq \alpha \leq 0.15$).
- For larger α than in the optimal range, optimization is more tolerant as the collection size increases since more analogies give more information.
- Using no analogies at all, relying on the regularization term only (equivalent to $\alpha \to \infty$), cannot yield a good functional map. For all the graphs, as α increases the error indeed converges to the error value with no analogies term (the black horizontal line).

Hence, we deduce that the analogies are indeed required, yet a small number of analogies is sufficient in order to retrieve a high-quality map, which is advantageous, since less data is needed.

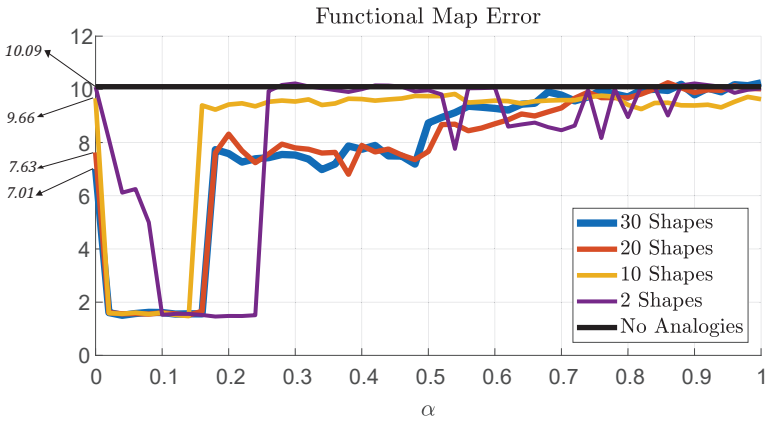


Figure 5.11: Functional map error for varying regularizer weight and several collection sizes. See the text for details.

Chapter 6

Correspondence between Two Shapes

In this section, we handle the problem of correspondence between two shapes only, i.e., each collection has only one shape, and show that our method can yield correspondences without requiring a whole collection. As previously shown, in order to obtain a pointwise inter-map, our algorithm requires collections of at least two shapes, since we need at least one analogy to solve the optimization problem (Eq. (3.4)). However, if only one shape is given we can *create* a collection from this shape using the method introduced in [HWAG09] by using modal analysis. Therefore, we perform pre-processing to the pipeline shown in Figure 1.2 in the following way: Each of the two given shapes acts as an input to the method proposed in [HWAG09, Sec. 4], where the Hessian of the shape is computed, for which the eigen-vectors corresponding to non-zero eigen-values act as the *modes* of the shape. By adding these modes to the given mesh, we can create a whole new collection out of one shape.

Figure 6.1 demonstrates the collections obtained from a given single shape (left column). One can notice that if the shapes are not too non-isometric, we obtain similar collections for which the shape difference operators can be computed and then it is possible to align the two collections. Note that even though shape (3) is in different pose than (1,2,4,5), the computed modes are similar and yield collections with similar *analogies*, such that it is possible to align them and solve the optimization problem (Eq. (3.4)).

Once we obtained two *collections* we can apply our algorithm and compute the inter-map for the two given shapes, acting as base shapes. Notice that the algorithm also matches the newly composed shapes which is necessary for solving the optimization problem. However, this matching is hidden from the user and is not returned as an output, since the input was not comprised of collections.

In order to allow the composed collections to be matched in a way that enables us to compute an inter-map, we need to assume that the two new shape spaces, derived from the new collections, can be aligned. In other words, it requires the modes of the two shapes to be similar enough, though not necessarily in the same order. To obtain such similar modes, we generally need similar shapes, i.e., neither in too different poses nor too non-isometric.

Figure 6.2 shows our algorithm's performance on inputs of two shapes only. We show the pointwise map obtained, with and without RHM refinement. We also show the optimized

