NeuroMorph: Unsupervised Shape Interpolation and Correspondence in One Go

Askar Kolushev

Department of Informatics — Technische Universität München

Abstract

NeuroMorph: Unsupervised Shape Interpolation and Correspondence in One Go is a paper written by Marvin Eisenberger, David Novotny, Gael Kerchenbaum, Patrick Labatut, Natalia Neverova, Daniel Cremers and Andrea Vedaldi and published in 2021. In the paper, the authors present an unsupervised way to find corresponding points of two 3D shapes and interpolate their poses while preserving their identities. The method uses a neural network to extract feature descriptors for explicit mapping of corresponding points and to generate vectors of interpolation transformation. Testing the resultant performance using various metrics and datasets shows that NeuroMorph matches or outperforms pre-existing supervised, unsupervised and axiomatic methods at both point correspondence and shape interpolation.

1 Introduction

Constructing a *smooth interpolation* between two given 3D shapes is a classical task in 3D computer vision. Such shapes are usually referred to as *source* and *target*. In its traditional form, the challenge is to find a continuous deformation of the source that would eventually make it identical to the target. Continuity of this deformation supposes that an intermediate shape can be queried at any time point along the transformation with an arbitrary precision.

An advanced version of this task — and the main goal of NeuroMorph — is to interpolate the poses of the two shapes while keeping the identity of the source shape intact. In other words, by the end of the transformation the source should "mimic" the overall stance of the target, but not turn into the same shape completely. Instead, it has to preserve its local structure to be clearly recognizeable at any stage of the transformation. These constraints are captured, for example, by LIMP [1], whose loss function separates intrinsic and extrinsic parts of a latent code to disentangle identity (style) and pose of the object.

In order to produce a continuous interpolation, any method should provide *geometric* and *statistical characterization* of the shape space.

Geometric characterization is often realised by considering individual shapes as points in a high-dimensional space, with their distance to the neighbors corresponding to the difference in their structure. Following this picture, such a space will contain all technically possible 3D shapes. Any given set of particular samples can then be viewed as a low-dimensional manifold inside that space. A geodesic path across this manifold between any two shapes from the dataset will present a continuous transformation from the source to the target, while keeping all intermediate shapes geometrically plausible. This turns finding this path into an elegant and

efficient way to smoothly interpolate two samples. One major drawback of this approach, however, is the high cost of embedding all given shapes into the manifold. For this reason, many methods, including LIMP [1], choose to construct the path without previously building the complete manifold.

Statistical characterization of the shape space varies in the format in which different models represent the 3D shapes. The most popular ones include 3D voxel grids, point clouds, 3D meshes and implicit representation. The latter views the whole space as a 3D field of points containing specific values in relation to the object. A classical example of this is Signed distance field that maps each point to its distance from the object's surface (negative inside the object). However, some models like ShapeFlow [8], whose goal is not generation but transformation of a shape, instead employ a velocity field or a time-dependent displacement field. Here, each point is associated with a 3D vector describing the transformation, and applying it to the source shape allows to deform it directly. This makes such models more flexible than LIMP [1] that finds an interpolation in the latent representation. This is because on top of the main task, models like LIMP have to solve a difficult problem of generating final shapes from the latent codes, and is thus limited to a specific family of objects.

In order to produce a continous interpolation, one has to analyse how different parts of one shape relate to their counterparts in the other. This brings us to the task of finding the *point correspondences*. For each source point, the goal is to find a mapping to an identical or equivalent target point. One should distinguish between sparse annotation, that involves matching only some points, and a dense one, which maps all the points of both shapes. Another variation presents a so-called soft correspondence (described in Section 2.1).

Most previously existing methods for matching corrseponding points required some degree of supervision. For example, LIMP [1] is trained on shapes with known pointwise correspondences. Interpolation with preservation of the source shape identity imposes additional requirements for the deformation, and is thus even more sensitive to the training data. High cost of obtaining appropriate datasets limits both quantity and quality of existing methods. Many of them are thus incapable of interpolating shapes of different object classes.

To overcome the need for pre-existing annotations, ShapeFlow [8] is trained in an unsupervised manner. However, it does not focus on finding the point correspondences explicitly, which is detrimental in case of large pose variations.

Thus, the idea behind NeuroMorph is that solving both problems explicitly at the same time does not complicate the two solutions, but makes them benefit from each other. On top of that, it does it in a single feed-forward pass and is being trained unsupervised, by matching the corresponding points using local feature descriptors. Unlike ShapeFlow [8], it does not represent the interpolation transformation with a dense vector field, but outputs time-dependent vectors for each vertex of the source shape individually.

