ספריות הטכניון
The Technion Libraries

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

Cross-Collection Map Inference by Intrinsic Alignment of Shape Spaces

Nitzan Shapira

Cross-Collection Map Inference by Intrinsic Alignment of Shape Spaces

Research Thesis

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science

Nitzan Shapira

Submitted to the Senate
of the Technion — Israel Institute of Technology
Tishrei 5775 Haifa September 2014

The research was carried out under the supervision of Prof. Mirela Ben-Chen, in the Faculty of Computer Science.

I would like to thank my supervisor, Prof. Mirela Ben-Chen, for the many hours of joint work and successful collaboration.

The generous financial help of the Technion is gratefully acknowledged.

Contents

List of Figures

Abstract

Inferring maps between shapes is a long standing problem in geometry processing. The less similar the shapes are, the harder it is to compute a map, or even define criteria to evaluate it. In many cases, shapes appear as part of a *collection*, e.g. an animation or a series of faces or poses of the same character, where the shapes are similar enough, such that maps *within* the collection are easy to obtain.

Our main observation is that given two collections of shapes whose "shape space" structure is similar, it is possible to find a correspondence between the collections, and then compute a cross-collection map. The cross-map is given as a functional correspondence, and thus it is more appropriate in cases where a bijective point-to-point map is not well defined. Our core idea is to treat each collection as a point-sampling from a low-dimensional shape-space manifold, and use dimensionality reduction techniques to find a low-dimensional Euclidean embedding of this sampling.

To measure distances on the shape-space manifold, we use the recently introduced *shape differences*, which lead to a similar low-dimensional structure of the shape spaces, even if the shapes themselves are quite different. This allows us to use standard affine registration for point-clouds to align the shape-spaces, and then find a functional cross-map using a linear solve. We demonstrate the results of our algorithm on various shape collections and discuss its properties.

Abbreviations and Notations

A, B	—	two shape collections
M, N	—	two shapes
$f : M \to \mathbb{R}$	—	a real-valued function defined on a surface M
$T : M \to N$	—	a bijective map between shapes M and N
$F : L^2(M) \to L^2(N)$	—	a map between the function spaces of M and N
$V_{M,N}$	—	the area-based shape difference between shapes M and N
$R_{M,N}$	—	the conformal-based shape difference between shapes M and N
σ_i	—	the i-th singular value
γ	—	the threshold for the intrinsic dimension estimation
G	—	the unknown functional map we wish to approximate
PCA	—	*Principle Component Analysis*
GMM	—	*Gaussian Mixture Model*
ICP	—	*Iterative Closest Point*
CPD	—	*Coherent Point Drift*
SDD	—	*Shape Difference Distance*
VDM	—	*Vector Diffusion Maps*

Chapter 1

Introduction

1.1 Background

Shape correspondence and shape collection analysis are fundamental tasks in geometry processing, at the core of many applications, such as animation, geometric modeling, 3D scanning and analysis of medical data, to mention just a few. Shape correspondence refers to the task of, given two shapes, finding a correspondence between them, while shape collection analysis refers to the task of, given a shape collection, obtain a meaningful insight about it, or improve certain aspects of it. A shape collection is set of shapes. In order to be able to perform a meaningful analysis, we would usually want to have the maps between the shapes in the collection. The different tasks that we would like to perform on a shape collection may include: improving the consistency of its maps, shape matching between two collections, and more.

In many cases, correspondences between shapes are represented as a point-to-point map, taking points on the first shape to points on the second. Such maps are appropriate when the shapes are similar, e.g. different poses of the same person. For more complicated cases, e.g. the same pose of different characters, it is not always clear which points should be in correspondence. In such cases, it is easier to model a correspondence as a function-to-function map [19], taking functions on the first shape to functions on the second. This approach is more flexible and allows us to encode the uncertainty inherent to the solution of an ill-posed problem.

Often, shapes do not appear in isolation, but in collections of related shapes with certain *structure*. For example, a densely sampled animation between two poses has a different structure than a set of unrelated poses. Extracting such a structure, and *aligning* two collections with similar structure, namely finding a correspondence between the shapes, is an important task, which can aid, e.g. statistical analysis of medical imaging data. If the collection is homogeneous, i.e., the shapes are similar to each other, it is often feasible to obtain a good correspondence between shapes within the collection, and then leverage this information for further analysis of the collection's structure.

We address the following problem: given two homogeneous shape collections with

intra-collection maps, we seek a correspondence between the shapes (namely which shape in collection A maps to which shape in collection B), as well as the cross-map between all the shapes (see Fig. 1.1). While this seems harder than finding a map between every pair of shapes in the two collections, we demonstrate that the structure within each collection can be extracted and represented concisely, such that if the structures are similar, the collections can be aligned. This allows us to compute only maps between *corresponding* shapes, which are easier to obtain.

1.2 Our Approach

To align two collections of shapes whose geometries are different, we need to define a shape representation such that the similar structure can emerge. Thus, we need a representation which can encode relations like "shape A is to B, like C is to D", and allow us to compare *differences of differences* of shapes. Recently, exactly such a representation was introduced [20], where the difference between two shapes, for which we have a correspondence, can be represented using a *linear operator* which acts on functions on one of the shapes. This representation is based on the *functional maps* framework [19], which has already been used in various contexts. Comparing two such operators provides a meaningful measure of the shape difference as it can encode not only whether two shapes are different, but also *where* they differ. This brings all the "shape differences" in the same collection to a common ground, and allows us to use standard techniques for Euclidean point clouds to perform the analysis.

However, our data is usually high-dimensional. In order to achieve an effective and meaningful representation for the distances between the shapes, we would like to transform the data into a lower dimension. Therefore, we assume that each collection includes shapes which are sampled from a low-dimensional shape space manifold. We then use *diffusion maps* [6], a standard non-linear dimensionality reduction technique to find a low-dimensional Euclidean embedding which reproduces as best as possible the original intrinsic distances on the shape space manifold. We repeat this procedure for both collections and obtain a low-dimensional embedding of both collections. After achieving these embeddings, a "shape" is simply a point in a low-dimensional point cloud, and the distances between the shapes are simple Euclidean distances.

After obtaining two point clouds, we would like to align them. This alignment is a standard registration of point clouds. Our chosen method of registration is using an affine one, which enables the registration algorithm to tolerate a certain error in the distances between the shapes, since the collections may not be exactly aligned. We also allow reflection, to compensate for the isomentries which may arise after the dimensionality reduction step. Our chosen algorithm for the affine registration is *coherent point drift* [17], a state-of-the-art cloud registration algorithm which is suitable for affine registration.

Once the collections are aligned, we can use the assumption on the common structure, to specify constraints of the type "A is to B as C is to D" as linear constraints on

a functional map which maps a shape from the first collection to a shape in the second. These constraints are enough to recover the cross-collection functional map *without requiring any additional descriptors*. After defining these constraints, the cross-collection functional map can be easily computed using a linear solve. We then extend this map to the rest of the collection, through composition, which puts both collections in correspondence.

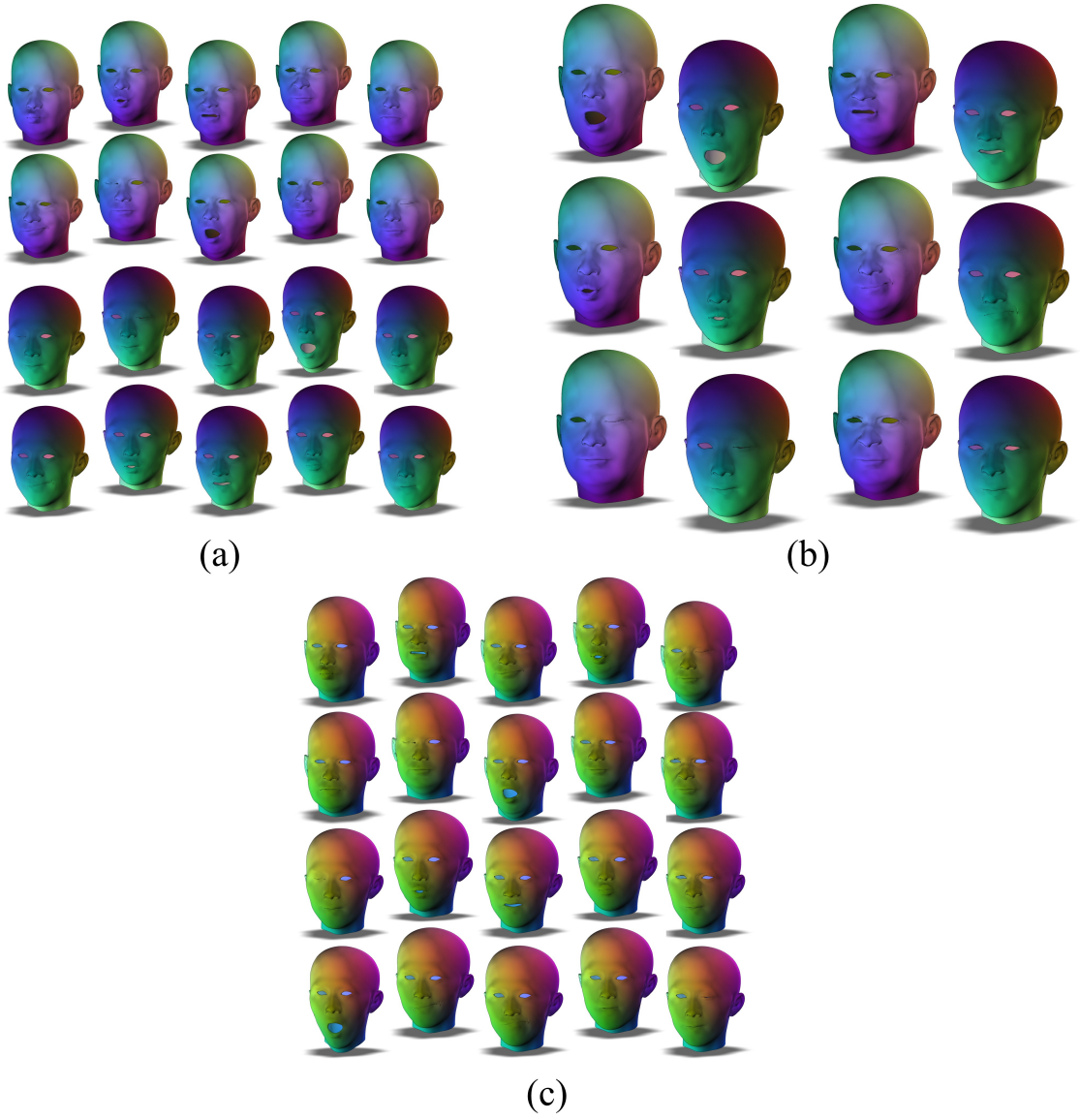

Figure 1.1: Using our method, we can find correspondences and map functions defined on shapes from two collections, given only maps within the same collection. This map is easy to compute, and provides a meaningful representation of the relation between shapes from different collections, for which a point-to-point map is difficult to obtain and is not well defined. (a) original collections, (b) collection alignment, (c) functional map approximation between all shapes in both collections.

Chapter 2

Previous Work

The concept of registering two shape collections using dimensionality reduction techniques has not (to the best of our knowledge) been proposed before, nor has the task of finding a cross-collection functional map. Nevertheless, the tasks of finding a map between two shapes and the analysis of shape collections have been widely studied in different contexts.

2.1 Finding a Map Between Two Shapes

The task of finding a map between two shapes has been studied from different points of view, under various assumptions, e.g. in [15, 19]. See [23] for a recent survey. Various methods have been proposed to recover a map between two shapes. The main disadvantages of existing methods is that it requires some prior, e.g. the user must define a set of corresponding points on both shapes in order for the algorithm to produce a valid result. In [7], the authors used Classical MDS to construct an invariant signature for isometric surfaces, which is an embedding of the geodesic structure of the surface in a small dimension in which geodesic distances are approximated by Euclidean ones, therefore translating the problem of matching non-rigid objects into a simpler problem of matching rigid objects. It is worth noting that our method incorporates much more information, as the whole collection is considered when computing the map.

2.2 Shape Collections and Map Consistency

Considering the entire collection when computing maps *within* the collection has also been addressed previously, e.g. in the context of improving the maps inside a shape collection. In [10–12, 18], the authors add the constraint of global map consistency in order to improve a set of initial maps, using different optimization methods. In addition, structural information from the collection is commonly used for co-segmentation, e.g. in [13, 14, 26], among others. Another work related to the idea of search engines for shape models, presented in [2], uses structural descriptors of shapes in order to find partial

correspondences between pairs of shapes, enabling efficient searching and exploration within the shape collection. In general, we differ from these methods by our assumption of having *two* homogeneous shape collections, with good maps within the collection, instead of having a single heterogeneous collection. This allows us to assume there exists common structure, and use it to align the collections as a whole.

2.3 Dimensionality Reduction of a Shape Collection

Dimensionality reduction of a shape collection is also a technique which has been used in the past. Spectral analysis of shape collections using dimensionality reduction techniques has been used in [9,24,25] for shape de-noising. Other related works have been studying the pre-image problem for a shape collection. In [24], the authors proposed a way to compute the pre-image of a shape collection using diffusion maps and Karcher Means. The pre-image was computed using a Nyström extension. Another work which used projection into a shape manifold is [8], where the projection was performed using a non-linear interpolation according to a Delaunay triangulation of the collection. In [20], the authors proposed a way to represent the intrinsic shape-space of a collection using *functional operators*. This representation allows to represent differences between shapes as *comparable objects* instead of merely distances, and then to embed the shape-space in two dimensions, e.g. for visualization. Another related work is [1], in which the authors used dimensionality reduction on a collection of facial images, in order to both analyze and synthesize these kind of images. The authors have also dealt with the challenge of finding a meaningful comparison method for two such images, and chose the Jacobs, Belhumeur, and Basri (JBB) method. The dimensionality reduction was performed using Least-Squares multidimensional scaling. However, we also find correspondences between shapes from the two different collections, a challenge which has not been addressed in [1].

In both [20] and [1], the dimensionality reduction was performed using a linear technique such as Principal Component Analysis or MDS, which assumes the underlying shape-space manifold is *linear*. In many cases, though, such an assumption is too restrictive, and projecting the shape-space on a linear manifold might destroy its structure. We use non-linear dimensionality reduction instead to overcome this difficulty.

2.4 Video Sequence Registration

Another usage for the registration of two collections is in the context of video sequences [4,5,16]. In these studies, the purpose is to align two video sequences using registration techniques. There, however, the problem is better posed, as there are considerably less parameters and more available data.