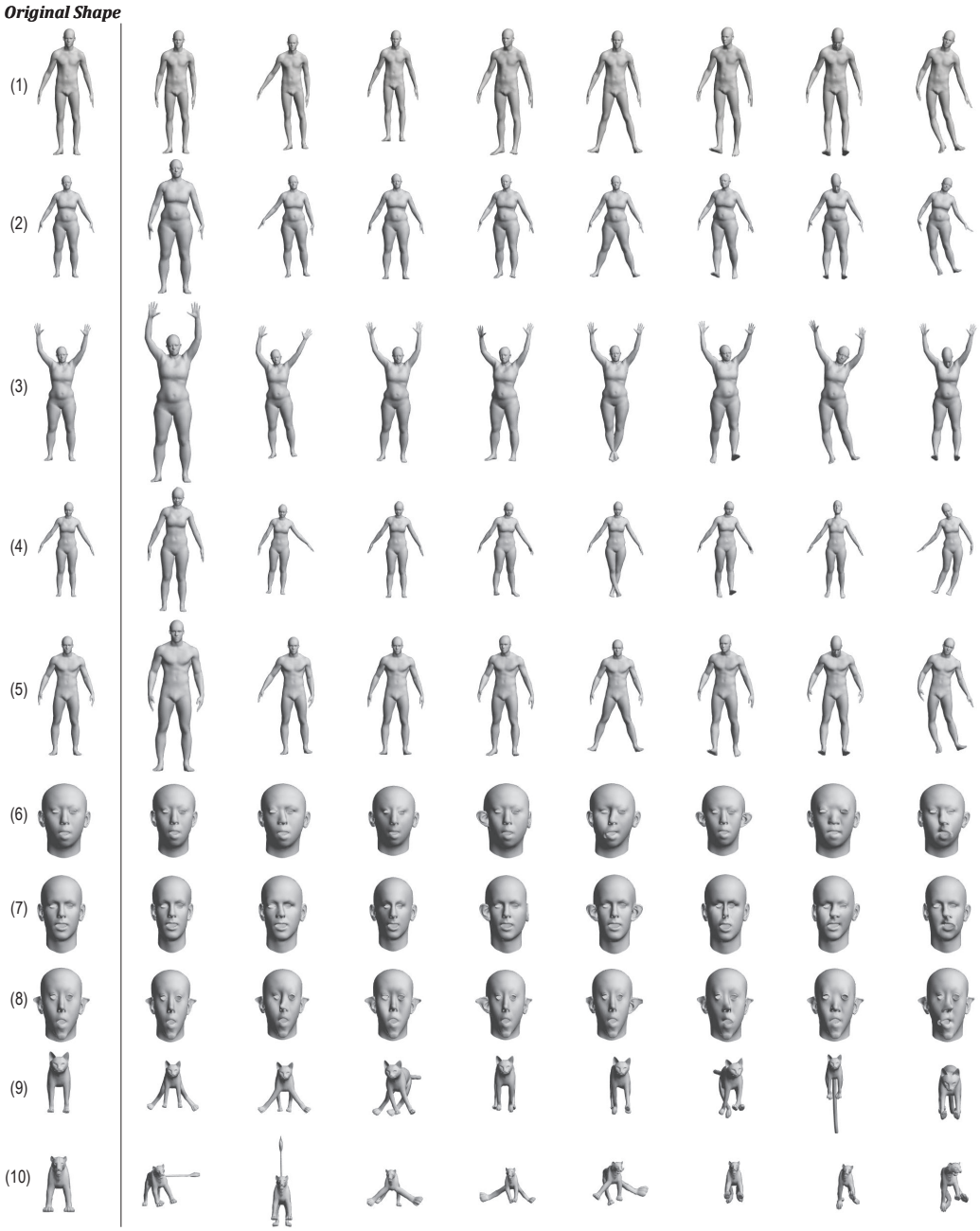


Figure 6.1: Input shapes (left column) and the collections created using their modes by computing the Hessian matrix as explained in [HWAG09].

functional map, the corresponding ground truth (if it exists) and an example of function transfer for better visualization. In most cases, we obtain a high-quality map which can be well refined by post-processing with RHM.

Note that Figure 6.2:(2) shows an example of a map which is flipped in the upper part, however, post-processing with RHM fixes it and returns a high-quality consistent map. Figure 6.2:(3) shows that even if the shapes are not in the same pose we can still get an excellent map since we get similar modes yielding similar analogies (see Figure 6.1:(1-5)). However, Figure 6.2:(6) gives an example for which our algorithm fails to obtain a good map. If carefully examined, it can be seen in Figure 6.1:(9-10) that many shapes do not correspond, hence, aligning and inferring analogies is now challenging and it is hard to obtain a high-quality map in this case. We conclude that these two shapes are not isometric enough to apply this method successfully.