2 Method description

2.1 Goal

NeuroMorph is a machine learning-based method that finds soft point correpondence by extracting local feature descriptors and produces continous shape interpolation. It accepts source and the shapes as two 3D meshes $\mathcal{X}$ and $\mathcal{Y}$ and outputs a *correpondence matrix* Π and an *interpolation flow* $\Delta(t)$:

$$f : (\mathcal{X}, \mathcal{Y}) \longmapsto (\Pi, \Delta(t)) \tag{1}$$

Correspondence matrix $\Pi \in \mathbb{R}^{n \times m}$ probabilistically sends the n vertices of the source to the m vertices of the target. It means that the i'th row of the matrix corresponds to a vertex p_i of the source, and the j'th value in that row states the probability that p_i is mapped to the vertex p_j of the target:

$$\Pi_{i,j} = \mathbb{P}(p_i \leftrightarrow p_j) \tag{2}$$

Intuitively, it makes the matrix Π row-stochastic, i.e., such that the values in each row sum up to 1.

The final result of interpolation will eventually be derived as $\Pi\mathbf{Y}$. Since our goal is to keep the identity of the source, we do not want to perfectly match it with the target. The probabilistic format of Π, known as *soft correspondece*, allows to control the proximity of the alignment. Using a correct loss function (described in Section 2.3) can preserve the fine details of the source shape. If needed, *hard correspondence* (strict mapping) can be obtained by imposing a threshold on the values of Π. The results, however, will be rough and noisy, but can be refined using the Smooth shells [4] as a post-processing step.

Interpolation flow $\Delta(t) \in \mathbb{R}^{n \times 3}$ constitutes a stack of n 3D vectors, each representing a shift of a source vertex necessary to produce an intermediate shape at time t:

$$\mathbf{X}(t) = \mathbf{X} + \Delta(t) \tag{3}$$

The time parameter $t \in [0, 1]$ can specify any moment between initial (0) and final (1) steps of interpolation with an arbitrary precision. Along the $[0, 1]$-range, this must transform the source from its inital configuration to the one closest to the target (as defined by the probabilistic matching Π):

$$\mathbf{X}(0) = \mathbf{X} \tag{4}$$

$$\mathbf{X}(1) = \Pi\mathbf{Y} \tag{5}$$

2.2 Approach

The trainable neural network that NeuroMorph uses to find corresponding points does not output Π directly. Instead, it extracts local features of each point. Then, the method uses them to calculate individual entries of the matrix Π.

First, each vertex of a shape is assigned a feature vector initialized as $\tilde{\mathbf{X}} = (\mathbf{X}, \mathbf{N})$, where $mathbfX$ stores the coordinates of all the vertices, and $\mathbf{N}$—their normal vectors. A normal vector of each vertex is an average of face normals adjacent

to it. After that, the model refines each feature descriptor using an EdgeConv [18] graph convolution operator. The procedure is performed in two repeated steps called *local feature aggreagation* and *global feature pooling*.

Local feature aggregation combines a feature vector $\tilde{\mathbf{x}}_i$ of each vertex i with vectors $\tilde{\mathbf{x}_j}$ of all vertices j in its neighborhood $\mathcal{E}$ one by one. Each resultant vector is passed to a small residual network h_Φ, and the resultant updated vector $\tilde{\mathbf{x}'}$ is taken by max-pooling over all the neighborhood:

$$\tilde{\mathbf{x}}'_i := \max_{j:(i,j)\in\mathcal{E}} h_\Phi(\tilde{\mathbf{x}}_j, \tilde{\mathbf{x}}_j - \tilde{\mathbf{x}}_i) \tag{6}$$

Global feature pooling simply means appending a feature vector of each vertex with the global maximum across all of them:

$$\tilde{\mathbf{x}}''_i := (\tilde{\mathbf{x}}'_i, \max_{1\leq i\leq n} \tilde{\mathbf{x}}'_i) \tag{7}$$

Both steps are repeated iteratively five times ($\tilde{\mathbf{X}} \to \tilde{\mathbf{X}}' \to \tilde{\mathbf{X}}'' \to \ldots$). The procedure, denoted as Φ, is applied to both source and target and produces a matrix of vertically stacked feature vectors for each of them:

$$\tilde{\mathbf{X}} = \Phi(\mathcal{X}) \in \mathbb{R}^{n\times d} \tag{8}$$

$$\tilde{\mathbf{Y}} = \Phi(\mathcal{Y}) \in \mathbb{R}^{m\times d} \tag{9}$$

After that, NeuroMorph considers all pairs of vertices and calculates the probability of their correspondence as a *cosine similarity* of their feature vectors:

$$s_{i,j} := \frac{\langle \tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_j \rangle_2}{\|\tilde{\mathbf{x}}_i\|_2 \|\tilde{\mathbf{y}}_j\|_2} \tag{10}$$