2.5 Difference from Our Approach

To conclude, the main difference between previous work and our approach is that none of the previous approaches attempted to register directly two shape collections. The main obstacle to doing that is finding a common representation such that an alignment is possible. By leveraging the functional approach, namely considering maps and shape differences as linear operators, we can map both shape-spaces to a common space, where alignment is possible. Furthermore, we can use this alignment to compute a functional map, which takes into consideration both collections as a whole.

2.5.1 Algorithm Outline

Given two shape collections A and B with internal correspondences, we do the following:

- Choose a *base shape* in each collection, and use it to calculate the shape differences representation.
- Reduce the dimensionality of the collections using diffusion maps on the shape differences representation to obtain two point clouds, then use affine registration to align the clouds, and find corresponding shape pairs.
- Define *shape-analogies constraints* between corresponding pairs and the base shapes, and obtain an approximated functional map between the base shapes.

The outline of the algorithm is also presented in Figure 2.1.

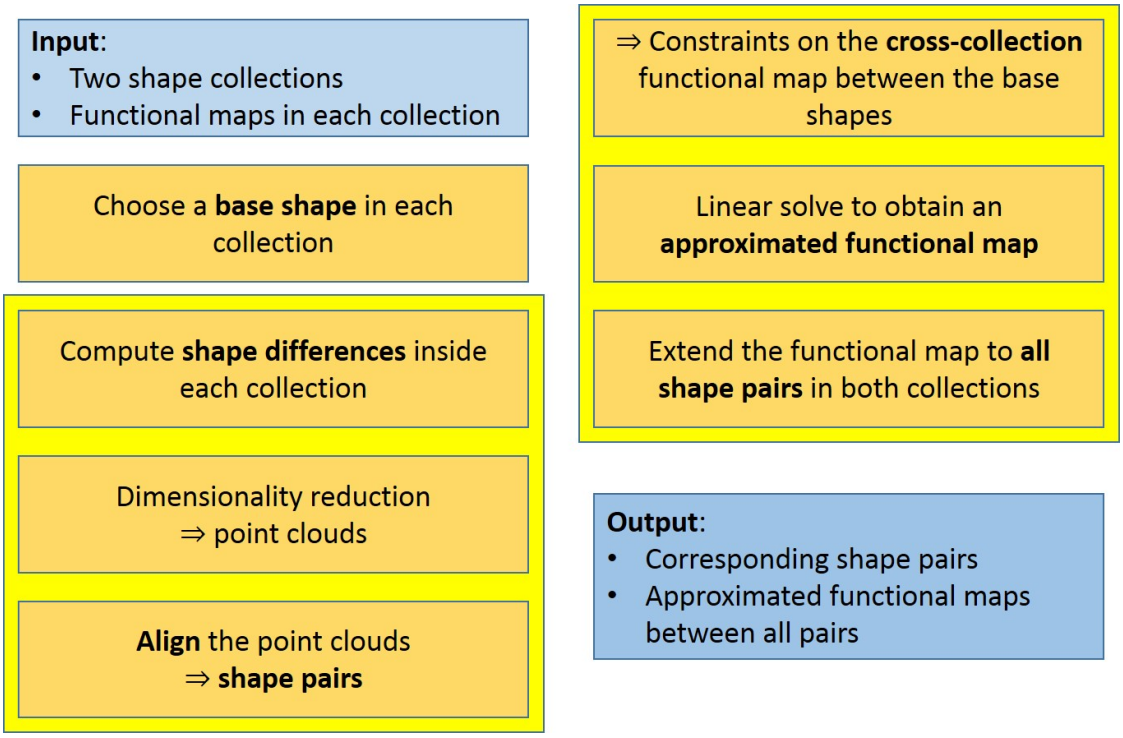

Figure 2.1: The pipeline of our algorithm.

Chapter 3

Collection Representation

Given two shape collections, we assume they are sampled from two low-dimensional space-shape manifolds. Our goal is to represent each shape as a point in $\mathbb{R}^n$, such that Euclidean distances in this representation have some intrinsic geometric meaning, and we can later align the resulting point clouds. If we are given additionally maps between every two shapes *within* each collection, we can use these maps to compute a notion of a *shape difference*. This is a linear operator, which encodes the *variation* induced by the map. Assuming that shapes which are corresponding in two collections undergo a similar transformation under a map from some *base shape*, e.g. the change from a neutral face to a frowning face is similar for two different characters, such a shape difference would provide the intrinsic representation we require.

3.1 Assumptions

For our algorithm to be applicable, we must make a few assumptions on the input shape collections. First, we assume that both collections are *homogeneous*, e.g. represent the same character or the same object in different poses, and we are given maps between all pairs. Next, we assume that the collections have *similar structure*. Specifically, this means that there is a subset of shapes in both collections which can be paired, such that the differences between them are similar. For example, in Figure 3.1 we show two shapes (a cat and a lioness) which have different geometries, however the difference between the two cats is similar to the difference between the two lionesses. Furthermore, we assume that for each collection there exists a *base shape* which can be used to evaluate the shape differences to all the other shapes in the collection. Finally, we assume that the low dimensional Euclidean embeddings of the two collections are *not symmetric*, and thus can be reliably registered.

We emphasize that the two collections can differ in their number of shapes. In such cases, some of the shapes in the larger collection are expected to be considered as outliers, enabling the other shapes to be matched successfully. Figure 3.2 shows an example of the alignment of two collections with a different number of shapes: a dense and a sparse

Figure 3.1: Similar *structure* in two collections. Although the geometry of the cat (left) differs from the geometry of the lioness (right), the *difference* between the cats is similar to the difference between the lionesses.

sampling from an animation sequence.

3.2 Functional Maps

The recently proposed functional maps framework [19] is used to represent maps between surfaces. We use this framework in order to consider functional map on surface, rather than the standard bijective point-to-point maps. This is a more general notion of a map and is more suitable for cases in which a point-to-point map is not always meaningful. A functional maps functions defined on surfaces. Namely, given two surfaces M and N, a map $T : N \to M$ between them induces a map between function spaces $F : L^2(M) \to L^2(N)$, where $L^2(\cdot)$ is the set of square integrable real-valued functions on a surface. This *functional map* F takes a function $f : M \to \mathbb{R}$ and maps it to $g : N \to \mathbb{R}$, and is defined using $g = F(f) = f \circ T$. In addition, the original map T can be recovered from F. As explained in [19], F is a *linear* transformation between function spaces. Therefore, given a choice of basis, it can be represented as a matrix in the discrete setting. An example of such a functional map is presented in Figure 4.3.

3.3 Shape Differences

A *shape difference* [20] is a linear operator which encodes the disparity between two shapes M and N under a given functional map F. We use the two types of shape differences defined in [20]. The first type is based on the area distortion and the other is based on the conformal distortion, as induced by the map. Together these two shape differences completely encode the map F.

The *area-based shape difference* is marked $V_{M,N}$ and the *conformal-based shape difference* is marked $R_{M,N}$, where the respective map is usually clear from the context. The specific calculation for the discrete case is demonstrated in Section 3.4. We emphasize that both $V_{M,N}$ and $R_{M,N}$ are not numbers but *operators*. Note that two shape differ-

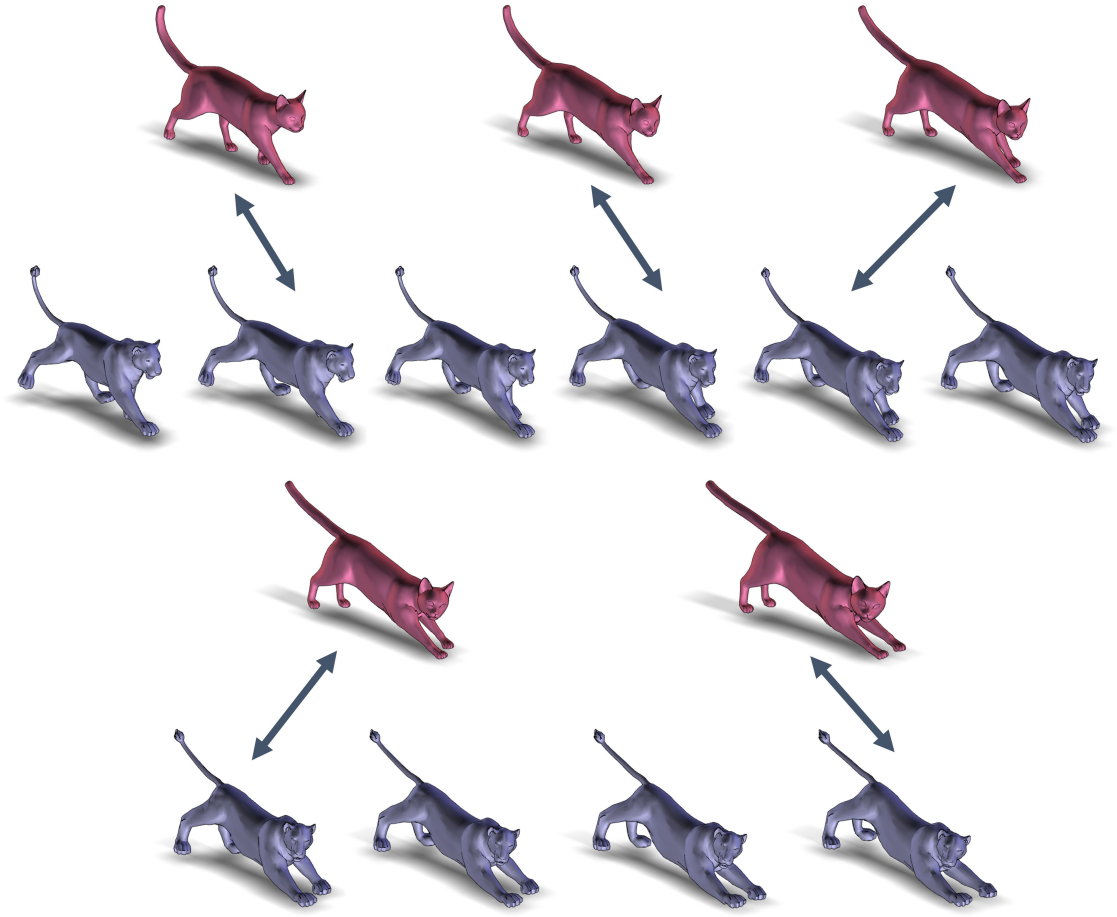

Figure 3.2: Two similar animations of different shapes, sampled at different rates (30 shapes and 60 shapes). Some shapes in the dense collection are considered as outliers, as they do not have compatible matches in the sparse collection. After removing the outliers, the remaining shapes are matched correctly.

ences, V_{M,N_1} and V_{M,N_2}, represent linear operators with the same domain and range, $L^2(M)$, even if $N_1 \neq N_2$. Hence, the shape difference between M and N_1 is comparable to the shape difference between M and N_2, as they are both linear operators acting on functions on M. As explained in [20], the shape differences matrices encode the map up to an area preserving, or conformal self-map, therefore they are fully informative up to the given notion of distortion.

3.4 Discrete Representation

In order to represent a functional map discretely, we need to pick a basis for the space of discrete functions on meshes. We choose the eigenvectors of the Laplace-Beltrami operator, as proposed in [20], as it provides a multi-scale basis which allows to represent smooth functions with a small number of basis functions. As described in [20], given a

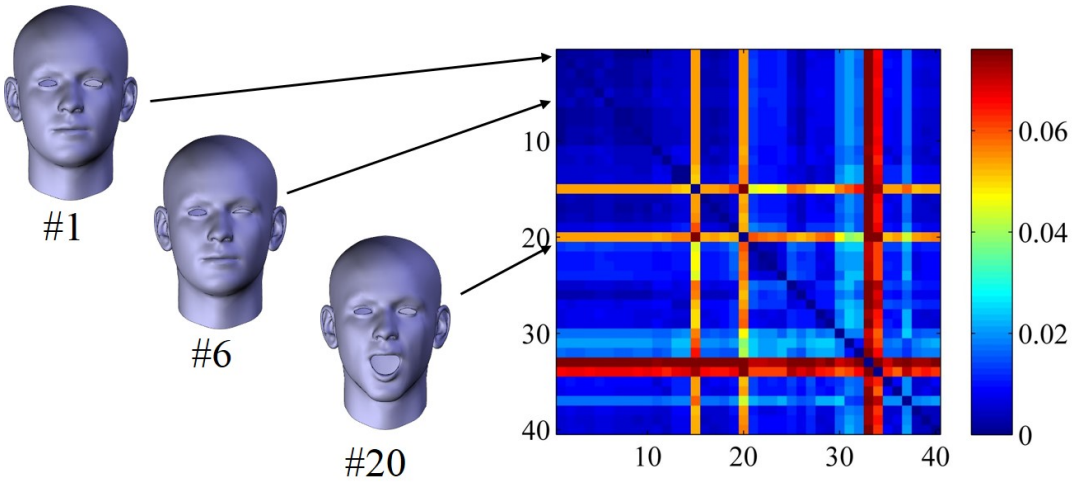

Figure 3.3: The *shape difference distances* (SDDs) between shape pairs in a collection of 40 blend shapes. The more different the shapes are, the higher the SDD is.

functional map F we compute the shape differences using:

$$V_{M,N} = F^\top F, \quad \text{and} \quad R_{M,N} = (D^M)^{-1} F^\top D^N F, \tag{3.1}$$

where $D^M = diag(-\{\lambda_i^M\})$, λ_i^M is the i^{th} eigenvalue of the Laplacian of M, and similarly for N. We typically use between 30 and 70 eigenfunctions for the representation.

Finally, the *shape difference distance* (SDD) between two shapes N_1, N_2, given a base shape M is defined as:

$$d_M(N_1, N_2) = \sqrt{\|V_{MN_1} - V_{MN_2}\|_F^2 + \|R_{MN_1} - R_{MN_2}\|_F^2} \tag{3.2}$$

Figure 3.3 shows the SDDs between shape pairs in a collection of 40 blend shapes.

3.5 Base Shape Selection

We represent all the shapes in a collection as a shape difference to a chosen *base shape* M. Specifically, each shape is represented as a linear operator which takes functions on M and returns functions on M, therefore, all the shape differences are encoded in the basis of M (Figure 3.4).