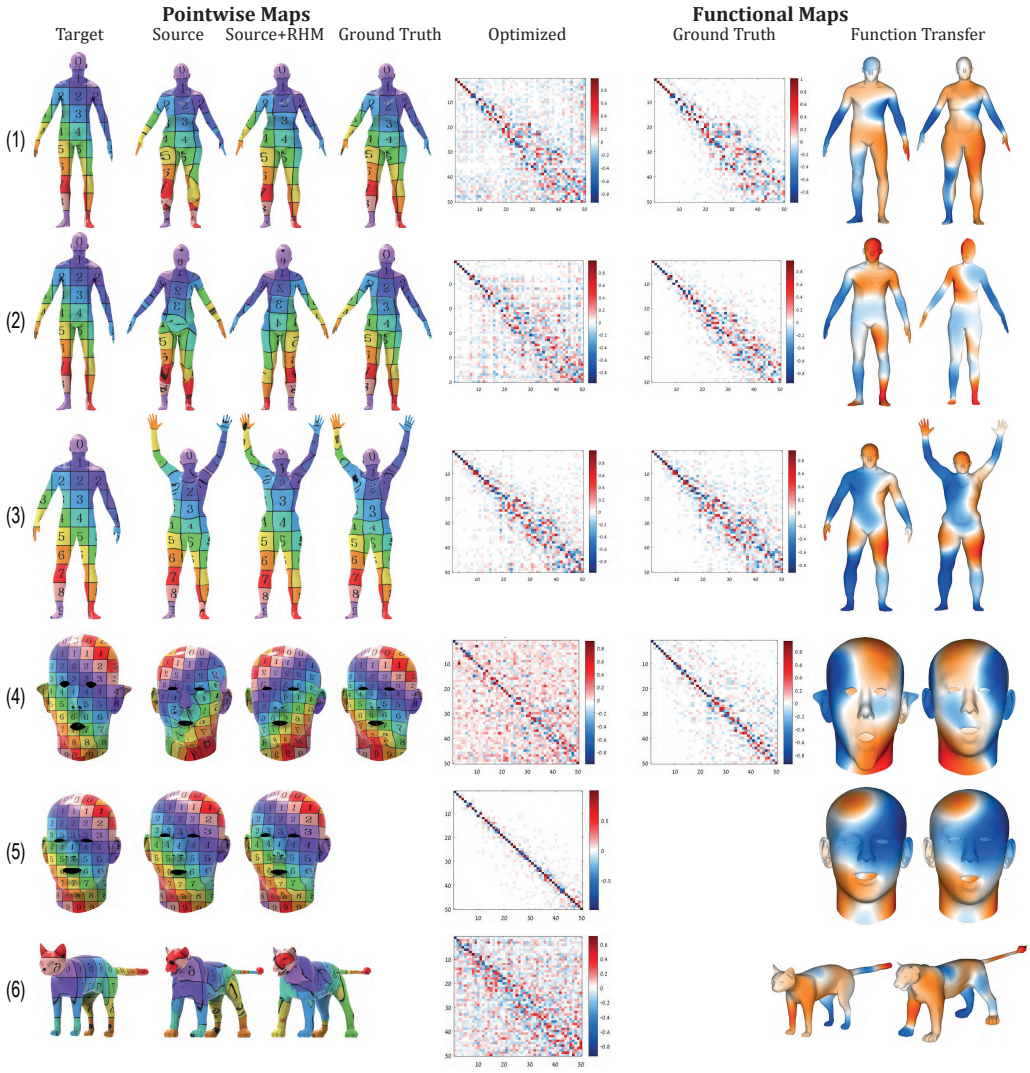


Figure 6.2: Correspondence between two shapes - qualitative results on various data-sets. The computed pointwise map is shown with and without RHM post-processing, as well as the optimized and ground truth functional maps (if it exists).

Chapter 7

Conclusion

In this work, we presented a robust and simple to implement method for matching two shape collections with high rates of accuracy. Unlike previous approaches, our method can perfectly handle small and large collections alike, while for the latter we maintain high accuracy rates even when one of the collections contains excess shapes that do not need to be matched, thus allowing to handle various noisy sampling of the shape space.

Furthermore, a high-quality inter-map is obtained using only the analogies as the semantic information, allowing our algorithm to serve as a fully-automatic method for computing correspondences, surpassing other state-of-the-art automatic methods for non-isometric shape correspondence. As an additional proof of its robustness, we showed that even if not every shape has a match or matching was imperfect, the second step of the algorithm, computing the inter-map, is resilient to it. Finally, the variation within the collection can be large, and both quasi-isometric or non-isometric making our approach applicable to various data-sets.