The results are normalized per-row using *softmax operator*, and the corresponding values are written as entries of the final matrix:

$$\Pi_{i,j} := \frac{\exp(\sigma s_{i,j})}{\sum_{k=1}^{m} \exp(\sigma s_{ik})} \tag{11}$$

The parameter σ here affects the deviation of the per-row probability distribution. Specifically, it controls how "spiky" concentrated the probability of assigning one point to another is. When the value is low, the probabilities will be distributed more evenly.

Derivation of the time-dependent interpolation flow $\Delta(t)$ boils down to essentially the same task of providing a vector per vertex of a given shape. Thus, NeuroMorph tackles this it using the similar EdgeConv-based architecture. The difference is the purpose of these vectors, as each defines the displacement of a vertex at time t. As such, they obviously must be 3-dimensional and time-dependent. To achieve that, they are initialized by capturing initial positions, distances from the final the destinations $\Pi\mathbf{Y}$ and the time parameter t, equally broadcasted to all vertices:

$$\mathbf{Z} := (\mathbf{X}, \Pi\mathbf{Y} - \mathbf{X}, \mathbf{1}t) \tag{12}$$

After that, the procedure is equivalent to the one described above ($\mathbf{Z} \to \mathbf{Z}' \to \mathbf{Z}'' \ldots \to \mathbf{V}$) and again repeated five times. The final matrix $\mathbf{V} \in \mathbb{R}^{n\times 3}$ is scaled by the time parameter t:

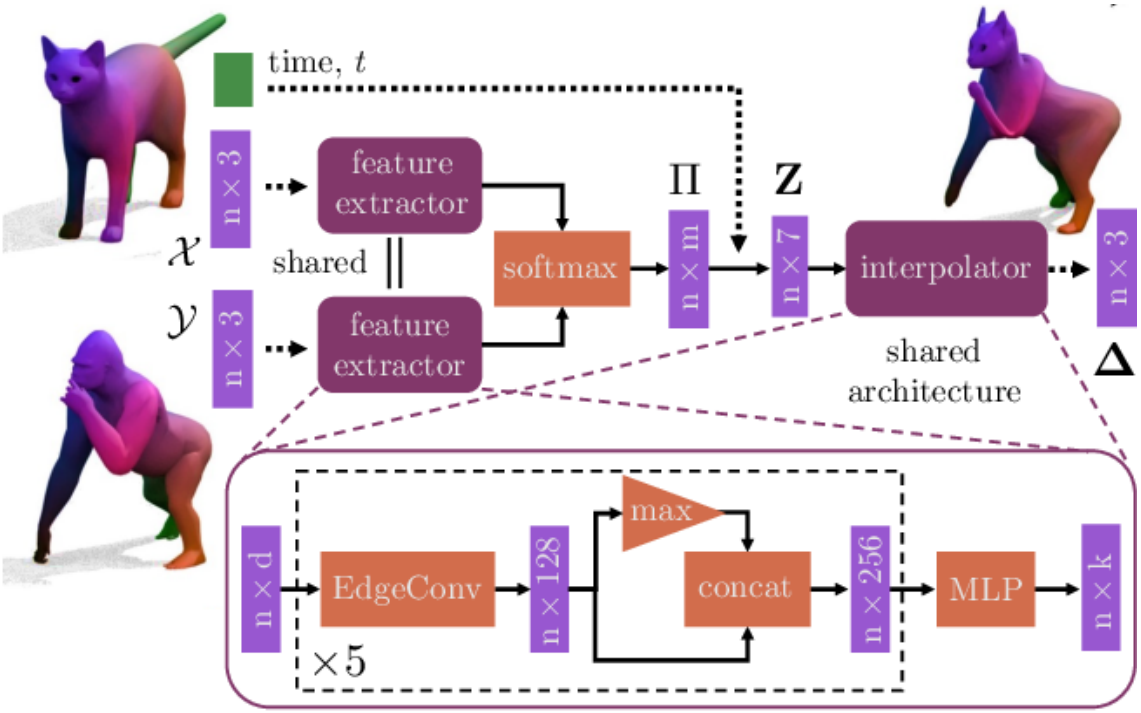

Figure 1: Full architecture of NeuroMorph

$$\Delta(t) := t\mathbf{V}(t) \tag{13}$$

The whole architecture of both models in the method is shown in Figure 1.