It has been shown in [20], that if there is cycle consistency in the collection (namely, given any three shapes M, N, K, and functional maps $F_{MN} : L^2(M) \to L^2(N)$ and similarly for F_{NK}, F_{MK}, we have $F_{MK} = F_{NK} F_{MN}$), then it is possible to transport shape differences between different base shapes by applying a change of basis. Specifically, given a functional map G from M_1 to M_2 we can compute $\tilde{V}_{M_1 N_1} = G^{-1} V_{M_2 N_1} G$. If

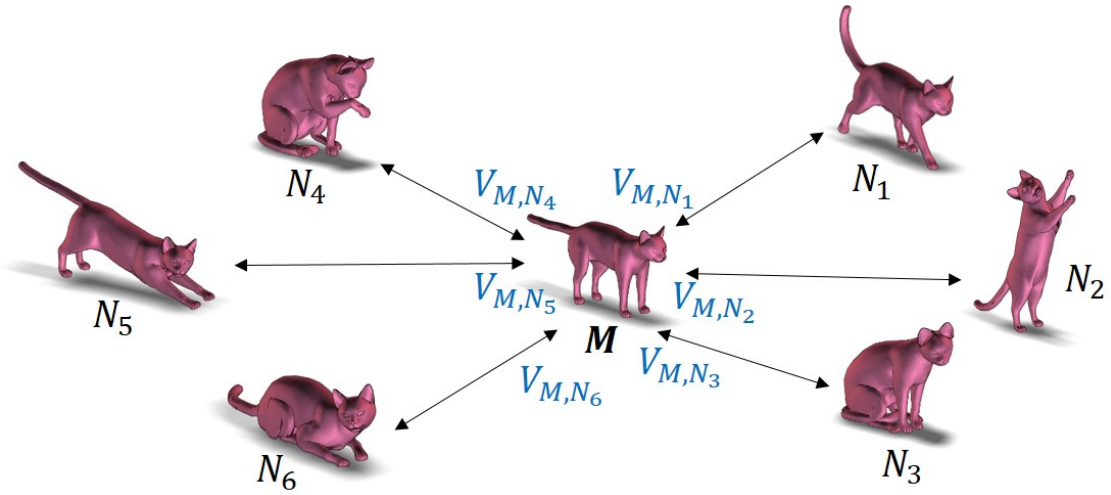

Figure 3.4: Given a map between two surfaces, $T : N \to M$, we obtain a map between function spaces $F : L^2(M) \to L^2(N)$ using $g = F(f) = f \circ T$. Given a choice of basis, it can be represented as a matrix in the discrete setting.

additionally G is orthogonal (namely, the map is volume preserving), then we have:

$$V_{M_1N_1} - V_{M_1N_2} = \tilde{V}_{M_2N_1} - \tilde{V}_{M_2N_2} \tag{3.3}$$

hence the distance between the shape differences as viewed on M_1 are equivalent to those viewed on M_2. The further G is from being orthogonal, the more influence the choice of base shape will have on the resulting distances, which can potentially be harmful for our registration process.

Therefore, there are two practical problems. First, since we only use the first k eigenvectors of the Laplace-Beltrami operator, we lose cycle consistency, and Equation 3.3 does not hold anymore. Second, the distances are "distorted" by G, and therefore there is a dependence on the choice of base shape in the two collections. For example, if G has a non-trivial kernel, e.g. if there exists a part on M_1 which does not exist on M_2, then there is loss of information when changing base shapes. Effectively, the difference between shapes which differ at the missing part cannot be represented using the base shape M_2.

Therefore, we would like to choose, in both collections, base shapes on which the differences between all the shapes are well represented. To do that, we search for a shape M such that F_{MN} for all the shapes N in the collection is close to an orthogonal matrix. We define this concept as the *shape irregularity*. Specifically, we compute:

$$\arg\min_{M \in A} \sum_{N \in A, N \neq M} \sum_{i=1}^{k} |\sigma_i - 1|^2 \tag{3.4}$$

where σ_i is the i-th singular value of the functional map F_{MN} from M to N, and k is

the number of basis vectors we are using for the representation.

Figure 3.5 demonstrates the effect the choice of base shape has on our algorithm as described in Sections 4 and 5. When the worst shapes (b, d) according to the shape irregularity measure are chosen as base shapes, the resulting approximated functional map is not satisfactory (e), as can be seen from the density of the matrix, the errors in transferring a smooth function between the source and target shapes, and the alignment results (only 18% of the shapes were paired correctly). For a good choice of base shapes (a,c), the resulting alignment is 90%, and the functional map is close to the ground truth (f). For a choice of base shapes which is non-optimal, we still get a reasonable, yet sub-optimal, functional map (g). Hence, while our algorithm is dependent on the choice of base shapes, this is done automatically in a manner which optimizes the resulting functional map between the collections. Furthermore, the result is stable under a choice of sub-optimal base shape.

After choosing the base shapes in the collections, we compute the shape difference representation for every shape, and compute the intrinsic distances between the shapes using Equation (3.2).

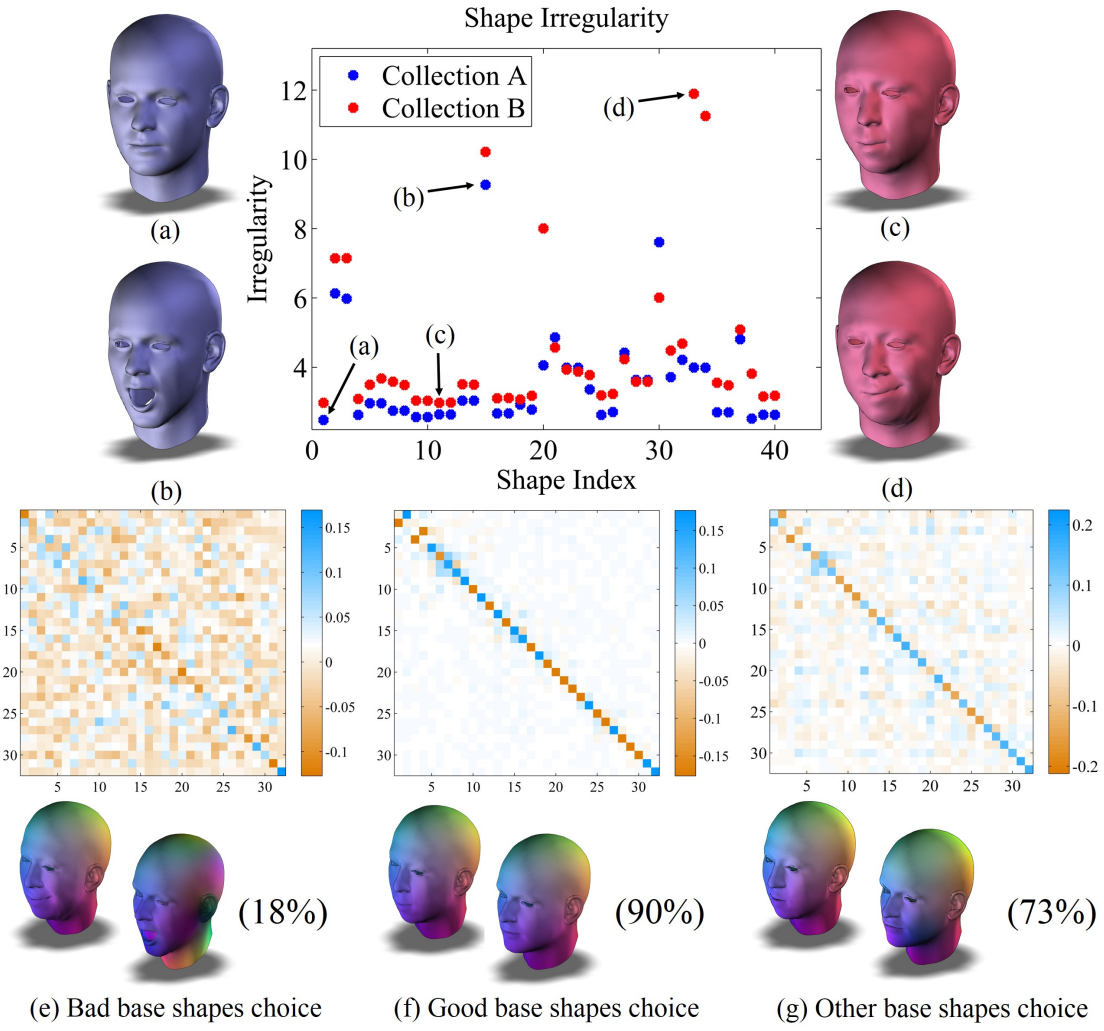

Figure 3.5: Shape irregularities in two collections, the chosen base shapes, the alignment results (in percentages) and the approximated map. (a, b) Best and worst base shapes in collection A. (c, d) Best and worst base shapes in collection B. (e) A bad pair of base shapes yields a bad map, as can be seen by pushing a coordinate function. (f) Choosing a good pair results in a good map. (g) Choosing non-optimal base shapes (#19 in A and #40 in B) yields sub-optimal, yet reasonable, results.

Chapter 4

Collection Alignment

After obtaining the shape differences, we would like to find a correspondence between the two collections. However, the shape differences in different collections cannot be compared directly since different base shapes are used. We therefore assume that each shape collection is a point sampling from a low-dimensional shape space, and use the intrinsic shape difference distances to embed this point cloud in Euclidean space. We then align the resulting point clouds.

4.1 Diffusion Maps

The "diffusion maps" algorithm [6] is a widely known method for non-linear dimensionality reduction which has been used in many diverse fields, such as computer vision, medical imaging and shape analysis. It has also been used for the analysis of shape collections [25].

In diffusion maps, we first construct a symmetric weighted graph where each node corresponds to a data point. The weights of the edges represent the similarities between the data points. In our setting, these weights are determined according to the SDD between the shapes, as defined in Equation (3.2). Then, we calculate the diffusion matrix by normalizing the rows of the matrix of the graph. Taking powers of the diffusion matrix allows us to observe the data at different scales and see the global connectivity of the data set.

We mark our data set by X and its dimension by n. We first construct a symmetric weighted graph where each node x_i corresponds to a data point. We use the Gaussian kernel: $k(x_i, x_j) = exp(-\|x_i - x_j\|^2/(2\sigma^2))$, where σ is a user-defined parameter. This function is called the *diffusion kernel*. It is symmetric: $w(x_i, x_j) = w(x_j, x_i)$; and non-negative: $w(x_i, x_i) \geq 0$ for all x_i, x_j. We denote the *kernel matrix* by K such that: $K_{ij} = k(x_i, x_j)$.

We now have a symmetric matrix where each row and column corresponds to a data point. Then, we calculate the row-normalized *diffusion matrix* P, with entries $P_{ij} = p(x_i, x_j)$, by: $P = D^{-1}K$, where D is the diagonal matrix consisting of the row

sums of K. If we think of a random walk, then this matrix contains the probability of jumping from i to j in a single step.

Calculating the probabilities P^t for increasing values of t enables us to observe the data set at different scales. This process is often called the *diffusion process*, where we can see the global connectivity of the data set. With increased values of t, the probability of following a path along the intrinsic structure of the data increases.

Next, we define a diffusion metric based diffusion matrix. It is related to the diffusion matrix P and is given by:

$$D_t(x_i, x_j)^2 = \sum_{u \in X} |p_t(x_i, u) - p_t(u, x_j)|^2 = \sum_k |P^t_{ik} - P^t_{kj}|^2$$

Finally, we map points in the data set into an Euclidean space according to the diffusion metric. Such a map is called a *diffusion map*. After this mapping, the diffusion distance in the data space becomes the Euclidean distance in this new *diffusion space*, denoted by Y. Since this map maps points into an Euclidean space derived from the geometric structure of the data set, we expect that its dimension will be smaller than the original dimension of the data set. As proven in [6], the diffusion distances can be expressed using the following map:

$$y_i^d = \begin{pmatrix} \lambda_1^t \psi_1(i) \\ \lambda_2^t \psi_2(i) \\ \vdots \\ \lambda_d^t \psi_d(i) \end{pmatrix}$$

Where $\psi_1(i)$ indicates the i-th element of the first eigenvector of P, etc. For this map, the Euclidean distance between y_i^d and y_j^d is the diffusion distance between the original data points x_i and x_j. We choose d such that $d \ll n$ and thus we achieve the dimensionality reduction.

4.2 Symmetries in the Diffusion Space

A core trait of the Diffusion Maps method, is that its output is not immune to symmetries. Namely, if we take two similar collections A and B and reduce their dimensionality using diffusion maps, we will receive two low-dimensional point clouds which are similar up to reflections along the axes in the diffusion space. We approach this challenge in Section 4.5.

Figure 4.1 shows the energy in each dimension after embedding a shape collection using diffusion maps and using a linear method (PCA). While PCA finds a hyperplane in the embedding space, which is a linear embedding, diffusion maps finds a hyper surface which is not necessarily linear. A shape collection is usually embedded on a nonlinear manifold. Therefore, a nonlinear method such as diffusion maps is able to recover its

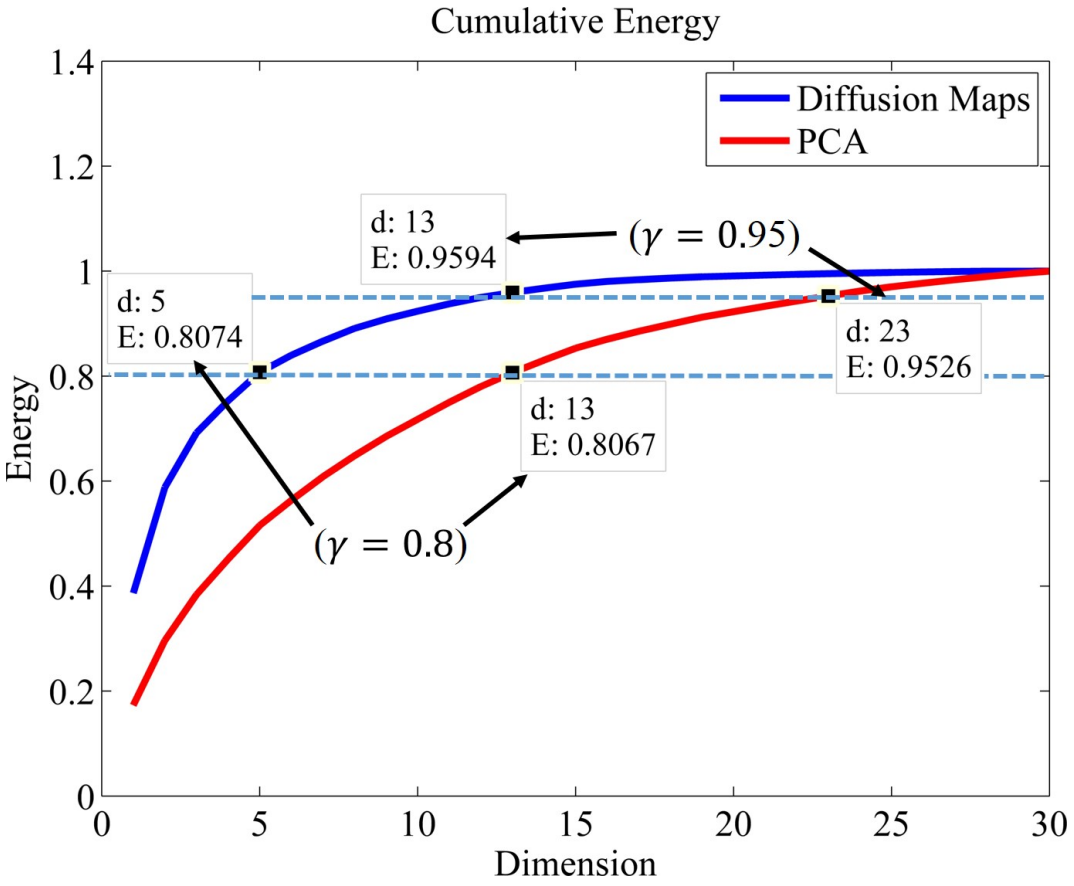

Figure 4.1: Dimensionality reduction allows us to reveal the intrinsic dimension of a shape collection. Here we can see the cumulative energy (E in Equation 4.1) of a collection of 40 blend shapes when applying PCA or diffusion maps and the estimated dimension of the data. Choosing a higher value for γ would yield a higher estimated dimension. Since diffusion maps is a non-linear technique, it is capable of recovering the true, non-linear structure of the data (9-dimensional). PCA, on the other hand, assumes a linear structure and therefore identifies an higher intrinsic dimension of 19.

true dimension more accurately than a linear method.