We believe that our approach can serve as an important new tool in the shape analysis and correspondence toolbox. Future work and generalizations include using other inner product metrics to tailor the shape differences for specific applications [CO19], and using better regularization constraints [RPWO19]. Moreover, it might be beneficial to use *multiple* base shapes, and to learn the shape differences operators from data. Finally, since the functional map approach is agnostic to the geometry representation, it can be interesting to apply our approach to settings where the data has an intrinsic parametric structure allowing to generate many corresponding instances, e.g. parametric CAD models.

Appendix A

Additional Comparison with SBC14

A.1 Algorithm Outline Comparison

In the next page, we give an outline of the two methods and emphasize the differences (in **bold**). From the comparison one can conclude that:

- Computing the **shape irregularity index** for all shapes in both collections is computationally costly compared to random base shape choice and recomputing the shape difference operators using the optimal base shape. Therefore, we decrease the running time without reducing our algorithm's performance.

- **Non-linear** dimensionality reduction is sensitive to the collection size and requires large collections in order to obtain reasonable results. However, we use **linear** dimensionality reduction and are thus resilient to the collection size, enabling the algorithm to handle small and large collections alike.

- Our alignment method is parameter free in contrast to CPD used in SBC14. Hence, our method does not require data-set dependent parameters, allowing it to be more robust and completely automatic.

- Solving the optimization problem without a **regularization term** leads to poor and unstable results. While, as previously demonstrated, our methods achieves high quality and stable inter-maps.

- Recovering a **pointwise map** (steps 6-7 in our method) was not applicable using [SBC14], as demonstrated in Section A.2.

Our Algorithm:

1. Use the input intra-maps to construct the shape differences between the input shapes in the same collection.
2. Use the shape differences to construct a low-dimensional shape space embedding for both collections:
   (a) Choose a base shape **randomly**.
   (b) Compute the shape difference operator for each shape with respect to the **randomly chosen** base shape.
   (c) Construct the shape space by embedding the shape difference operators in a low dimension space by **linear** dimensionality reduction using **SPD distance and multidimensional scaling (MDS)** [Mea92].
3. Align the two shape spaces using **Procrustes analysis (by convex semidefinite programming (SDP) relaxation [MDK$^+$16])** to obtain the matching pairs and **automatically determine an *optimal base shape* in each collection**.
4. **For each collection, recompute the shape difference operators with respect to the new base shape.**
5. Use the matches and the base shapes to compute a functional inter-map for the base shape pair by solving an optimization problem with an analogies term and **a regularization term**.
6. **Recover a point-to-point inter-map from the functional map using [EBC17]**.
7. **Optional: Refine the map obtained in (6) using reversible harmonic maps (RHM) [ESBC19]**.

SBC14's Algorithm:

1. Use the input intra-maps to construct the shape differences between the input shapes in the same collection.
2. Use the shape differences to construct a low-dimensional shape space embedding for both collections:
   (a) Choose a base shape using **shape irregularity index**.
   (b) Compute the shape difference operator for each shape with respect to the **computed** base shape.
   (c) Construct the shape space by embedding the shape difference operators in a low dimension space by **non-linear** dimensionality reduction using **diffusion maps**.
3. Align the two shape spaces using **affine registration (by coherent point drift (CPD)** to obtain the matching pairs.
4. Use the matches and the base shapes to compute a functional inter-map for the base shape pair by solving an optimization problem with an analogies term **only**.

A.2 Pointwise Map Comparison

Now we would also like to compare the *pointwise* map can be extracted from the functional map. Figure A.1 compares the obtained point-to-point map using our method and using SBC14, with and without RHM refinement on some shapes from FAUST dataset. It can be seen that SBC14 cannot achieve a fair map, not even as an input for RHM post-processing. Hence, we conclude that the method of SBC14 lacks the fine tuning needed to obtain comparable pointwise maps to state-of-the-art methods.