2.3 Training

Finding a correct point correspondence is enough to produce a simplistic smooth interpolation, since all that is needed is to move the source points to their corresponding target positions along straight lines. However, two problems will appear in that case. First, the interpolation will end up turning the source shape into a target as closely as structurally possible. Our goal, however, is to imitate the target pose without changing the source identity. Secondly, the intermediate stages of such an interpolation will generally not care about the intermediate shapes, making the source unrecognizeable. To tackle these issues, a sophisticated loss function was constructed training:

$$\ell := \lambda_{reg}\ell_{reg} + \lambda_{arap}\ell_{arap} + \lambda_{geo}\ell_{geo} \tag{14}$$

The function was derived with two goals to be achieved by interpolation: to bring the source as close as possible to the target and to preserve its identity in the process. The function affects both point correspondence and shape interpolation and consists of three components: *registration loss*, *as-rigid-as-possible loss* and *geodesic distance preservation loss*.

Registration loss is responsible for minimizing the distance between the target and the shape $\mathbf{X}_T$ that the source turns into by the end of interpolation.

$$\ell_{reg}(\mathbf{X}_T, \mathbf{Y}, \Pi) := \|\Pi\mathbf{Y} - \mathbf{X}_T\|_2^2 \tag{15}$$

Applying registration loss is enough to achieve a naive deformation that would turn source into target (almost) exactly. The only way to preserve the identifying details of the source is to transform in rigidly. A *rigid transformation* allows translation and rotation of the object, but not stretching, scaling or changing its topology. A perfectly rigid transformation is impossible when our goal is to alter the object's pose. However, the following metric [16] allows to penalise deviating from it:

$$E_{arap}(\mathbf{X}_k, \mathbf{X}_{k+1}) := \frac{1}{2} \min_{\substack{\mathbf{R}_i \in SO(3) \\ i=1,\dots,n}} \sum_{(i,j)\in\mathcal{E}} \|\mathbf{R}_i(\mathbf{X}_{k,j} - \mathbf{X}_{k,i}) - (\mathbf{X}_{k+1,j} - \mathbf{k} + \mathbf{1}, i)\|_2^2 \tag{16}$$

While $t \in [0, 1]$ represents a continous range of time values, the training is done on discrete steps. In the equation above, the current step is denoted with k. Applying this metric to the samples from all used time steps produces a function known as *as-rigid-as-possible loss* [16] — the second component of the loss used for training NeuroMorph:

$$\ell_{arap}(\mathbf{X}_0, \dots, \mathbf{X}_T) := \sum_{k=0}^{T-1} E_{arap}(\mathbf{X}_k, \mathbf{X}_{k+1}) + E_{arap}(\mathbf{X}_{k+1}, \mathbf{X_k}) \tag{17}$$

Finally, to decrease the amount of local stretch of the mesh, a *geodesic distance preservation loss* is used. Given geodesic distances between all pairs of vertices in both shapes in a form of matrices $\mathbf{D}_\mathcal{X}$ and $\mathbf{D}_\mathcal{Y}$, its goal is to preserve them after the interpolation:

$$\ell_{geo}(\Pi) := \|\Pi\mathbf{D}_\mathcal{Y}\Pi^T - \mathbf{D}_\mathcal{X}\|_2^2 \tag{18}$$

To enable unsupervised training of the model, the correspondence matrix Π used in equations 16 and 18 is the one constructed by the model itself.

3 Experiments and results

The performance of NeuroMorph on both solved tasks were tested separately.

For finding point correpondence, the first dataset that NeuroMorph and its competitors were tested on is *FAUST*. It contains shapes of 10 humans in 10 different poses each. The tests were done on a newer version of the datset, where each shape was re-meshed individually. This makes the task more challenging, but the tests produce more realistic results. The second one, called *SHREC20*, contains 14 shapes of different animals. A good feature of these shapes is that they are *non-isometric*, i.e., contain varying topology — added holes, incomplete geometries etc. A drawback of this dataset is that the ground truth labelling of corresponding points there is sparse. This does not allow to use most supervised methods on it. To leverage the best features of both datasets, a new one was created specifically for this testing. *G-S-H* (Galgo, Sphynx, Human) dataset contains shapes of a dog Galgo, a

cat Sphynx and a Human in various poses. Unlike most other datasets, it combines non-isometries with dense ground-truth labels.