4.3 Coherent Point Drift

When aligning two point clouds we need to assume some prior on the allowed transformations between them. In general, since our sampling is relatively sparse compared to the dimension (e.g. 40 shapes in dimension 9), we need to assume a somewhat restrictive prior to avoid over-fitting. Assuming the transformation between the point clouds is rigid (i.e. rotation and translation) is too restrictive, as is uniform scaling. Allowing an affine map between the point clouds allows the algorithm to tolerate some error in the SDDs between the shapes (e.g. because the collections are not exactly aligned, or the

choice of base shape is not optimal), while still avoiding over-fitting. In addition, we allow reflection, as the diffusion map embedding is only defined up to isometries. We use "coherent point drift" (CPD) for the alignment, which is a state-of-the-art registration algorithm that supports affine registration.

The CPD algorithm enables both rigid and non-rigid registration of two point clouds. The registration is not symmetric, namely, cloud B is registered to cloud A or vice-versa, but not both. We have chosen the CPD algorithm for several reasons. First, unlike other methods, CPD is specifically capable of handling d-dimensional clouds, where $d > 3$. Our dimensionality reduction usually outputs a cloud which is not 3-dimensional (can be 10-dimensional, for example). Second, it is suitable for both affine and non-rigid registration. In our setting, the points in the embedding domain are subject to affine transformations with reflections, caused by the transformation of the shape differences and the dimensionality reduction.

The main idea of CPD is as follows. Given two point sets, a Gaussian Mixture Model (GMM) is fitted to the first point set, whose Gaussian centroids are initialized from the points in the second set. Then, a process in which the Guassian centroids move from the initial position to their final position is considered. In order to keep the structure of the point set, a *motion coherence* constraint is imposed over the velocity field.

Fig. 4.2 (right) shows the cloud of collection A after aligning it to the cloud of collection B using the resulting affine transformation.

4.4 Intrinsic Dimension Estimation

In many cases, we do not know the intrinsic dimension d in advance. In such cases, we can estimate d from the data. We use a method similar to the one proposed in the VDM algorithm [21]. We set a threshold γ between 0 and 1. Then, we sum the energy along increasing dimensions until the ratio to the total energy exceeds the chosen threshold. Namely, we choose the minimal d such that:

$$E(d) = \frac{\sum_{u \in Y^d} \|u\|}{\sum_{u \in Y^n} \|u\|} > \gamma \tag{4.1}$$

where Y^d and Y^n are the result of reducing Y to d or n dimensions (no reduction), respectively. For example, $\gamma = 0.9$ indicates that the chosen dimension consists at least 90% of the total energy of the data set. In our setting, given two shape collections A and B with estimated intrinsic dimensions d_A and d_B, we choose $d = \max\{d_A, d_B\}$. This way, we do not lose information about either collections.

Figure 4.1 demonstrates the effect of γ on the resulting estimated dimension, as well as the advantage of a non-linear dimensionality reduction technique over a linear one (PCA), using the discussed dimension estimation technique.

4.5 Shape Pairing

An important observation is that CPD is an asymmetric registration method. Namely, cloud B is registered to cloud A or vice-versa. However, in our setting, we do wish for a symmetric registration. Therefore, we perform the registration in the following way: first, we do not allow a point in the source cloud to match more than one point in the target cloud. If a source point matches more than one target point, we choose the target point which is closer as the match. Second, we match both A to B *and* B to A, and then choose the direction which yields more matching points. Finally, a point which does not match any other point after the described process is considered to be an *outlier*. A result of this symmetric alignment is presented in Figure 4.4.

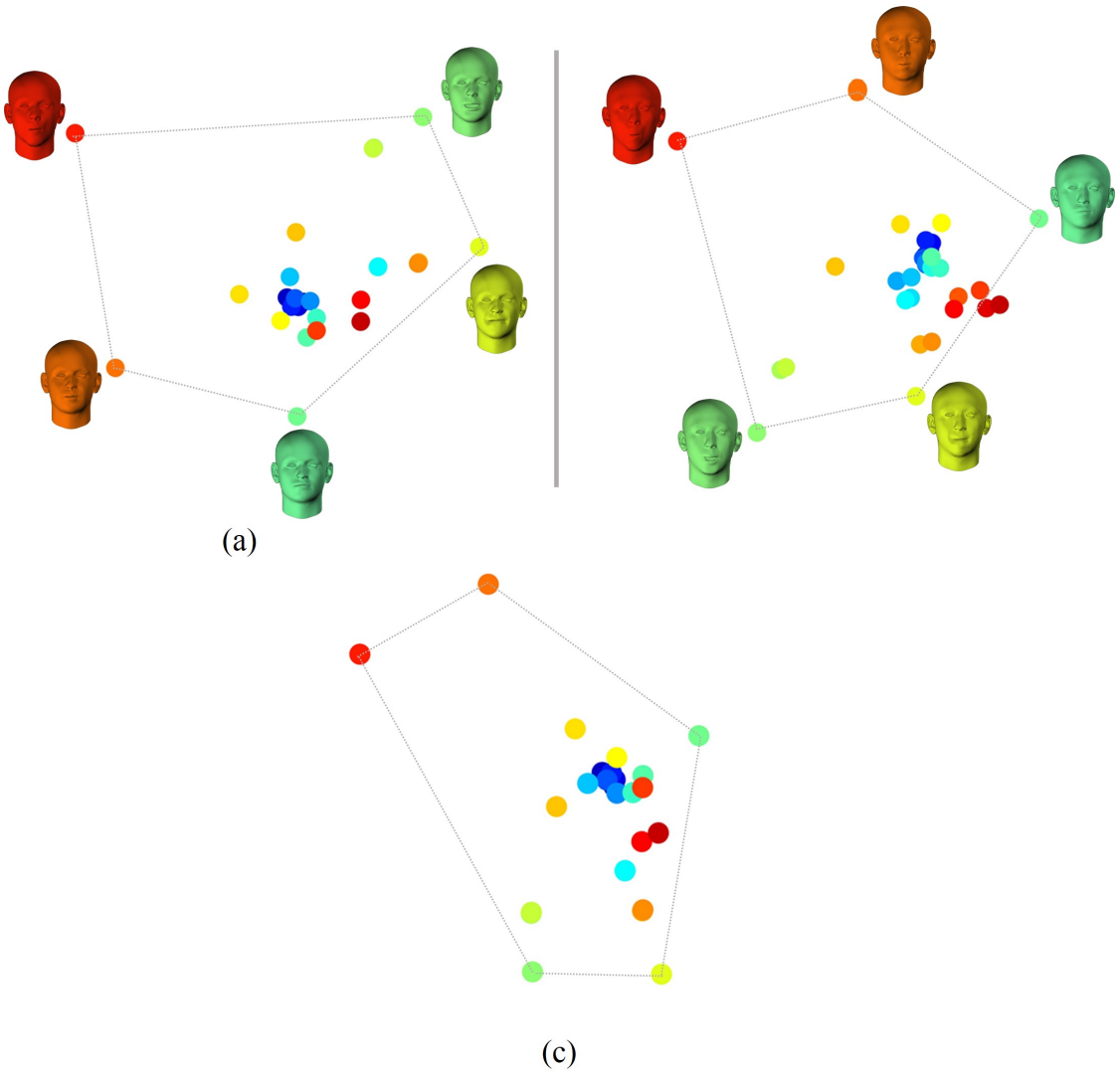

Figure 4.2: Two collections of 40 blend shapes after reducing their dimensionality using diffusion maps and projecting the resulting 9-dimensional cloud into 2D: (a) collection A cloud, (b) collection B cloud, (c) A to B alignment using an affine transformation with reflection. Even though additional energy is contained in higher dimensions, some of the similarities can be seen in 2D, such as the corresponding shapes (in matching colors) along the edges of the marked polygon.

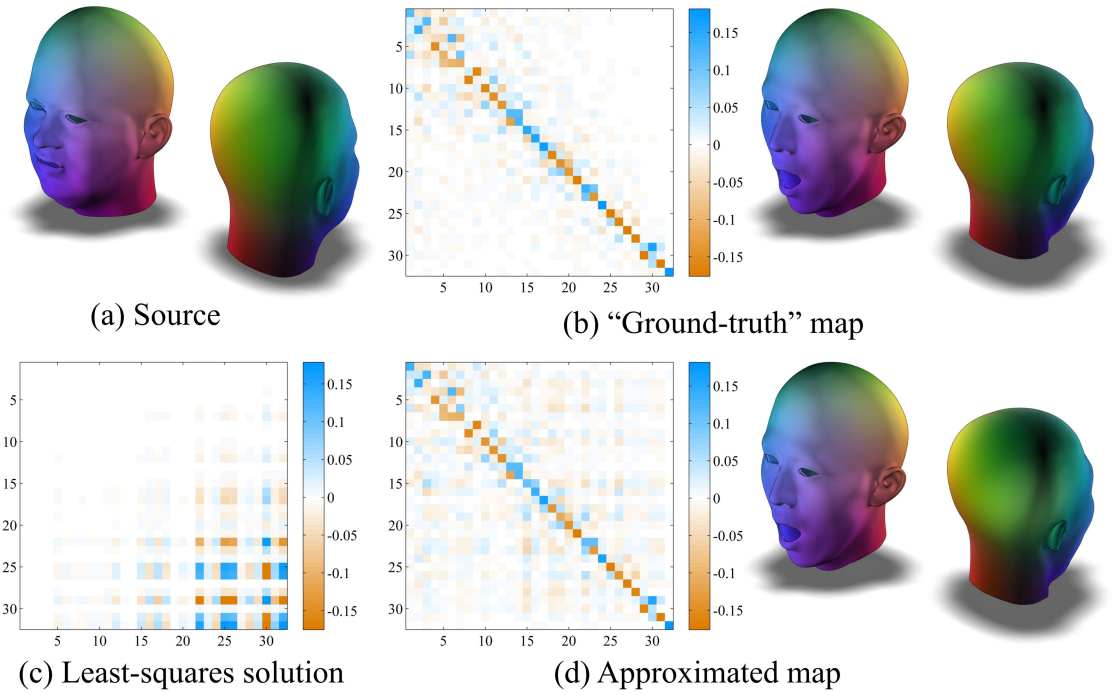

Figure 4.3: Functional map approximation: (a) source shape in collection A (b) "ground-truth" map to the target shape in collection B, used for comparison and computed from a manually created point-to-point map, (c) least-squares solution G to the map between the base shapes, (d) approximated functional map using iterative refinement and map composition as described in Section 5.2.

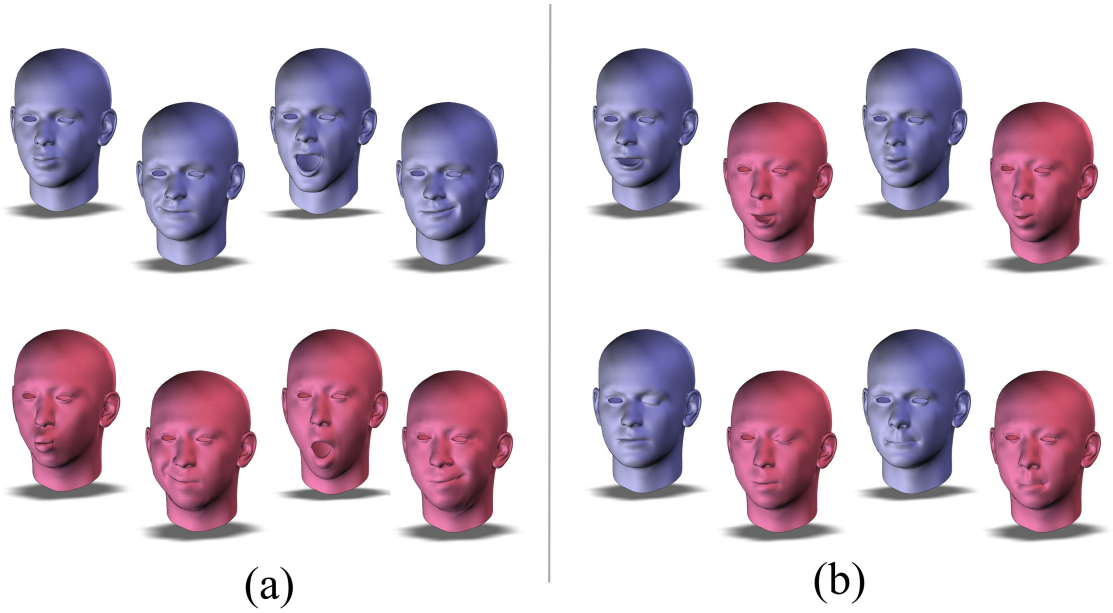

Figure 4.4: Aligning two collections of 40 blend shapes: (a) identified outliers, (b) correct matches. See also Figure 6.2.

Chapter 5

Functional Map Inference

5.1 Shape Analogies Constraints

So far we have used the shape differences for computing distances between shapes within the same collection. However, shape differences encode more information, which can be leveraged for computing a functional map *between the collections*.

Specifically, if we know that two shapes $M_A, N_A \in A$ correspond to two shapes $M_B, N_B \in B$, and we assume that the collections have similar structure, we can additionally assume that the shape differences correspond. Namely that $V_{M_A N_A}$ is similar to $V_{M_B N_B}$, and similarly for R. In the previous section we computed a pairing between shapes in both collections, hence, given such pairs we can pose constraints which enforce this similarity.

