

Supplementary Material for: Unsupervised Representation Learning for Diverse Deformable Shape Collections

Sara Hahner ^{*,1,2}

Souhaib Attaiki ^{*,3}

Jochen Garcke ^{1,2}

Maks Ovsjanikov ³

¹Fraunhofer SCAI,
Sankt Augustin, Germany

²Institute for Numerical Simulation,
University of Bonn, Germany

³LIX, École Polytechnique,
Institut Polytechnique de Paris, France

1. Supplementary Material

We compile the results and discussions that were not accommodated in the main manuscript due to page constraints. Specifically, Section 1.1 provides details on the implementation aspects of our pipeline. Section 1.2 elucidates our motivation for unsupervised feature learning on the *TRUCK* dataset and offers additional reconstruction results. Section 1.3 provides additional generative results. Section 1.4 introduces an ablation study concerning the components of our pipeline. Finally, Section 1.5 delves into our interpretation of pooling in 2D and surface meshes, and it formulates and provides proof for the Lemma mentioned in Section 4.1 of the main text.

1.1. Implementation Details

For our experiments concerning the extraction of point-to-point maps in Section 4.2 of the main text, we use a functional map of size $k = 30$. Concerning the feature extractor DiffusionNet [9], we use the default segmentation configuration provided by the authors¹. After extracting the first set of functional maps, we refine them using ZoomOut [7] using 30 iterations, from $k = 30$ to $k = 120$. For the Laplace-Beltrami computation, we use the cotangent discretization scheme [8].

Concerning our autoencoder architecture in Section 4.3 of the main text, we use the same segmentation configuration of DiffusionNet for both the encoder and decoder. When using true point-to-point maps as supervision, we do not apply dropout inside the DiffusionNet blocks. We chose F , the number of features output by the DiffusionNet layer in the encoder, and k_2 , the dimensions of the CCLB, for

all the shape collections in a way, such that the embedding dimension is approximately $k_2 \times F = 1024$.

In all our experiments, we train our networks using Adam optimizer [5] with an initial learning rate of 0.001. For the autoencoder training losses in Section 4.4 of the main text, we use $\lambda = 10$. Concerning the reconstruction loss, due to the large size of the matrices D^S and D^X , the shapes X and IS are resampled to 20000 vertices if they are larger than it, only during the loss computation.

1.2. *TRUCK* shape collection and additional reconstruction results

In a car crash simulation, the different car components are generally represented by surface meshes, which makes our method applicable to this kind of data. From simulation run to simulation run, the car model parameters are modified to achieve multiple design goals, e.g. crash safety, weight, or performance. Depending on the chosen model and simulation parameters, the car model often deforms in different patterns. Since the simulations nowadays contain detailed information for up to two hundred time steps and more than ten million nodes, their analysis is challenging and generally assisted by dimension reduction methods. One goal is the detection of clusters corresponding to different deformation patterns in the components' embeddings. We visualize our 2D embedding of two components that deform in 32 simulations over time in Figure 5 in the main text. This way, relations between model parameters and the deformation behavior are discovered more easily, and the analysis of car crash simulations is accelerated [2, 3]. We provide reconstruction results from the supervised experiment of a car component from the *TRUCK* dataset in Figure 1, which manifests two different deformation patterns over time, that are visible in the embedding space in Figure 5.

In the supplementary, we add an additional experiment on the *SCAPE* dataset [1] in the supervised "unknown poses" experiment on *FAUST* by using 10 additional

(*) denotes equal contribution

¹<https://github.com/nmwsharp/diffusion-net>

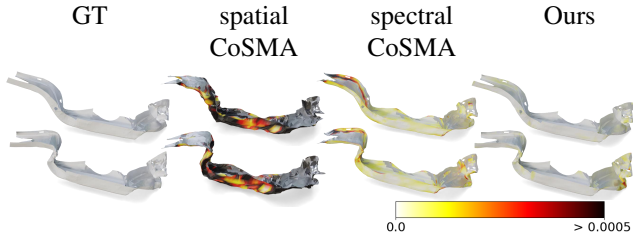



Figure 1. Reconstructions of a car component, which deforms in two different patterns (first and second row), from the *TRUCK* dataset. Vertex-wise error is highlighted.