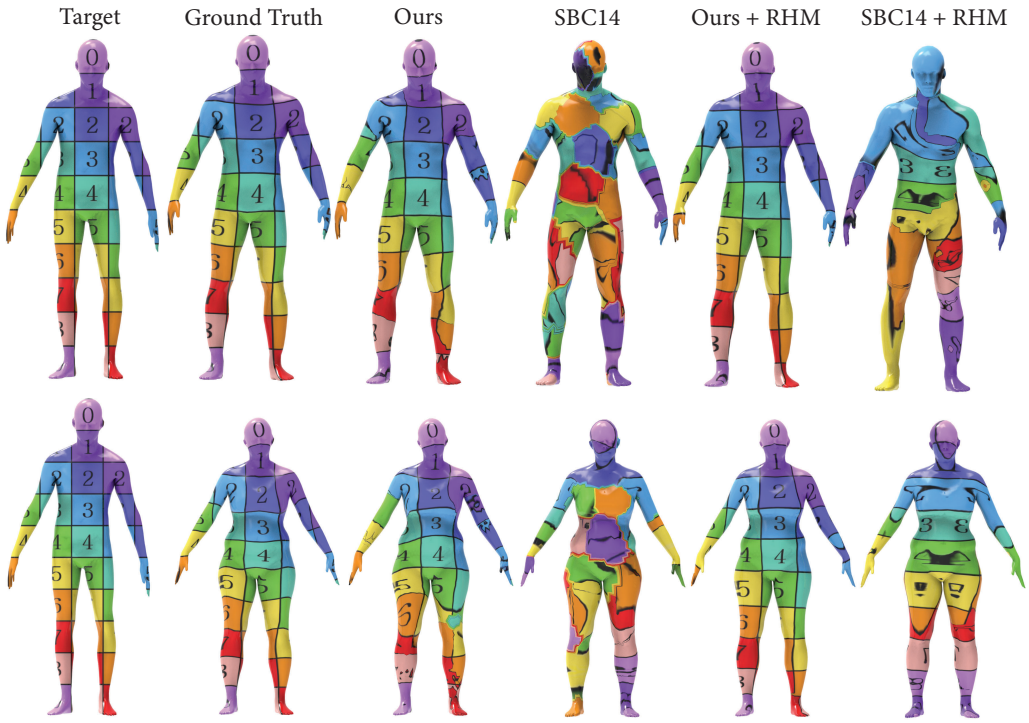


Figure A.1: FAUST - qualitative comparison with SBC14, with and without final refinement using RHM.

Acknowledgments

This thesis was partially supported by the European Research Council (ERC starting grant No. 714776 OPREP), and the Israel Science Foundation (grant No. 504/16).

Bibliography

[Bha09] Rajendra Bhatia. *Positive definite matrices*, volume 16. Princeton university press, 2009.

[CO19] Etienne Corman and Maks Ovsjanikov. Functional characterization of deformation fields. *ACM Transactions on Graphics (TOG)*, 38(1):8, 2019.

[CRA+17] Luca Cosmo, Emanuele Rodola, Andrea Albarelli, Facundo Mémoli, and Daniel Cremers. Consistent partial matching of shape collections via sparse modeling. In *Computer Graphics Forum*, volume 36, pages 209–221. Wiley Online Library, 2017.

[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.

[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.

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

[GSDG18] Vignesh Ganapathi-Subramanian, Olga Diamanti, and Leonidas J Guibas. Modular latent spaces for shape correspondences. In *Computer Graphics Forum*, volume 37, pages 199–210. Wiley Online Library, 2018.

[GYQ+18] Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. Automatic unpaired shape deformation transfer. In *SIGGRAPH Asia 2018 Technical Papers*, page 237. ACM, 2018.

[HAGO19] Ruqi Huang, Panos Achlioptas, Leonidas Guibas, and Maks Ovsjanikov. Limit shapes–a tool for understanding shape differences and variability in 3d model collections. In *Computer Graphics Forum*, volume 38, pages 187–202. Wiley Online Library, 2019.

[HCO18] Ruqi Huang, Frédéric Chazal, and Maks Ovsjanikov. On the stability of functional maps and shape difference operators. In *Computer Graphics Forum*, volume 37, pages 145–158. Wiley Online Library, 2018.

[HG00] Kai Hormann and Günther Greiner. Mips: An efficient global parametrization method. Technical report, ERLANGEN-NUERNBERG UNIV (GERMANY) COMPUTER GRAPHICS GROUP, 2000.

[HRA+19] Ruqi Huang, Marie-Julie Rakotosaona, Panos Achlioptas, Leonidas Guibas, and Maks Ovsjanikov. Operatornet: Recovering 3d shapes from difference operators. *arXiv preprint arXiv:1904.10754*, 2019.

