Supplementary Material: Adaptive Spiral Layers for Efficient 3D Representation Learning on Meshes

Francesca Babiloni[1,2], Matteo Maggioni[1], Thomas Tanay[1], Jiankang Deng[1,2],
Ales Leonardis[1], Stefanos Zafeiriou[2]
[1]Huawei, Noah's Ark Lab [2]Imperial College London
{f.babiloni22, j.deng16, s.zafeiriou}@imperial.ac.uk
{matteo.maggioni, thomas.tanay, ales.leonardis}@huawei.com

This supplementary material provides further implementation details, experimental results, and visualizations, not included in the main text.

A. Implementation Details

We provide additional details on the architectures used in the experiments of the main paper.

A.1. 3D Correspondence Architecture

We perform experiments on the FAUST dataset [1]. Following [10], our network takes as input features the vertex coordinates in their raw 3D XYZ form and involves a sequence of linear layers, (Lin) and mesh operators (MConv) to process them. The overall architecture is built as a stack of: Lin(16) $\rightarrow$ 3 $\times$ {MConv(16)} $\rightarrow$ Lin(16) $\rightarrow$ Dropout(0.5) $\rightarrow$ Lin(6890), where the number of output features is indicated in parentheses. Trainable weights are initialized using Glorot initialization [6], and trainable biases are initialized with a constant value of 0. We use an exponential linear unit (ELU) as the non-linear activation function on the output of each mesh-convolutional layer. We use a Dropout regularization layer [13]. Different from previous state-of-the-art methods, our architecture uses a fixed number of hidden features (16) for all layers. The kernel size for the local operator is set to $K = 9$. In the main manuscript, we compare against SpiralConvolution using the architecture detailed above. In the experiments discussed in Sec. 5.1.1 of the main paper, we replace the MConv operator with a SpiralConvolution [7] and benchmarked three different values of hidden features $(16, 32, 64)$. All experiments ran on a single Tesla V100. We used the deep learning computing libraries of Pytorch [11] and MindSpore [8]. The number of floating-point operations (FLOPS) and parameter count are benchmarked using the fvcore package.

A.2. 3D Reconstruction with 3DMM Architecture

We perform experiments on the CoMA dataset [12]. Following [3], our architecture is an encoder-decoder network with 4-stages, where the decoder structure mirrors the encoder and replaces downsampling layers (Pool) with upsampling layers (UnPool). We used the same downsampling and upsampling approach introduced in [12]. To process the output, the network uses a sequence of mesh operators (MConv), and fully-connected layers in the latent space (FC) characterized by a latent space dimension $d = 16$. After the last encoder layer, the output is obtained via a last MConv refinement layer. The overall architecture can be thus defined as:
3 $\times$ {MConv(32) $\rightarrow$ Pool(4)} $\rightarrow$ {MConv(64) $\rightarrow$ Pool(4)} $\rightarrow$ FC(16) $\rightarrow$ FC(16) $\rightarrow$ {UnPool(4) $\rightarrow$ MConv(64)} $\rightarrow$ 3 $\times$ {UnPool(4) $\rightarrow$ MConv(32)} $\rightarrow$ MConv(3), with ELU activation functions used after each MConv layer in the encoder and decoder. Starting from this baseline architecture, we experimented with different 3D Morphable Model (3DMM) variants, ablating the placement of our operator and kernel sizes. In particular we report results for three different configurations: Ours-II, -III, -IV. Ours-II variant uses MConv implemented as [SpiralConv, SpiralConv, Ours, Ours]. Ours-III uses [SpiralConv, Ours,Ours, Ours]. Lastly, Ours-IV uses our mesh-operator in every MConv layer. Moreover, we experimented with different kernel sizes for the local processing $K = [4, 9, 14]$. All experiments ran on a single Tesla V100.

A.3. Learnable Pooling

In Sec. 4 of the main paper, we suggest relaxing the SpiralConvolution bias of local and static receptive fields using a learnable pooling function which allows our operator to dynamically expand its receptive field to arbitrary sizes, and potentially even to the entire mesh without the need to compute the weight tensor of N^2 parameters. We give here some additional implementation details. First, recall that

the output of the learnable pooling can be written as:

$$X^3_{inr} = \left[X^2_{in\underline{m}r} W^S_{\underline{m}l}\right]_{in\underline{l}r} W^P_{in\underline{l}} \quad (1)$$

where W^S_{ml} is a triangular matrix of ones implementing a cumulative sum over X^2_{inmr} along the m dimension and $W^P_{iln} = f^P(X_{inmr})$ is a dynamic function predicting the size of the receptive field for each mesh and vertex as one-hot vectors along the l dimension, with ($l \in [1, M]$).