Method	<i>SCAPE</i>	<i>FAUST</i> “unknown poses”
Ours	3.2	2.4

Figure 2. Left: Euclidean errors between the reconstructed and original meshes of *FAUST* and *SCAPE* datasets. Right: Reconstructed meshes from *SCAPE* dataset. Vertex-wise error is highlighted.

SCAPE shapes only for testing. We consider the most diverse *SCAPE* shapes and provide Euclidean errors as well as reconstruction results in Figure 2.

1.3. Additional Generative Experiments

We provide additional generated shapes by combining two positions and transferring a pose from one to another individual.

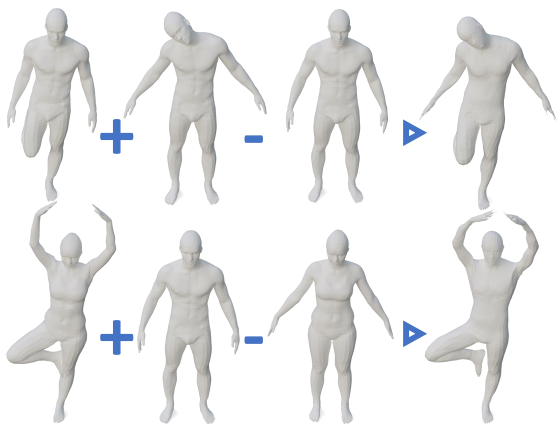


Figure 3. Combining two positions of *FAUST* test shapes (upper row) and transferring the pose from a female to a male individual.

1.4. Ablation Study

To validate our pipeline’s components, we performed an ablation study.

First, we wanted to examine the role of spectral pooling and the reconstruction loss. It’s important to note that the reconstruction loss can’t be used by itself since it’s rotation-invariant. For this experiment, we utilized the *FAUST* dataset in an interpolation setting, mirroring Section 5.2.1. We carried out four experiments: the first one employed our complete pipeline with supervised maps; the second used unsupervised maps; the third operated with unsupervised maps but omitted the reconstruction loss; and the fourth involved supervised maps without the spectral pooling (instead, we opted for global pooling as presented in Table 1 of the main text, following the approach of previous studies like [6]). The outcomes are presented in Table 1. They indicate that each component is crucial for achieving the best results. Notably, the spectral mesh pooling’s contribution to the combined embedding space is significant; using just global pooling leads to a marked drop in performance.

Setting	<i>FAUST</i> dataset
w/o limit shape	16.7
w/o reconstruction loss	4.3
with unsupervised maps	2.0
with supervised maps	0.7

Table 1. Ablation study on the component of our pipeline.

Secondly, we study the impact of the size of the projection on the limit shape k_2 . To do this, we use the *GALLOP* dataset in a supervised setting similar to Section 5.2.1. We keep the size of the embedding space fixed (equal to 1024), which is determined by k_2 , the size of the limit shape multiplied by the feature dimensions F of the encoder. We increase k_2 monotonically from $k_2 = 1$ to $k_2 = 70$, while adapting the feature dimension F accordingly. The results are summarized in Figure 4, which shows that the higher the dimension k_2 of the limit shape, the better the performance, corresponding to a bigger pooling in the spectral space. However, it can also be seen that performance starts to deteriorate with bigger k_2 . This is explained by the fact that we keep the dimension $k_2 \times F$ of the embedding space fixed, and hence fewer features are extracted with higher k_2 , which is not sufficient for encoding and high-quality decoding.

1.5. Motivation: Pooling in 2D as a projection to a common basis and Lemma 1

In Section 4.1 of the main text, we introduced a new spectral mesh pooling operator. This operator reduces the di-

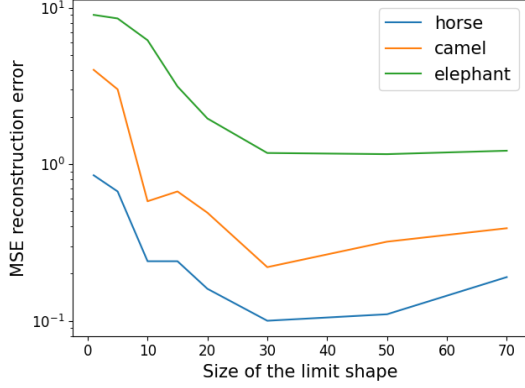


Figure 4. Impact of k_2 , the size of the limit shape. If $k_2 = 1$ this corresponds to global average pooling; see Section 1.5.

dimensionality of the meshes, enabling us to manage meshes with varying connectivity and represent them in a unified low-dimensional embedding space.