Specifically, let $M_i \in A$ and $N_i \in B$ be such that (M_i, N_i) are a corresponding shape pair. Further, let M_A be the base shape on A, and N_A its corresponding shape on in B. Finally let G be the unknown functional map between M_A and N_A. Since we cannot compare $V_{M_A M_i}$ with $V_{N_A N_i}$ directly as they are defined on different function spaces, we apply G on the left and on the right such that all operators take functions on M_A and return functions on N_A. This is demonstrated in Figure 5.1. This leads to the following equations:

$$\begin{aligned} \|GV_{M_A M_i} - V_{N_A N_i} G\|_F = 0, \\ \|GR_{M_A M_i} - R_{N_A N_i} G\|_F = 0. \end{aligned} \tag{5.1}$$

This leads to the following energy:

$$\arg\min_G \sum_{i=1}^{K} \left(\|GV_{M_A M_i} - V_{N_A N_i} G\|_F^2 + \|GR_{M_A M_i} - R_{N_A N_i} G\|_F^2 \right) \tag{5.2}$$

where K is the number of matching pairs. In order to minimize this energy, we solve a set of equations which are linear in the elements of G. This is a homogeneous problem and thus it can be solved using SVD. Intuitively, these constraints enforce *shape analogies*, namely, M_A is to M_i as N_A is to N_i. Note, that in [20] similar constraints were used for

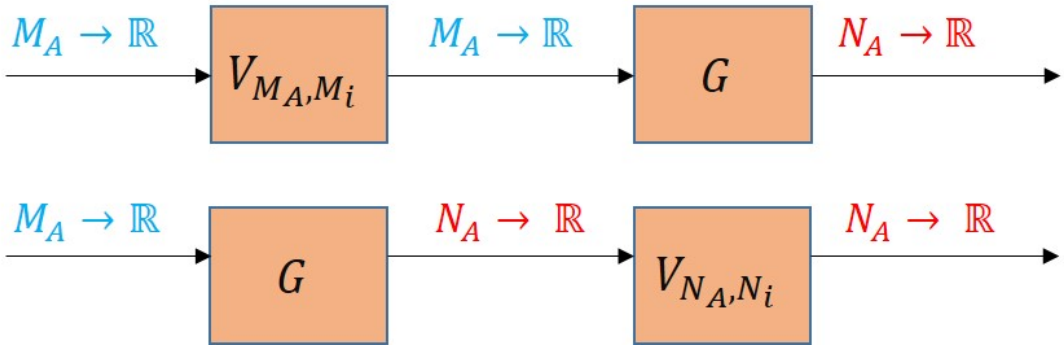

Figure 5.1: When posing the shape analogies constraints, the internal shape differences are operators which are defined on different domains. Therefore, we apply G on both sides in order to get operators which are defined on the same domain and thus are comparable.

finding corresponding shapes given the map G, whereas we solve for the map given the corresponding shapes. Note that these constraints are completely automatic, as the only input they require is the shape pairing between M_i and N_i and between M_A and N_A.

5.2 Iterative Refinement

In general, the matrix G which minimizes the energy in Equation (5.2) does not correspond to a bijection, as we did not enforce any additional constraints beyond the shape analogies. However, we can proceed using a post-processing iterative refinement algorithm, as proposed in [19], used to refine a given matrix to make it closer to a point-to-point map. We refer to G_0 as an initial estimate to G and denote the Laplacian eigenvectors matrices of A and B by φ_A and φ_B. As noted in [19], if $G_0 : M \to N$ is a functional map corresponding to a volume preserving map, then G_0 should be such that each column of $G_0\varphi^M$ coincides with some column of φ^N. We treat φ_A and φ_B as two point clouds with dimensionality equal to the number of eigenvalues which we used. In addition, for a volume preserving map we also expect the mapping matrix G_0 to be orthonormal, thus we can perform a rigid alignment between φ_A and φ_B by the following iterative algorithm:

1. For each column v of $G_0\varphi^M$ find its closest $\tilde{v}$ in φ^N.
2. Find the orthonormal G which minimizes $\sum \|Gv - \tilde{v}\|$.
3. Set $G_0 = G$ and iterate for a fixed number of iterations.

This algorithm is effectively ICP in eigenspace, using the minimizer of Equation (5.2) as the initial solution.

Using this method, we are able to reconstruct an approximated functional map. Note that since this is a homogeneous problem the solution will be up to a constant

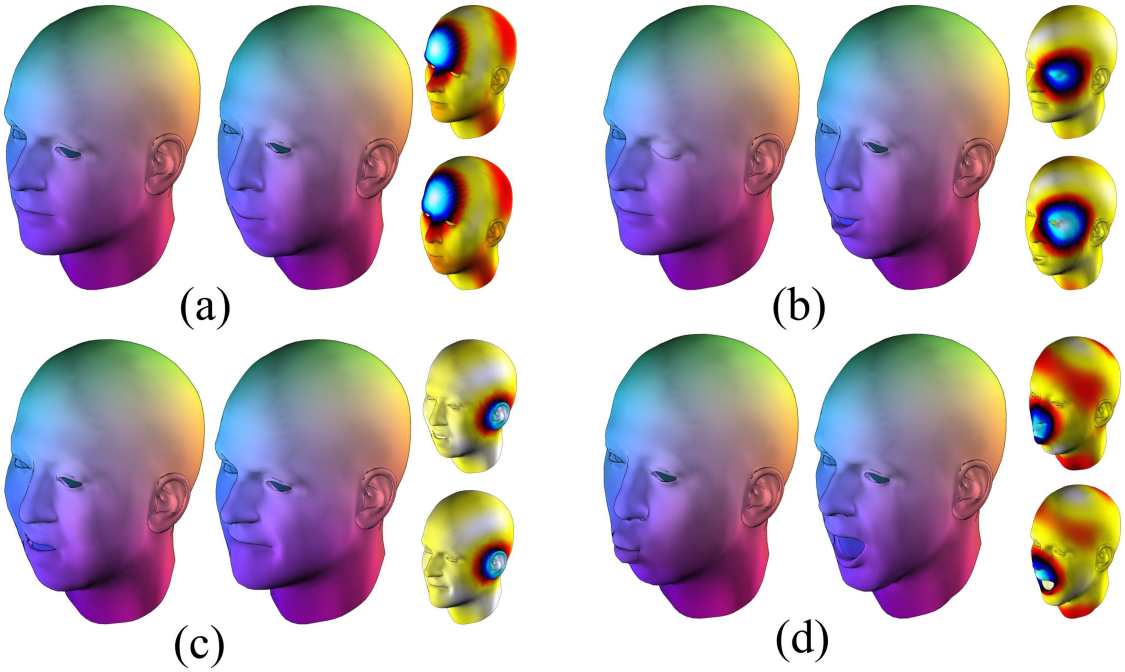

Figure 5.2: Pushing coordinates functions (left → right) and Gaussians (top → bottom) through approximated functional maps between collections A and B: (a) A base shape to B base shape, (b) A arbitrary shape to B arbitrary shape, (c) B arbitrary shape to A arbitrary shape, (d) B outlier to A outlier. Notice that a functional map between two outliers is approximated successfully.

multiplication (positive or negative). We can ignore the scaling factor – the functional map is a linear operator and we normalize every function pushed through it. However, the sign of this constant *does* affect the resulting G. Therefore, we apply iterative refinement separately for G_0 and $(-G_0)$ and choose the solution which minimizes the noted sum of distances.

We note again that G is a functional map between the *base shape* in A and its corresponding shape in B. In order to get the functional map between two arbitrary shapes $M_i \in A$ and $N_j \in B$, we compose the functional maps to the base shape M_A and to its corresponding shape N_A. We mark $F_{M_i M_A} : L^2(M_i) \to L^2(M_A)$ and $F_{N_A N_j} : L^2(N_A) \to L^2(N_j)$ and compose them with G:

$$F_{M_i N_j} = F_{N_A N_j} \cdot G \cdot F_{M_i M_A} \tag{5.3}$$

Note that using Equation (5.3) we can compute a functional map between *any* two shapes in the collections, including shapes which were considered outliers or were not matched during the registration step. Figure 4.3 demonstrates the process of approximating a map between two shapes using the algorithm described above.

In order to calculate the functional maps in the opposite direction (namely, from B to A), we simply produce the corresponding equations by swapping A and B, and proceed

as described above. Figure 5.2 shows approximated maps between various shapes in both collections. The maps are evaluated between shapes which belong to a matching pair, as well as between shapes which were classified as outliers or were a part of wrong match.

Chapter 6

Experimental Results

We tested our algorithm on different data sets. We present the results and compare them to "ground-truth" results:

- A known point-to-point map between the two collections, if such exists, is used to compute a "ground-truth" functional map for comparison purposes only. We compare our results to this map.

- If the correspondence between the two collections is known (for example, corresponding facial expressions), we demonstrate our registration results with respect to this known correspondence: correct matches, wrong matches, outliers (shapes which were not matched at all) and the corresponding percentages.

Our parameters setting was as follows. We used 32 eigenvalues of the Laplacian for the computation of functional maps and shape differences. The parameters for diffusion maps were $t = 1$, $\sigma = 1$ and $\gamma = 0.9$ for the intrinsic dimension estimation. For CPD we used $\omega = 0.1$ and default values for the other parameters, as described in [17].

6.1 Limitations

First, our algorithm assumes a similar structure in both collections – if the two given collections do not have a similar structure we will not be able to align them. Second, as explained in [20], the shape differences are based on externally supplied maps between shapes, and they therefore depend on the quality of these maps. Another requirement is for the collection to contain a minimum amount of shapes (e.g. at least 30). Given a smaller amount of shapes, the collection alignment is not feasible, since the point cloud is too sparse compared to its dimension. In addition, a small collection means that the number of terms in Equation (5.2) will be smaller, leading to a larger approximation error. Finally, the algorithm depends on several parameters which must be chosen in advance.

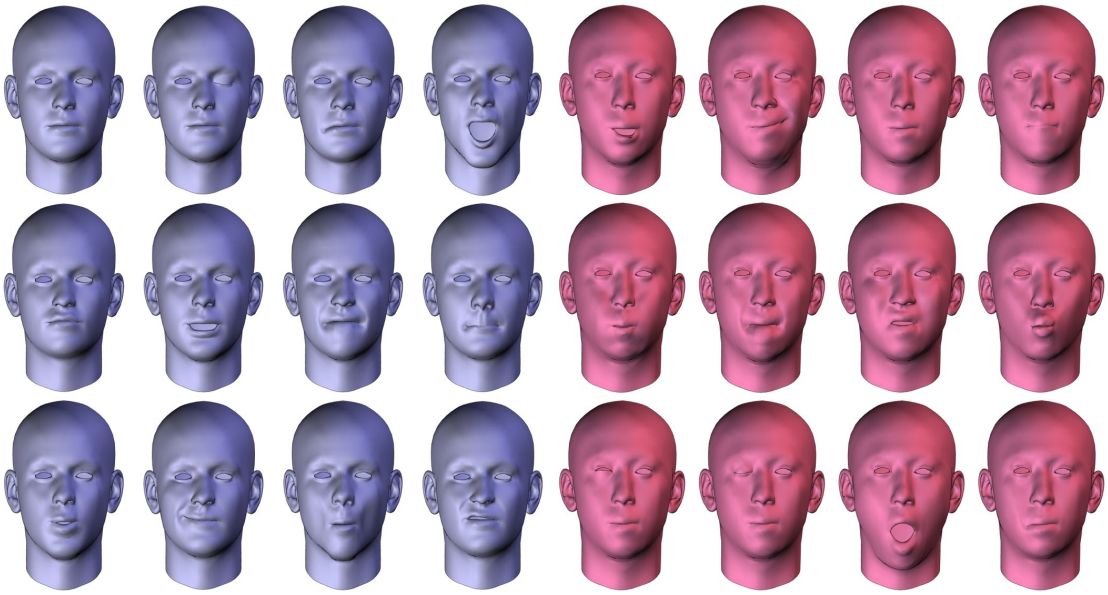

Figure 6.1: Two collections of 40 blend shapes which have been used in the experiments.

6.2 Blend Shape Collections

As presented throughout the paper, we tested our method on two collections of 40 blend shapes each (Figure 6.1).

- Diffusion maps produced two 9-dimensional point clouds.
- Registering the two collections resulted in 36 correct matching pairs (90%), no wrong matches and 4 outliers in each collection (10%). The results are shown in Figure 4.4.
- A functional map approximation was recovered and successfully extended to all the shape pairs, including the non-matching shapes in each collection. The results are presented in Figure 5.2.

6.3 FaceWarehouse Database

FaceWarehouse [3] is a database of 150 individual testers. Each collection consists of 47 different facial expressions and the collections are in correspondence. As mentioned before, we used this known correspondence only for comparison purposes. We tested our algorithm on four different pairs of testers. The results for these pairs are presented in Figures 6.3.

6.3.1 Testers Pair 1

The results are presented in Figure 6.2 and Figure 6.3. Diffusion maps produced two 11-dimensional point clouds. Our algorithm identified correctly 30 matching facial expressions (64%), 8 outliers (17%) and 9 pairs were wrong matches (19%). As can be seen in Figure 6.2, most pairs which were wrong matches are indeed similar. The functional map was approximated successfully. The shape irregularities of the two collections are very similar, which explains the successful map approximation.

6.3.2 Testers Pair 2

The results are presented in Figure 6.4. Diffusion maps produced two 11-dimensional point clouds. Our algorithm identified correctly 28 matching facial expressions (60%), 6 outliers (13%) and 13 pairs were wrong matches (27%). The functional map was approximated successfully. The shape irregularities of the two collections are similar, but less than *testers pair 1.* This yields in a lower matching percentage, but the map approximation is still successful in this case.

6.3.3 Testers Pair 3

The results are presented in Figure 6.5. Diffusion maps produced two 11-dimensional point clouds. Our algorithm identified correctly 17 matching facial expressions (36%), 18 outliers (38%) and 12 pairs were wrong matches (26%). The functional map was approximated successfully The shape irregularities of the two collections are similar, but less than *testers pair 1.* In this case, the matching percentage is low, but the map approximation is still successful.

6.3.4 Testers Pair 4

The results are presented in Figure 6.6. Diffusion maps produced two 13-dimensional point clouds. Our algorithm identified correctly 22 matching facial expressions (47%), 11 outliers (23%) and 14 pairs were wrong matches (30%). In this case, the shape irregularities graph shows that the two collections have quite a different structure. This fact leads to a functional map approximation with errors.

6.3.5 FaceWarehouse Summary

We discuss the performance of our algorithm according to the irregularity of its shapes as defined in Section 3.5. As we can see in the corresponding figures, when considering two collections after alignment, the shape irregularities graph provides a measure of the similarity between their structures. According to our experiments, we can see that correspondence in the shape irregularities graph predicts successful alignment and functional map approximation.