[HWAG09] Qi-Xing Huang, Martin Wicke, Bart Adams, and Leonidas Guibas. Shape decomposition using modal analysis. In *Computer Graphics Forum*, volume 28, pages 407–416. Wiley Online Library, 2009.

[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.

[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.

[Mea92] Al Mead. Review of the development of multidimensional scaling methods. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 41(1):27–39, 1992.

[NBCW+11] Andy Nguyen, Mirela Ben-Chen, Katarzyna Welnicka, Yinyu Ye, and Leonidas Guibas. An optimization approach to improving collections of shape maps. In *Computer Graphics Forum*, volume 30, pages 1481–1491. Wiley Online Library, 2011.

[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.

[ROA+13] Raif M Rustamov, Maks Ovsjanikov, Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Leonidas Guibas. Map-based exploration of intrinsic shape differences and variability. *ACM Transactions on Graphics (TOG)*, 32(4):72, 2013.

[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.

[RPWO19] Jing Ren, Mikhail Panine, Peter Wonka, and Maks Ovsjanikov. Structured regularization of functional map computations. In *Computer Graphics Forum*, volume 38, pages 39–53. Wiley Online Library, 2019.

[SBC14] Nitzan Shapira and Mirela Ben-Chen. Cross-collection map inference by intrinsic alignment of shape spaces. In *Computer Graphics Forum*, volume 33, pages 281–290. Wiley Online Library, 2014.

[SLHH18] Yifan Sun, Zhenxiao Liang, Xiangru Huang, and Qixing Huang. Joint map and symmetry synchronization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 251–264, 2018.

את פתרון בעיית המיפוי בשיטות אלה לפתרון שאינו אוטומטי לחלוטין.

אנו מציעים שיטה אוטומטית לחלוטין שמתאימה בין אוספי צורות וגם מחשבת את המיפוי בין כל צורה באוסף הראשון לכל צורה באוסף השני. בהינתן המיפויים בין הצורות השייכות לאותו אוסף, שהינם כאמור בדרך כלל פשוטים יחסית לחישוב שכן מדובר במציאת המיפוי בין צורות איזומטריות, אנו מחלצים את אופרטורי הפרשי הצורות המתאימים לאוסף ונעזרים בהם על מנת לבנות "מרחב צורה" לכל אחד משני אוספי הצורות. את מרחב הצורה אנחנו מנסים לשכן במרחב ממימד נמוך ככל האפשר, אך ששומר את רוב המידע על השינויים בתוך אותו אוסף. לאחר מכן, על ידי כך שאנו מתייחסים לשיכון האופרטורים במרחב הצורה כענן נקודות במרחב ממימד נמוך, אנו מחפשים את ההתאמה הטובה ביותר בין שני ענני הנקודות בעזרת טרנספורמציות קשיחות בלבד. במילים אחרות, אנו מחפשים מטריצת סיבוב ומטריצת פרמוטציה שיביאו להתאמה הטובה ביותר בין ענני הנקודות. באופן כללי, בעיה זו קשה לפתרון אך אנו עושים שימוש בשיטה חדשנית הפותרת בעיה זו ביעילות רבה. למעשה, ההתאמה בין ענני הנקודות מהווה את ההתאמה בין זוגות הצורות התואמים משני האוספים. לבסוף, אנו עושים שימוש בהתאמה זו על מנת להגדיר בעיית אופטימיזציה שמטרתה למצוא את המיפוי בין פונקציות עבור זוג צורות תואם מסוים. כאשר מתקבל מיפוי איכותי בין פונקציות ניתן לשחזר את המיפוי מנקודה לנקודה עבור הזוגות התואמים משני האוספים. כך באמצעות מיפוי זה והמיפויים הנתונים בתוך כל אוסף ניתן לקבל את המיפויים בין כל צורה באוסף הראשון לכל צורה באוסף השני וזאת בעזרת הרכבת המיפויים.

בניגוד לשיטות קיימות למציאת ההתאמות בין אוספי צורות, השיטה שלנו יכולה להתמודד עם אוספי צורות קטנים וגדולים כאחד, ואינה דורשת כיוונון של פרמטרים כתלות באוספי הצורות הנתונים. מעבר לכך, בניגוד לרוב השיטות הקיימות למציאת המיפוי בין צורות שאינן איזומטריות, השיטה שלנו עושה שימוש בשונות בין הצורות באוסף עצמו על מנת לקבל את המיפוי בין צורות מאוספים שונים, ולכן אינו דורש קלט נוסף כמו נקודות עניין או דסקריפטורים.