In the case of classical representation learning for 2D images with convolutional networks, one has a fixed-size grid and, in fact, all samples are in 1-to-1 correspondence. The convolutional filters with stride 1 calculate vertex-wise features, then pooling summarizes many vertex-wise features, going from n pixels to k . This is done symmetrically for all the images and the features from different samples are comparable to each other because of the 1-to-1 correspondence. Let us consider the canonical basis for images of size $n = 4 \times 4$, and the pooling size $k = 2 \times 2$, in this case:

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

is a common basis for all the images with 2×2 pixels. The projection vector from n pixels towards one of low-dimensional basis has ones in the corresponding corner. For the first common basis, it is

$$\frac{1}{4} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

displayed in the shape of the image. Therefore, pooling in 2D can also be interpreted as a projection from n dimensions to a set of common basis functions in k dimensions. This projection reduces the dimensionality of the data while the dimensionality of the pixel-wise features stays the same. Because of the 1-to-1 correspondence, all the images are described in the same basis. A similar pooling operator cannot be constructed for meshes with different mesh connectivities. We can only obtain point-to-point maps between the shapes that allow the projection of a function from one shape to another.

To solve the pooling for meshes, we propose to adapt the CCLB method (initially developed for deformation detection) and introduce a novel intrinsic spectral mesh pooling. We project vertex-wise features that are calculated for every shape separately to the common CCLB basis, reducing the dimension from the number of vertices to the size of the limit shape. We calculate the limit shape basis CCLB as described in section 3.4. It has dimension k_2 and uses eigen-decompositions of the Laplacians of size $k_1 \geq k_2$. It will be the common basis for the low-dimensional embedding space. For the spectral unpooling, we project the features from the limit shape basis back to the vertex representation.

If the dimensionality of the CCLB is 1 ($k_1 = k_2 = 1$), the projection of the shape features into the CCLB corresponds to a global \pm mean pooling for all the shapes in the collection. Furthermore, the inverse of this operation duplicates the average feature into the shape’s vertices, similar to upsampling in the 2D case. Also, the sign of the resulting global mean pooling function from all shapes in the shape collection to the CCLB is the same, which makes the different low-dimensional representations comparable to each other. We formally state this observation in the following lemma.

Lemma 1. *If $k_1 = k_2 = 1$, there are only two possible solutions for the projection $\tilde{Y}_i^\dagger \Phi_i^\dagger$ from the vertex-wise features to the CCLB for all shapes $S_i, i = 1, 2, \dots$ and the projection from the CCLB to a template shape $\Phi_t \tilde{Y}_t$. Either*

$$\tilde{Y}_i^\dagger \Phi_i^\dagger x_i = \text{mean}(x_i) \quad \forall i \text{ and } \Phi_t \tilde{Y}_t = \mathbf{1}_{n_t} \quad (1)$$

or

$$\tilde{Y}_i^\dagger \Phi_i^\dagger x_i = -\text{mean}(x_i) \quad \forall i \text{ and } \Phi_t \tilde{Y}_t = -\mathbf{1}_{n_t} \quad (2)$$

with $\text{mean} : \mathbb{R}^{n_i \times d} \rightarrow \mathbb{R}^d$ is the vertex-wise average function, and $\mathbf{1}_{n_t}$ is the column-vector with only ones in \mathbb{R}^{n_t} , and n_t being the number of vertices of the template shape.

Proof. At first, we proof that $\tilde{Y}_i^\dagger \Phi_i^\dagger x_i = \pm \text{mean}(x_i)$ for a fixed i . If $k_1 = k_2 = 1$ we have

$$\Phi_i = \pm \mathbf{1}_{n_i} \quad (3)$$

being the eigenvector corresponding to the smallest eigenvalue $\Lambda_i = 0$, because the sum of all values in each row of the Laplacian \mathcal{L}_i is 1. Therefore,

$$\Phi_i^\dagger x_i = \frac{1}{n_i} \Phi_i^T x_i = \pm \text{mean}(x_i). \quad (4)$$