To summarize, we were able to align the collections of different testers with various percentage rates and functional map approximations were successful for all the tester pairs. Dissimilarities in the shape irregularities graph lead to a higher approximation error, as can be seen in the case of testers pair 4.

6.4 Varying Collection Size

This experiment is intended to measure the effect of the size of the collections on registration and map approximation. We used the blend shape collections presented throughout the article, but took only a subset of the shapes (the same subset in both collections). The alignments results were as follows:

- Using 20 shapes: 35% correct matches.
- Using 25 shapes: 36% correct matches.
- Using 35 shapes: 63% correct matches.
- Using 40 shapes: 90% correct matches.

The approximated maps corresponding to the size of the collections are presented in Figure 6.7. As we can see, using small collections leads to poor alignment, since the clouds are very sparse compared to their dimension. As we increase the number of shapes, alignment becomes feasible and thus the approximated map improves.

6.5 Small Collections With Perfect Alignment

This experiment is intended to test the approximated map in a rough setting when using small collections with perfect alignment (namely, we provided the matching shape pairs in advance). We used collections of 10 shapes from the Sumner and Popović database [22]. The resulting map was noisy but was still able to capture some of the information. The results are presented in Figure 6.8.

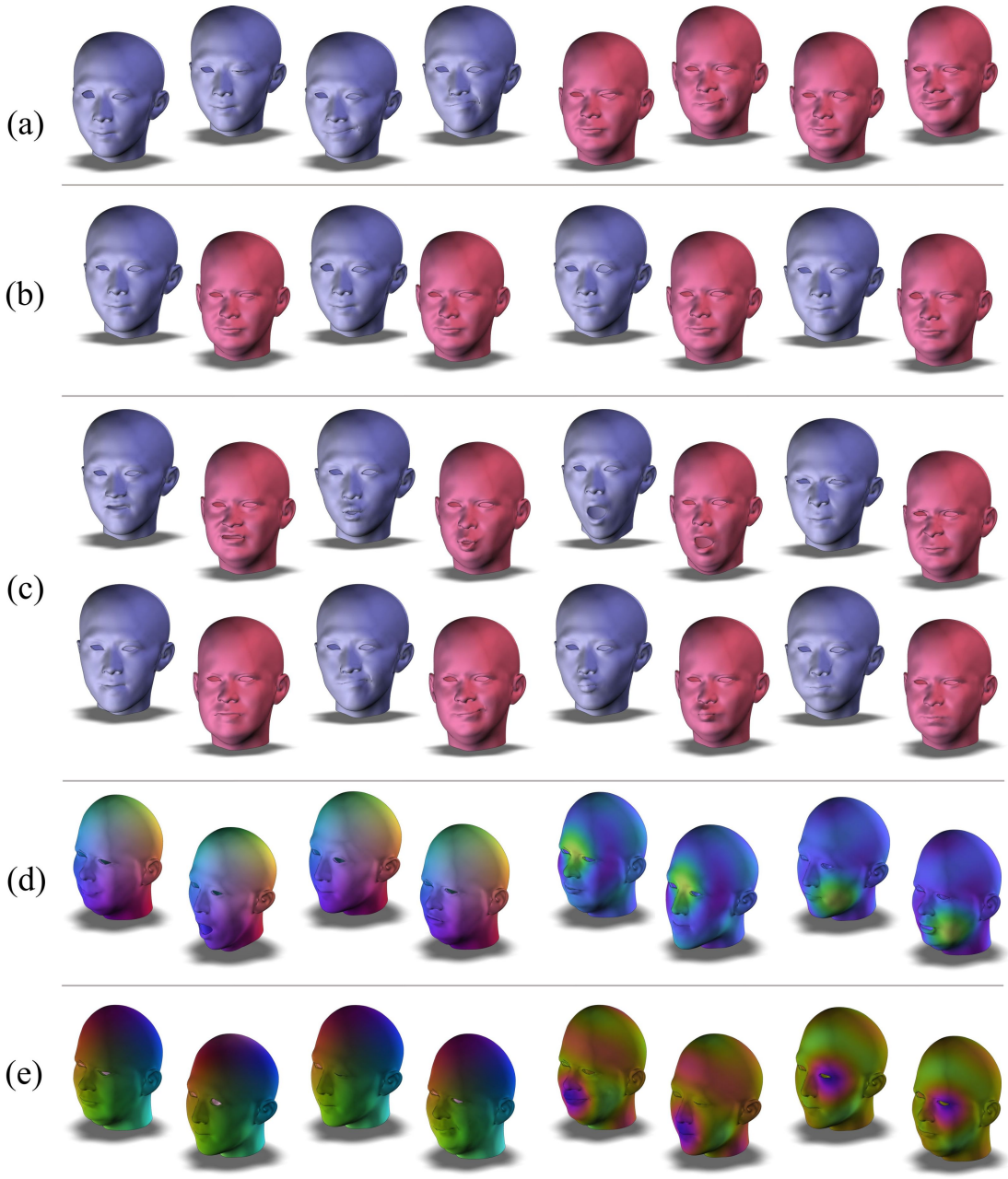

Figure 6.2: Correspondence and maps between two testers from the FaceWarehouse database [3]. Functional maps were approximated correctly for all shapes. (a) identified outliers (17%), (b) wrong matches (19%), (c) correct matches (64%), (d) maps between shapes which are part of a correct match, (e) maps between shapes which are outliers or part of a wrong match.

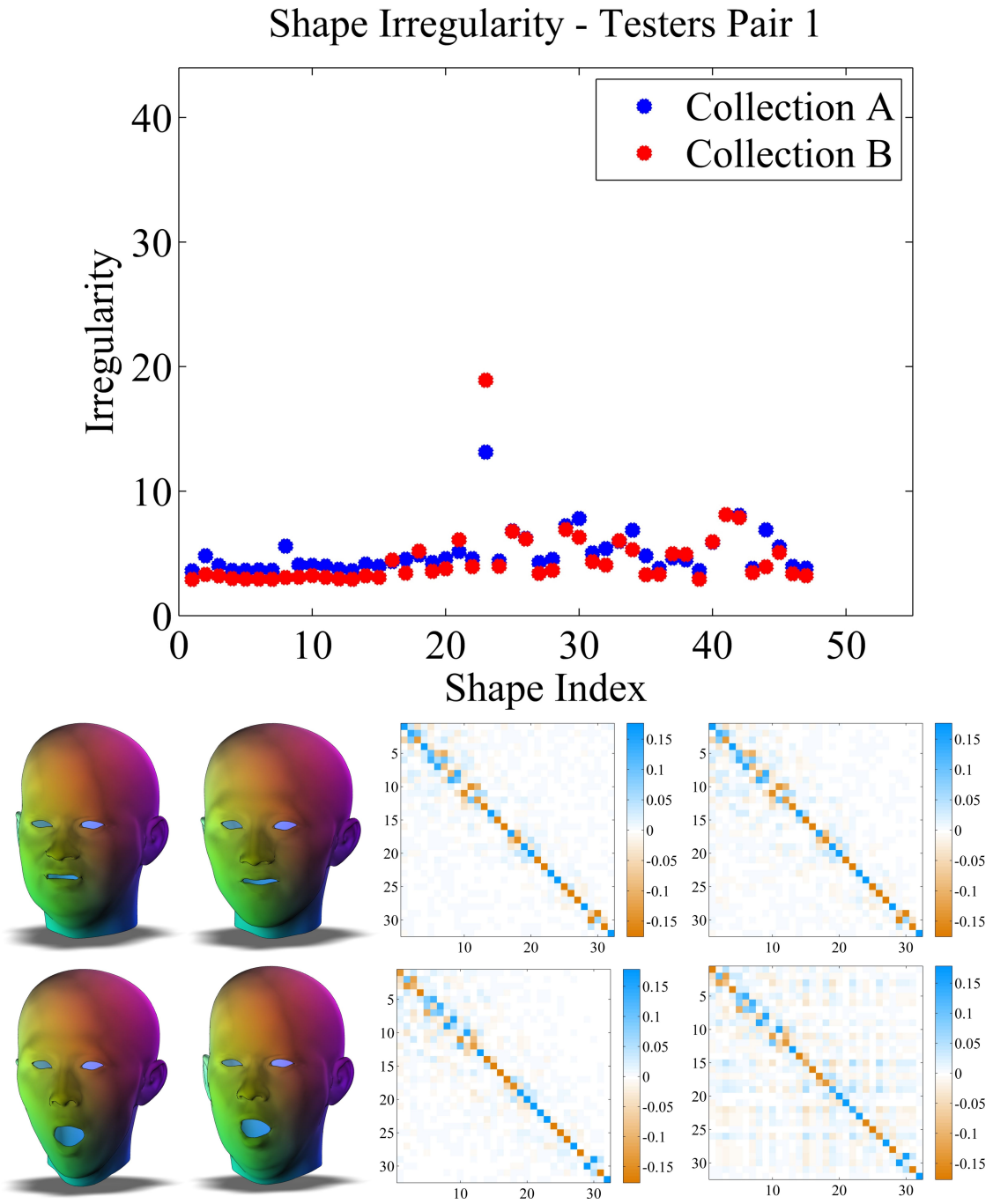

Figure 6.3: Map approximations between *testers pair 1* of the FaceWarehouse database. Two maps between arbitrary shapes are shown, the true functional map matrix (left) and the approximated one (right), and the shape irregularities when the two collections are aligned perfectly.

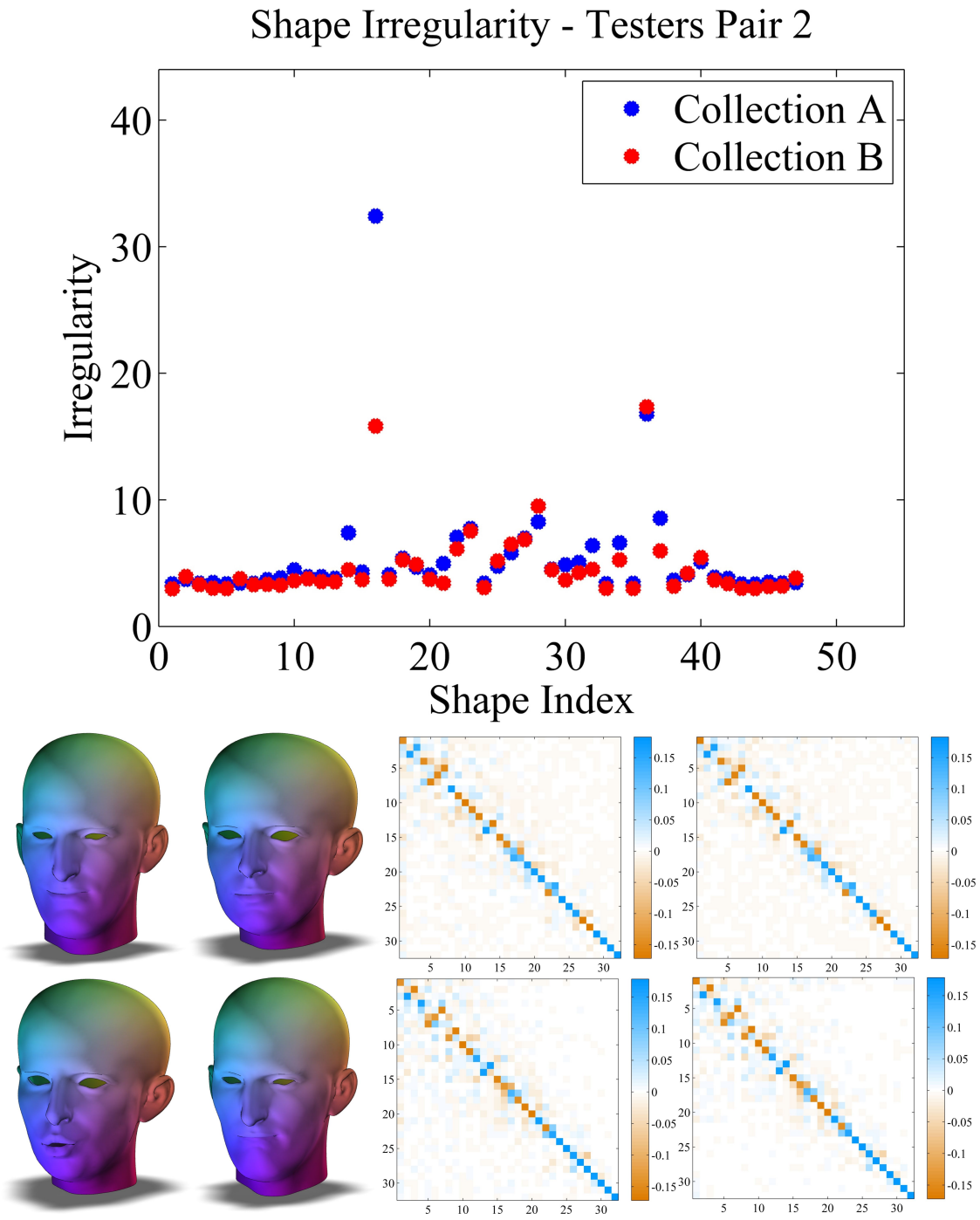

Figure 6.4: Map approximations between *testers pair 2* of the FaceWarehouse database. Two maps between arbitrary shapes are shown, the true functional map matrix (left) and the approximated one (right), and the shape irregularities when the two collections are aligned perfectly.