The performance of different methods was compared using Princeton benchmark protocol [9]—the metric is a geodesic distance between predicted and ground-truth matches normalized by the square root area of the mesh. Some of the tested methods, including NeuroMorph, use post-processing refinement. For those, the performance on FAUST was compared both with and without these steps, as shown in Table 1. The results obtained using unsupervised methods on other datasets can be seen in Figure 2. NeuroMorph has shown itself as superior to all its competitors on all three datasets.

		err.	p.p.	w/o p.p.
Axiom.	BCICP [13]	6.4	—	—
	ZoomOut [11]	6.1	—	—
	Smooth Shells [4]	2.5	—	—
Sup.	3D-CODED [6]	2.5	—	—
	FMNet [10]	5.9	PMF [17]	11
	GeoFMNet [2]	1.9	ZO [11]	3.1
Unsup.	SurFMNet [14]	7.4	ICP [12]	15
	Unsup. FMNet [7]	5.7	PMF [17]	10
	Weakly sup. FMNet [15]	1.9	ZO [11]	3.3
	Deep shells [5]	1.7	—	—
	NeuroMorph	1.5	SL [4]	2.3

Table 1: Comparison of testing results for various point matching methods on FAUST. The measurements are given as mean geodesic error. Some methods include post-processing refinement using additional methods (PMF [17] — product manifold filter, ZO [11] — ZoomOut, ICP [12] — iterative closest points, SL [4] — smooth shells).

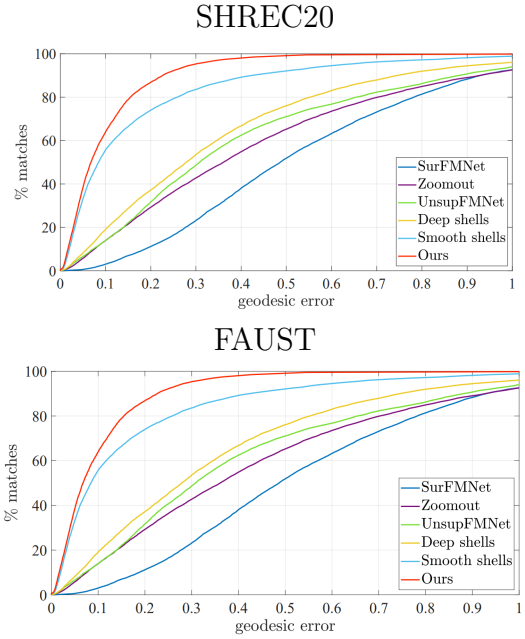

Figure 2: Geodesic error of unsupervised correspondences in percent of diameter.

For shape interpolation, other than FAUST, the performance was tested on *MANO* — a dataset of shapes of a human hand in various poses.

Two metrics were used for comparison. *Conformal distortion* measures deformation of individual mesh triangles during the transformation. Assuming that interpolation is described as a linear function $F(\mathcal{X}) = A\mathcal{X} + b$, the metric is calcualted as

$$\kappa_F(A) = \frac{trace(A^T A)}{\det(A)} \tag{19}$$

The second metric, *Chamfer distance*, is traditionally used for point clouds. It estimates general similarity of shapes by summing and normalizing square distances of each point in one shape to the closest one in another:

$$CD(S_1, S_2) = \frac{1}{S_1} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \frac{1}{S_2} \sum_{y \in S_2} \min_{x \in S_1} \|y - x\|_2^2 \tag{20}$$

The results measured with both metrics are displayed in Figure [**?**]. Here, NeuroMorph was compared against Hamiltonian interpolation [3], LIMP [1] and Shape-

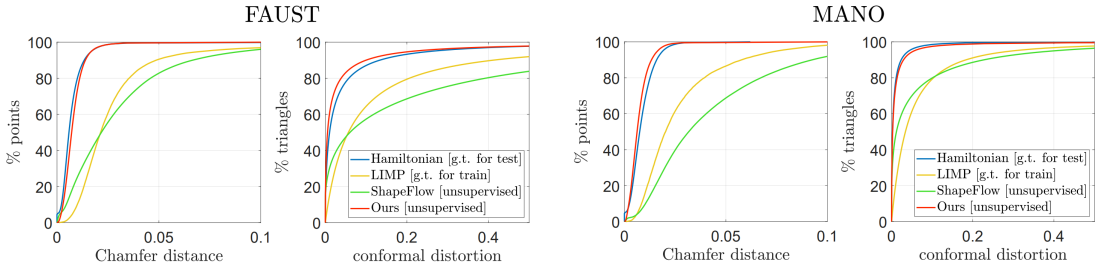

Figure 3: Performance of three 3D shape interpolation methods in comparison to NeuroMorph, measured using Chamfer distance and conformal distortion

Flow [8]. It has performed on par with Hamiltonian and overtaken both other competitors.