The functional map

$$C_{ij} = \Phi_j^T \Phi_i \in \mathbb{R}^{1 \times 1} \quad (5)$$

is 1 or -1, projecting only constant functions from shape j to shape i . It holds $C_{ij} = C_{ji}$. If $k_1 = 1$, the optimization

problem to compute the Consistent Latent Basis (CLB)

$$\min_Y \|C_{ij}Y_i - Y_j\| \text{ s.t. } \sum_i Y_i^T Y_i = I \quad (6)$$

has the solutions:

$$\begin{aligned} \text{if } C_{ij} = C_{ji} = 1 &\Rightarrow Y_i = Y_j \in \{-1, 1\} \\ \text{else } C_{ij} = C_{ji} = -1 &\Rightarrow Y_i = -Y_j \in \{-1, 1\}. \end{aligned} \quad (7)$$

Since $\Lambda_i = 0$, the matrix E in algorithm 1 from [4] is 0. Therefore, the possible solutions for its eigenvector U are -1 and 1. For the calculation of the CCLB, this leads to

$$\tilde{Y}_i = Y_i U \in \{1, -1\}. \quad (8)$$

The inverse holds

$$\tilde{Y}_i^\dagger = \tilde{Y}_i. \quad (9)$$

From (4) and (9) follows

$$\tilde{Y}_i^\dagger \Phi_i^\dagger x_i = \pm \text{mean}(x_i) \quad (10)$$

and all entries of the matrix have the same sign.

In a second step, we prove by contradiction that the non-zero entries of the matrix products $\tilde{Y}_i^\dagger \Phi_i^\dagger$ have the same sign for all $i = 1, 2, \dots$.

Assume that the sign of $\tilde{Y}_i^\dagger \Phi_i^\dagger$ is different from the sign of $\tilde{Y}_j^\dagger \Phi_j^\dagger$ for $i \neq j$. Without loss of generality, assume the sign of $\tilde{Y}_i^\dagger \Phi_i^\dagger$ to be positive. Therefore, the sign of \tilde{Y}_i^\dagger is the same as the sign of Φ_i^\dagger . Then, either \tilde{Y}_j^\dagger or Φ_j^\dagger has a different sign.

If $\tilde{Y}_j^\dagger = -\tilde{Y}_i^\dagger$, then $Y_j = -Y_i$ and therefore $C_{ij} = C_{ji} = -1$ because Y_i and Y_j solve (6). From (5) follows that Φ_j and Φ_i have different signs, which is a contradiction to Φ_i^\dagger having the same sign as Φ_j^\dagger .

If in the other case Φ_j^\dagger has a different sign than Φ_i^\dagger , $C_{ij} = C_{ji} = -1$ because of (5). It follows $Y_i = -Y_j$, which is a contradiction to \tilde{Y}_j^\dagger having the same sign as \tilde{Y}_i^\dagger .

Finally, the entries of the matrix product $\Phi_t \tilde{Y}_t$, which projects the features from the CCLB representation to the template shape, have the same sign as $\tilde{Y}_i^\dagger \Phi_i^\dagger$. \square

References

- [1] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. SCAPE: Shape Completion and Animation of People. *ACM Transactions on Graphics*, 24(3):408–416, 2005. 1
- [2] Bastian Bohn, Jochen Garcke, Rodrigo Iza-Teran, Alexander Paprotny, Benjamin Peherstorfer, Ulf Schepsmeier, and Clemens August Thole. Analysis of car crash simulation data with nonlinear machine learning methods. *Procedia Computer Science*, 18:621–630, 2013. 1
- [3] Sara Hahner, Rodrigo Iza-Teran, and Jochen Garcke. Analysis and prediction of deforming 3D shapes using oriented bounding boxes and LSTM autoencoders. In *Artificial Neural Networks and Machine Learning – ICANN 2020*, pages 284–296. Springer International Publishing, 2020. 1
- [4] Ruqi Huang, Panos Achlioptas, Leonidas Guibas, and Maks Ovsjanikov. Limit shapes - a tool for understanding shape differences and variability in 3D model collections. *Eurographics Symposium on Geometry Processing*, 38:187–202, 2019. 4
- [5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 1
- [6] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1886–1895. IEEE, 2018. 2
- [7] Simone Melzi, Jing Ren, Emanuele Rodolà, Abhishek Sharma, Peter Wonka, and Maks Ovsjanikov. ZoomOut: Spectral upsampling for efficient shape correspondence. *ACM Transactions on Graphics*, 38(6):1–14, 2019. 1
- [8] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993. 1
- [9] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. DiffusionNet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics*, 41(3):1–16, 2022. 1