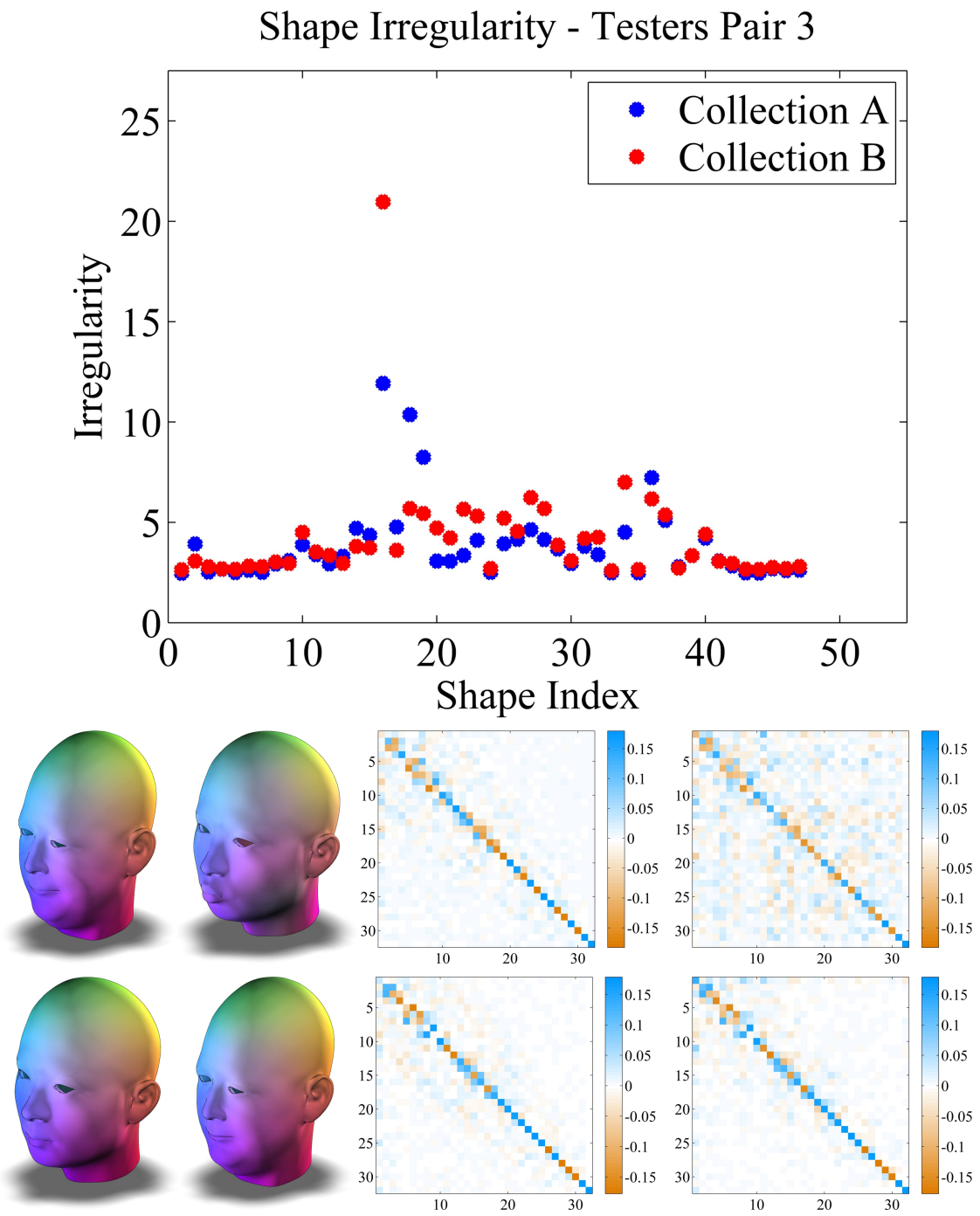

Figure 6.5: Map approximations between *testers pair 1* of the FaceWarehouse database. Two maps between arbitrary shapes are shown, the true functional map matrix (left) and the approximated one (right), and the shape irregularities when the two collections are aligned perfectly.

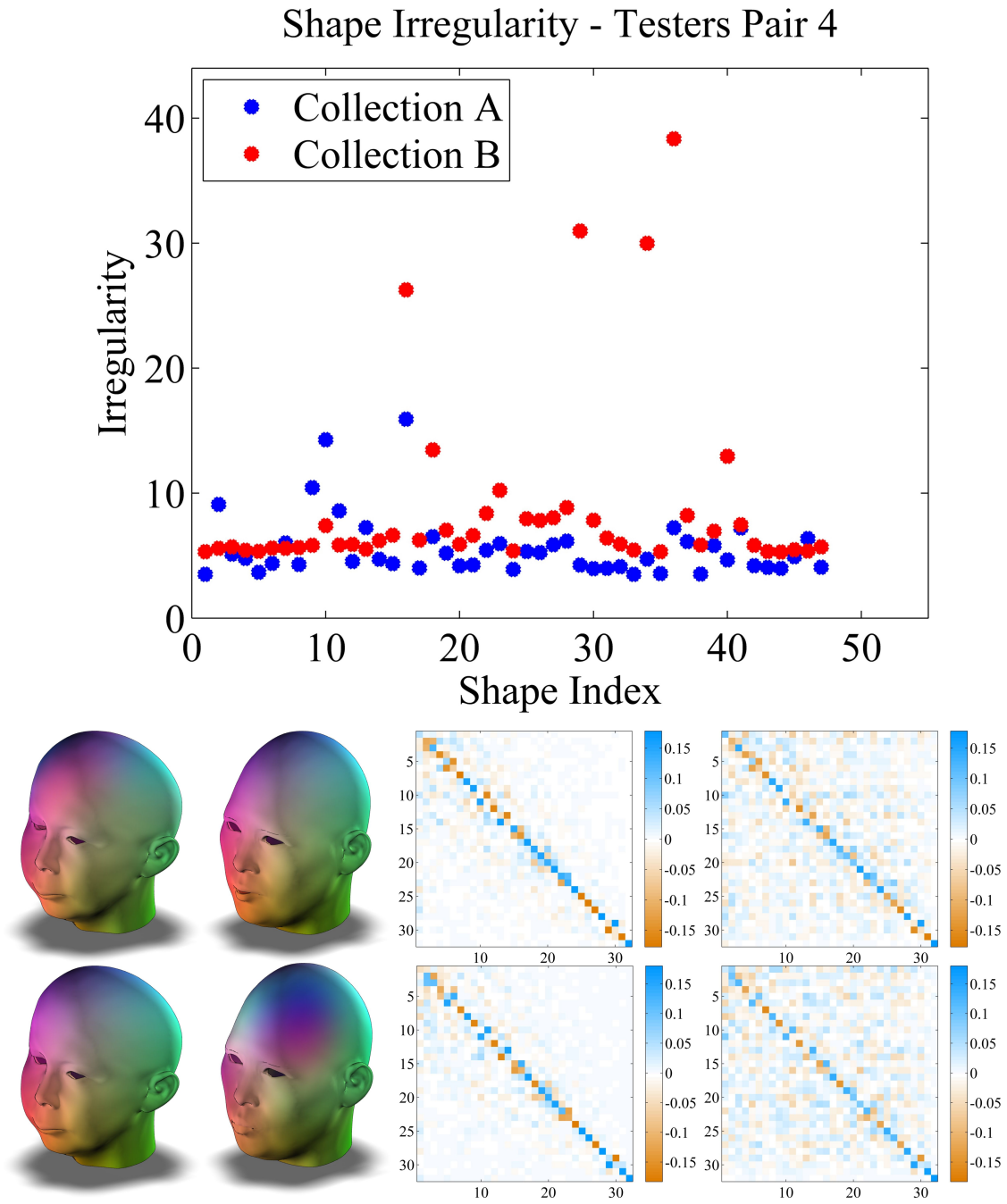

Figure 6.6: Map approximations between *testers pair 1* of the FaceWarehouse database. Two maps between arbitrary shapes are shown, the true functional map matrix (left) and the approximated one (right), and the shape irregularities when the two collections are aligned perfectly.

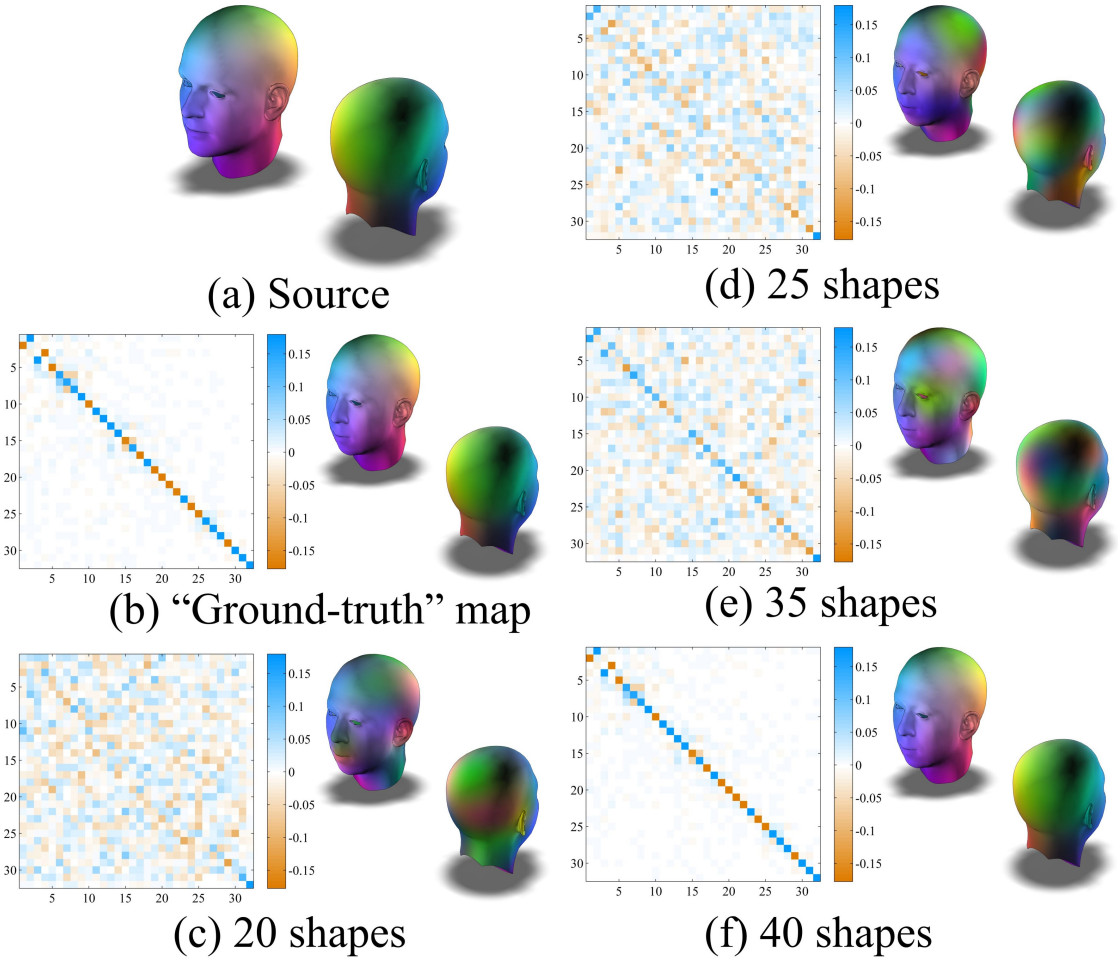

Figure 6.7: Alignment and map approximations for different subsets of the 40 blend shapes collections. We show the maps between the base shapes (collection A to collection B). (a) source shape in A, (b) "ground-truth" map, (c) using 20 shapes, 35% correct matches, (d) using 25 shapes, 36% correct matches, (e) using 35 shapes, 63% correct matches, (f) using 40 shapes, 90% correct matches.

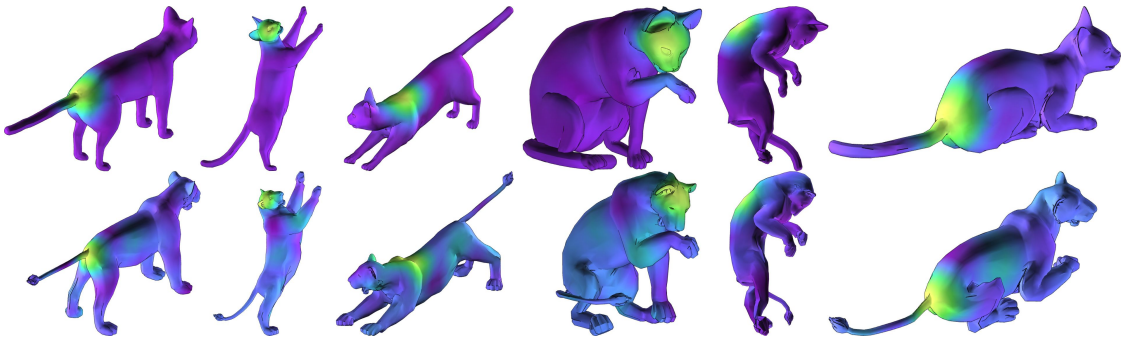

Figure 6.8: Functional map approximation for small collections (10 shapes) from the Sumner and Popović database [22]. Since the collections are small, registering them as point clouds is not feasible. However, given an optimal registration, a rough functional map can still be approximated. This functional map captures a certain amount of the data, but is noisy due to the small size of the collections.

Chapter 7

Conclusion and Future Work

We presented a novel approach for aligning two shape collections and approximating the functional cross-collection map, using only the maps within the collection as prior knowledge. We use shape differences to assess the distances between shapes intrinsically and generate a low-dimensional shape-space embedding. Then, we use affine registration in order to align the two point clouds. The shape differences framework is also used for posing shape analogies constraints for recovering the cross-collection functional map. We discussed the special cases in our method, such as the base shape selection, estimating the intrinsic dimension of the data and the significance of the size of the collections. We demonstrated the effectiveness of our algorithm on various collections and presented the success rate of the shape matching process as percentages of correct matches, as well as the approximated functional map, compared to a ground-truth map and presented on the shape themselves. Our method achieved smooth informative functional maps.

Our work provides a glimpse at the possibility of using existing shape analysis tools, such as dimensionality reduction and point registration, for analysing shape-space manifolds. The key to making the leap from shapes to shape spaces is having an intrinsic way to represent differences between shapes, which we achieved by using the shape difference linear operator. It is interesting to consider other functional operators for this task, as well as consider applying other common geometry processing tools directly to the shape-space manifold. An interesting future work will be to try our method on existing shape collections, and find a cross-map between them. Our method may also be used in order to find correspondences between shape collections which do not appear similar, but, in fact, do have a similar structure. Finally, as research progressed from analysing shapes in isolation to analysing collections of shapes, it is possible that the next layer of abstraction is analysing collections of collections. This can serve as a convenient way to model heterogeneous shape collections, simply as a collection of shape-spaces.

Bibliography

[1] Michal Aharon and Ron Kimmel. Representation analysis and synthesis of lip images using dimensionality reduction. *International Journal of Computer Vision*, 67(3):297–312, 2006.

[2] Silvia Biasotti, Simone Marini, Michela Spagnuolo, and Bianca Falcidieno. Sub-part correspondence by structural descriptors of 3d shapes. *Computer-Aided Design*, 38(9):1002–1019, 2006.

[3] Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. Facewarehouse: a 3d facial expression database for visual computing. 2013.

[4] Rodrigo L Carceroni, Flávio LC Pádua, Geraldo AMR Santos, and Kiriakos N Kutulakos. Linear sequence-to-sequence alignment. In *Proc. Conf. Computer Vision and Pattern Recognition.*, volume 1, pages I–746. IEEE, 2004.

[5] Yaron Caspi and Michal Irani. A step towards sequence-to-sequence alignment. In *Proc. Conf. Computer Vision and Pattern Recognition.*, volume 2, pages 682–689. IEEE, 2000.