4 Discussion / Conclusion

As shown in Table 1, NeuroMorph is superior to all of its competitors in finding corresponding points on the remeshed FAUST dataset. Sometimes, it was the case even without the post-processing step. Its better performance on other two datasets, illustrated on Figure 2, can be explained by the fact that other methods assume isometric nature of shapes, which the shapes there do not satisfy.

For shape interpolation, ShapeFlow [8] has shown the worst performance of all considered approaches. Despite being similar to NeuroMorph in being trained unsupervised, it does not explicitly find the point correspondence as part of the solution, which can explain its inferior results. LIMP [1] is a supervised method, requiring ground-truth labelling of points at the training time. But despite showing a better outcome than ShapeFlow, it still loses the battle to NeuroMorph. However, the most surprising was the comparison against Hamiltonian interpolation [3], which has shown itself almost exactly the same way in both metrics despite being an axiomatic method that requires point matching even at test time.

Overall, we see that separating the two tasks and inventing a separate method for solving each of them was a major disadvantage of the previous approaches. Considered them together, conversely, improves the quality of solution for both problems and produces results comparable and superior to supervised and axiomatic methods.

References

[1] Luca Cosmo, Antonio Norelli, Oshri Halimi, Ron Kimmel, and Emanuele Rodolà. LIMP: learning latent shape representations with metric preservation priors. *CoRR*, abs/2003.12283, 2020.

[2] Nicolas Donati, Abhishek Sharma, and Maks Ovsjanikov. Deep geometric functional maps: Robust feature learning for shape correspondence, 2020.

[3] Marvin Eisenberger and Daniel Cremers. Hamiltonian dynamics for real-world shape interpolation, 2020.

[4] Marvin Eisenberger, Zorah Lahner, and Daniel Cremers. Smooth shells: Multi-scale shape registration with functional maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[5] Marvin Eisenberger, Aysim Toker, Laura Leal-Taixé, and Daniel Cremers. Deep shells: Unsupervised shape correspondence with optimal transport, 2020.

[6] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[7] Oshri Halimi, Or Litany, Emanuele Rodolà, Alexander M. Bronstein, and Ron Kimmel. Unsupervised learning of dense shape correspondence. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4365–4374, 2019.

[8] Chiyu "Max" Jiang, Jingwei Huang, Andrea Tagliasacchi, and Leonidas J. Guibas. Shapeflow: Learnable deformations among 3d shapes. *CoRR*, abs/2006.07982, 2020.

[9] Vladimir Kim, Yaron Lipman, and Thomas Funkhouser. Blended intrinsic maps. *ACM Trans. Graph.*, 30:79, 07 2011.

[10] Or Litany, Tal Remez, Emanuele Rodolà, Alex Bronstein, and Michael Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. pages 5660–5668, 10 2017.

[11] Simone Melzi, Jing Ren, Emanuele Rodolà, Abhishek Sharma, Peter Wonka, and Maks Ovsjanikov. Zoomout: Spectral upsampling for efficient shape correspondence. *ACM Trans. Graph.*, 38(6), nov 2019.

[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, 07 2012.

[13] Jing Ren, Adrien Poulenard, Peter Wonka, and Maks Ovsjanikov. Continuous and orientation-preserving correspondences via functional maps. *ACM Trans. Graph.*, 37(6), dec 2018.

[14] Jean-Michel Roufosse, Abhishek Sharma, and Maks Ovsjanikov. Unsupervised deep learning for structured shape matching. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1617–1627, 2019.

[15] Abhishek Sharma and Maks Ovsjanikov. Weakly supervised deep functional map for shape matching, 2020.

[16] Olga Sorkine and Marc Alexa. As-Rigid-As-Possible Surface Modeling. In Alexander Belyaev and Michael Garland, editors, *Geometry Processing*. The Eurographics Association, 2007.

[17] Matthias Vestner, Roee Litman, Emanuele Rodolà, Alex Bronstein, and Daniel Cremers. Product manifold filter: Non-rigid shape correspondence via kernel density estimation in the product space, 2017.

[18] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds, 2018.