אנחנו מראים שהשיטה שלנו משיגה שיעורי התאמה גבוהים במיוחד ומחשבת מיפויים איכותיים בין צורות שאינן איזומטריות. כמו כן אנו מדגימים שהשיטה שלנו משתווה לאלגוריתמים האוטומטיים העדכניים ביותר עבור מציאת המיפוי בין צורות לא איזומטריות ואף משיגה אותם. לבסוף, אנו מדגימים שבמקרים מסוימים, ניתן לקבל באמצעות השיטה שלנו מיפוי איכותי מבלי לדרוש שני אוספים, אלא עבור קלט המורכב מזוג צורות בלבד, וזאת באמצעות יצירת אוסף באופן אוטומטי מתוך צורה בודדת.

תקציר

אוספי צורות משמשים כבסיס מחקרי רחב ליישומים רבים בעיבוד גיאומטרי ובגרפיקה ממוחשבת. אוסף צורות עשוי להתקבל למשל בעזרת דפורמציות של מודל תלת־מימדי נתון או באמצעות דגימה של אנימציה תלת־מימדית. בהינתן שני אוספי צורות, כמו לדוגמא שתי דמויות במגוון תנוחות תואמות, לעיתים עולה הצורך למצוא התאמה סמנטית בין זוגות תואמים של צורות. התאמה זו יכולה לשמש להעברת מידע בין שני האוספים. למשל ניתן להעביר תיוגים בין הזוגות המותאמים ועל ידי כך לסווג את התנוחה (עמידה, שכיבה, הליכה, ריצה וכדומה).

אתגר נוסף הוא למצוא באופן אוטומטי מיפוי מנקודה לנקודה בין זוג צורות מאוספים שונים, כאשר במקרים רבים צורות אלה אינן איזומטריות. מציאת המיפוי בין צורות הינה משימה עיקרית וחשובה בניתוח צורות: בהינתן שתי צורות, המטרה היא למצוא מיפוי בעל התאמה סמנטית בין נקודות על שתי הצורות. ההתאמה בין הצורות נחוצה לניתוח משותף שלהם, דבר שנפוץ באפליקציות רבות בגרפיקה ממוחשבת כגון העברת דפורמציות, העברת טקסטורה, ניתוח סטטיסטי, סיווג צורות ועוד.

ניתן לחלק את אתגר מציאת המיפוי בין הצורות 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 האחרים מאחר שכמות דרגות החופש הינה קטנה ומרחב הטרנספורמציות האפשריות בין הצורות קל לייצוג. הסוג השני הוא התאמה בין צורות שאינן קשיחות אך כן איזומטריות, למשל עבור אותו אוביקט תלת־מימדי הנמצא בתנוחות שונות. מציאת ההתאמה במקרה זה קלה יותר מאשר במקרה הלא איזומטרי, כיוון שבמקרה האיזומטרי ישנו קריטריון איכות ברור עבור המיפוי והוא שימור המרחקים הגאודזיים. 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, למרות שלאדם משימת ההתאמה הינה קלה, ניסוח מתמטי מדויק של הבעיה הינו קשה ועל אלגוריתם שמוצא מיפוי בין הצורות להתאים בין נקודות על הצורות בצורה סמנטית וכן להפחית מדד עיוות מקומי כלשהו. במקרים רבים, יש צורך במידע נוסף על מנת למצוא מיפוי איכותי בין צורות לא איזמטריות, כמו למשל נקודות עניין בעלות משמעות סמנטית דומה לשתי הצורות. לפעמים כדי לקבל נקודות עניין כאלה יש לסמנן באופן ידני על הצורות, דבר שהופך

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

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי.

התאמה ומיפוי בין אוספי צורות באמצעות הפרשי צורות

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר
מגיסטר למדעים בהנדסת חשמל

אהרון כהן

הוגש לסנט הטכניון --- מכון טכנולוגי לישראל
שבט התש"פ חיפה פברואר 2020

התאמה ומיפוי בין אוספי צורות באמצעות הפרשי צורות

אהרון כהן