[6] Ronald R Coifman and Stéphane Lafon. Diffusion maps. *Applied and computational harmonic analysis*, 21(1):5–30, 2006.

[7] Asi Elad and Ron Kimmel. On bending invariant signatures for surfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1285–1295, 2003.

[8] Patrick Etyngier, Renaud Keriven, and Florent Ségonne. Projection onto a shape manifold for image segmentation with prior. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 4, pages IV–361. IEEE, 2007.

[9] Patrick Etyngier, Florent Segonne, and Renaud Keriven. Shape priors using manifold learning techniques. In *Proc. Intl. Conf on Computer Vision*, pages 1–8. IEEE, 2007.

[10] Qi-Xing Huang and Leonidas Guibas. Consistent shape maps via semidefinite programming. In *Computer Graphics Forum*, volume 32, pages 177–186. Wiley Online Library, 2013.

[11] Qi-xing Huang, Fan Wang, and Guibass Leonidas. Functional map networks for analyzing and browsing large shape collections. In *ACM SIGGRAPH, to appear*, 2014.

[12] Qi-Xing Huang, Guo-Xin Zhang, Lin Gao, Shi-Min Hu, Adrian Butscher, and Leonidas Guibas. An optimization approach for extracting and encoding consistent maps in a shape collection. *ACM Transactions on Graphics (TOG)*, 31(6):167, 2012.

[13] Qixing Huang, Vladlen Koltun, and Leonidas Guibas. Joint shape segmentation with linear programming. In *ACM Transactions on Graphics (TOG)*, volume 30, page 125. ACM, 2011.

[14] Vladimir G Kim, Wilmot Li, Niloy J Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning part-based templates from large collections of 3d shapes. *ACM Transactions on Graphics (TOG)*, 32(4):70, 2013.

[15] Vladimir G Kim, Yaron Lipman, and Thomas Funkhouser. Blended intrinsic maps. In *ACM Transactions on Graphics (TOG)*, volume 30, page 79. ACM, 2011.

[16] Ivan Laptev, Serge J Belongie, Patrick Perez, and Josh Wills. Periodic motion detection and segmentation via approximate sequence alignment. In *Proc. Intl. on Conf. Computer Vision.*, volume 1, pages 816–823. IEEE, 2005.

[17] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(12):2262–2275, 2010.

[18] 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.

[19] 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):30, 2012.

[20] 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.

[21] Amit Singer and H-T Wu. Vector diffusion maps and the connection laplacian. *Communications on pure and applied mathematics*, 65(8):1067–1144, 2012.

[22] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 399–405. ACM, 2004.

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

[24] Nicolas Thorstensen, Patrick Etyngier, Florent Segonne, and Renaud Keriven. Diffusion maps as a framework for shape modeling. *Computer Vision and Image Understanding*, 115(4):520–530, 2011.

[25] Nicolas Thorstensen, Florent Segonne, and Renaud Keriven. Pre-image as karcher mean using diffusion maps: Application to shape and image denoising. In *Scale Space and Variational Methods in Computer Vision*, pages 721–732. Springer, 2009.

[26] Youyi Zheng, Daniel Cohen-Or, Melinos Averkiou, and Niloy J Mitra. Recurring part arrangements in shape collections. In *Computer Graphics Forum*, volume 33, pages 115–124. Wiley Online Library, 2014.

ביטוי בעננים הנקודות שהתקבלו. האלגוריתם בו אנו משתמשים עבור ההתאמה הוא *coherent point drift*, אלגוריתם אשר מתאים בין היתר גם להתאמה אפינית.

לאחר מציאת ההתאמה בין עננים הנקודות, אנו משתמשים בהנחה שלשני האוספים קיים מבנה דומה, על-מנת להגדיר אילוצים מהצורה "צורה A היא עבור B כמו שצורה C היא עבור D", בתור אילוצים לינאריים על המיפוי הפונקציונלי בין שני האוספים. אילוצים אלה מספיקים על-מנת לשחזר את המיפוי הפונקציונלי מבלי לדרוש כלל מידע נוסף. לאחר הגדרת אילוצים אלה, על זוגות הצורות המתאימות בלבד, המיפוי מחושב בקלות על-ידי פתרון בעיה לינארית אשר לוקחת בחשבון את כלל הזוגות התואמים שהתקבלו מההתאמה. לאחר מכן, אנו מרחיבים את המיפוי המחושב לכל הצורות בשני האוספים על-ידי הרכבה של המיפויים הפונקציונליים שהתקבלו, כאשר במקרה זה מדובר בהרכבת פונקציות פשוטה.

לבסוף, אנו מדגימים את האלגוריתם שלנו על מגוון מקרי קלט אפשריים, הלקוחים ממספר מקורות שונים. עבור כל מקרה קלט, אנו מריצים את כל שלבי האלגוריתם ומציגים הן את ההתאמה בין האוספים כאחוזי הצלחה לעומת ההתאמה האמיתית, אשר ידועה לנו ואנו משתמשים בה רק לצורך ההשוואה, וכן את המיפוי הפונקציונלי המחושב בין צורות שונות בשני האוספים. גם את המיפוי אנו משווים למיפוי האמיתי, אם הוא ידוע – וגם אז, אנו משתמשים בו רק לצורך ההשוואה. אנו מראים גם דוגמאות קלט נוספות, כגון ניסויים על אוספים אשר גודלם משתנה מהרצה להרצה, והשוואת התוצאות בין ההרצות השונות לצורך הבנת ההשפעה של גודל האוסף על טיב האלגוריתם.

אלגוריתם לפתרון הבעיה. על-מנת למצוא צורות מתאימות בין שני אוספי צורות בעלי גיאומטריה שונה, עלינו לבחור ייצוג עבור הצורות אשר מאפשר לזהות מבנים דומים בתוך האוספים. כלומר, נרצה לבחור ייצוג אשר מתאים ליחסים כגון "צורה A היא עבור B, כמו שצורה C עבור עבור D", ויאפשר לנו למעשה להשוות *הפרשים של הפרשים* בין צורות. ייצוג כזה (*shape differences*) הוצג לאחרונה, כאשר השוני בין שתי צורות אשר ידוע לנו המיפוי ביניהן, ניתן לייצוג על-ידי *אופרטור לינארי* אשר פועל על פונקציות המוגדרות על אחת מן הצורות. ייצוג זה מתבסס על *functional maps*, שיטה אשר כבר הוצג השימוש בה במגוון הקשרים. השוואת שני אופרטורים כאלה מספק מדד אינפורמטיבי עבור השוני בין הצורות, כיוון שהוא מאפשר לדעת לא רק האם שתי צורות הן שונות זו מזו, אלא בנוסף, *היכן* הן שונות.

אתגר משמעותי אשר אליו התייחסנו בעבודה הוא נושא בחירת *צורת הבסיס*. על-מנת לאפשר ייצוג של צורות באמצעות *shape differences*, נדרש לבחור בתוך כל אחד מהאוספים צורת בסיס, אשר תשמש כנקודת ייחוס עבור כל שאר הצורות וההפרשים יחושבו ביחס אליה. באופן אידאלי, בחירה זו יכולה להיות שרירותית. מצב זה מתקבל כאשר קיים *cycle consistency* של המיפויים בתוך האוסף – כלומר, כאשר מעגלים של הרכבות מיפויים מובילים למיפוי היחידה, וכן כאשר ניתן להגיע מכל ייצוג *shape differences* של צורה אחת לשנייה באמצעות מטריצה אורתוגונלית. במקרים פרקטיים, זה אינו המצב . בנוסף, אנו משתמשים בייצוג דיסקרטי (הן של הצורות והן של המיפויים) ועובדה זו גם היא שונה מהמצב האידאלי. לכן, יש חשיבות לבחירת צורת הבסיס. אנו מציגים שיטה לחישוב מדד עבור כל צורה באוסף, לו קראנו *shape irregularity*, אשר מציין כמה צורה זו *אינה מתאימה* להיות צורת בסיס. מדד זה מתבסס על הדרישה שעבור צורה המהווה בחירה טובה עבור צורת הבסיס, המיפוי הפונקציונלי ממנה לצורות האחרות יהיה קרוב למטריצה אורתוגונלית. בהתבסס על מדד זה אנו בוחרים את הצורה המתאימה ביותר להיות צורת הבסיס בכל אחד מהאוספים, באופן בלתי תלוי באוסף האחר.

על-מנת שנוכל להשיג ייצוג אינפורמטיבי של המרחקים בין הצורות וכיוון שהאוספים הם בדרך כלל ממימד גבוה, נרצה להמיר את האוסף למימד נמוך יותר. לכן, אנו מניחים כי כל אחד מהאוספים התקבל על-ידי דגימה של יריעה ממימד נמוך. לאחר מכן, אנו משתמשים בשיטה סטנדרטית להורדת מימדים, *diffusion maps*, כדי להשיג שיכון נמוך-מימדים במרחב אוקלידי. ייצוג זה משמר בצורה המיטבית את המרחקים בין הצורות על יריעה זו. אנו חוזרים על תהליך זה עבור שני האוספים כדי להציג ייצוג במימד נמוך עבור שניהם. לאחר מציאת שיכונים אלה, המרחקים בין הצורות הם למעשה מרחקים אוקלידיים במרחב ממימד נמוך זה. בסופו של שלב זה בידינו שני ענני נקודות ממימד נמוך לעומת המימד המקורי.

לאחר חישוב שני ענני-נקודות אלה, נרצה למצוא את ההתאמה ביניהם. אנו מוצאים התאמה זו ע"י התאמה אפינית רגילה. הבחירה בטרנספורמציה אפינית היא משום שטרנספורמציה כזאת מאפשר לאלגוריתם ההתאמה לספוג שגיאה מסוימת במרחקים בין הצורות, שכן לא תמיד ניתן למצוא התאמה מושלמת בין ענני הנקודות. בנוסף, אנו מאפשרים שיקופים, שכן אלגוריתם הורדת המימדים בו אנו משתמשים, *diffusion maps*, אינו רגיש לשיקופים ולכן הם אינם יבואו לידי

תקציר

רקע. מחקר זה עוסק במציאת התאמה ומיפוי בין אוספים של צורות. מציאת מיפוי וניתוח אוספים של צורות הם נושאים בסיסיים בתחומי הגרפיקה והגיאומטריה, אשר בשימוש במגוון תחומים, כגון אנימציה, מידול גיאומטרי, סריקה תלת-מימדית וניתוח מידע רפואי, וקיימים שימושים רבים נוספים. מציאת מיפוי בין צורות מתייחס לבעיה של, בהינתן שתי צורות, מציאת ההתאמה ביניהן, בעוד שניתוח אוספים של צורות מתייחס לבעיה של, בהינתן אוסף של צורות, הסקת מסקנה כלשהי עליו או שיפור היבטים מסוימים שלו, כגון: שיפור מיפויים בתוך האוסף, מציאת התאמות בתוך האוסף ועוד.

כאשר רוצים לייצג התאמה בין שתי צורות, במקרים רבים התאמה זו מיוצגת ע"י מיפוי נקודה-לנקודה, אשר ממפה נקודה על הצורה הראשונה לנקודה על הצורה השנייה. מיפוי מסוג זה מתאים כאשר הצורות דומות, לדוגמא: תנוחות שונות של אותו אדם, או הבעות פנים שונות של אותו אדם. עבור מקרים מסובכים יותר, כגון תנוחה דומה של שני אנשים שונים, לא תמיד ברור כיצד ניתן להגדיר מיפוי נקודה-לנקודה כזה, שכן הצורות עשויות להיות שונות זו מזו. במקרים כאלה, קל יותר להגדיר מיפוי פונקציה-לפונקציה, אשר ממפה פונקציה על הצורה הראשונה לפונקציה על הצורה השנייה. מציאת *מיפוי פונקציונלי* כזה הוא ייצוג גמיש וכללי יותר של מיפוי, ומאפשר לנו לייצג מידע אשר לעיתים קשה לייצג באמצעות מיפוי נקודה-לנקודה.

במקרים רבים, צורות אינן מופיעות לבדן אלא הן חלק מאוסף כלשהו בעל *מבנה* מוגדר. לדוגמא, לסריקה בתדירות גבוהה של אנימציה בין שתי תנוחות של אדם כלשהו יש מבנה שונה מאשר אוסף של צורות שאינן קשורות זו לזו. מציאת מבנה פנימי זה, וסידור שני אוספים בעלי מבנה דומה זה מול זה, כלומר – מציאת התאמה בין האוספים, הוא משימה חשובה אשר יכולה לסייע במגוון תחומים, כגון ניתוח סטטיסטי של תמונות רפואיות. אם האוספים הם הומוגניים, כלומר, הצורות בתוך האוסף הן דומות זו לזו, אז לעיתים קרובות ניתן למצוא מיפוי בין צורות בתוך כל אחד מהאוספים, ולאחר מכן למנף מידע זה לצורך ניתוח של מבנה האוסף.

הצגת הבעיה. אנו מתייחסים לבעיה הבאה. בהינתן שני אוספי צורות הומוגניים והמיפויים בתוך כל אחד מהאוספים, אנו מעוניינים למצוא התאמה בין הצורות בשני האוספים (כלומר, עבור צורה באוסף A למצוא צורה מתאימה באוסף B), וכן למצוא מיפוי בין כל הצורות בשני האוספים. בעיה זו אמנם עלולה להיראות קשה יותר מהבעיה המקורית של מציאת מיפוי בין שתי צורות שרירותיות, אך אנו מראים שניתן למצוא את המבנה הפנימי של כל אחד מהאוספים ולייצגו בצורה יעילה, כך שכאשר מבנים אלה הם דומים, ניתן לסדר את שני האוספים ולהתאים צורות בין שני האוספים השונים. באופן זה, אנו יכולים לחשב את המיפוי רק בין צורות אשר דומות (מתאימות) זו לו, ומיפוי זה הוא קל יותר לחישוב מאשר המיפוי בין שתי צורות שרירותיות.

המחקר בוצע בהנחייתה של פרופסור מירלה בן-חן, בפקולטה למדעי המחשב.

תודות

אני מודה למנחה שלי, פרופ׳ מירלה בן-חן, על העבודה המשותפת המוצלחת.

אני מודה לטכניון על התמיכה הכספית הנדיבה.

מציאת מיפוי בין אוספים של צורות באמצעות שיטות להורדת מימדים

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת התואר

מגיסטר למדעים במדעי המחשב

ניצן שפירא

הוגש לסנט הטכניון – מכון טכנולוגי לישראל

תשרי התשע״ד חיפה ספטמבר 2014

מציאת מיפוי בין אוספים של צורות באמצעות שיטות להורדת מימדים

ניצן שפירא