In practice, we implement f^P in two steps. First, we apply a linear layer to X_{inmr} to predict a tensor of receptive sizes $\mathbf{s}_{in}$, restricted to the range $[0, 1]$ with a clip operator and then scaled to [0,M]. Then we convert this tensor into one-hot encodings W^P_{iln}. To ensure that the operation is differentiable, we implement a smooth one-hot encoding as:

$$W^P_{iln} = (1-\beta)\cdot \text{OneHot}(\lfloor \mathbf{s}_{in} \rfloor) + \beta \cdot \text{OneHot}(\lceil \mathbf{s}_{in} \rceil) \quad (2)$$

where OneHot(.) is a function that converts an integer into its one-hot representation of length M, $\lfloor . \rfloor$ and $\lceil . \rceil$ are the floor and ceiling operators and $\beta = \mathbf{s}_{in} - \lfloor \mathbf{s}_{in} \rfloor$.

B. Additional Experimental Results

In this section, we analyze the generalization ability of our method by presenting additional experimental results on large scale networks and two extra datasets. Further, we analyze the sensitivity of our method to the choice of non-linear activation function, the number of components used to approximate the W_{imncd} tensor.

B.1. Efficiency

In Fig. 2 of the main paper, we measure computational efficiency using Floating Point Operations (FLOPS) as this is agnostic to both hardware and implementation. This analysis shows how our method with 0.82 GFLOPS *vastly* outperforms a SpiralConv with a similar complexity (0.81 GFLOPS). Here, we also report run-times measured in milliseconds (ms) per mesh on a V100 Tesla GPU. In the table 1, we compare 3DMM equipped with various mesh operators under fair conditions and comparable implementations. As clearly visible, as our method exhibits the highest accuracy and the lowest run-time among all compared operators.

Method	GAT	FeastNet	MoNet	SpiralNet	Ours
Params	50k	49k	**48k**	**48k**	54k
Error (mm)	0.762	0.750	0.708	0.635	**0.544**
Time	12.77s	15,37s	10.55s	8.18s	**6.88s**

Table 1: **Our method vs soft-attention operators**.

B.2. Evaluation on large models

To demonstrate that performance of our approach is primarily influenced by its intrinsic design, rather than the overall capacity, we test networks with higher parameter count. For this set of experiments, we use the experimental setup of [5] and augment the features of our model in a progressive manner as $[3, 16, 32, 64, 128]$, increasing the latent representation from $d = 16$ to $d = 32$. Results are reported in Table 2. Without any bells and whistles, our method vastly outperforms all compared methods trained under the same experimental setup. Ours-III reaches a median error of 0.106 mm and a parameter count reduced by almost a factor of three (2.74) when compared to the current state of the art of LSA. Furthermore, our method exhibits robust scalability. When its number of parameters is set to approximately 2M (Ours-III-v2) by increasing the embedding size to $[3, 32, 64, 128, 256]$, it significantly outperforms an analogous LSA network, by reducing the median error from 0.115 mm to 0.084 mm, setting a new state-of-the-art benchmark on COMA.

Method	Median Error (mm)	Params
CoMA*	0.248	**303K**
PCA*	0.210	482K
SpiralNet++	0.136	500K
LSA-small+ [5]	0.167	551K
LSA-Conv+ [5]	0.115	1886K
Ours-II	0.120	508K
Ours-III	**0.106**	674K
Ours-III-v2	**0.084**	1902K

Table 2: **Large scale 3DMM models**. Results of the method marked with "*" are reported as in [5]. Our method showcases scalability and sets a new state-of-the-art performance on the COMA dataset.

B.3. Evaluation on additional datasets

We replicate experiments on 3DMM reconstruction on two additional challenging datasets: Dynamic FAUST [2] (full-body 3D meshes in a wide range of motions) and SYNHAND [9] (varying hand shapes and poses). In agreement with the rest of the experiments, our novel adaptive formulation *consistently* outperforms SpiralConv in all considered cases.

SYNHAND We explore our method for the generation of hands with various shapes and poses sharing a common template. We follow [14, 4] and use the synthetic hand 5 million meshed dataset SYNHAND by randomly sampling 100K meshes with a 90:10 split for train and evaluation. We replicate the experimental setup of the main paper by comparing our mesh-aware and vertex-aware method to its static SpiralConv++ counterpart in the same architecture for two different kernel sizes $K = 4$ and $K = 9$. As visible in Figure 1 our method outperforms SpiralConv++ by a large margin, with a trend consistent with the rest of the experimental results.

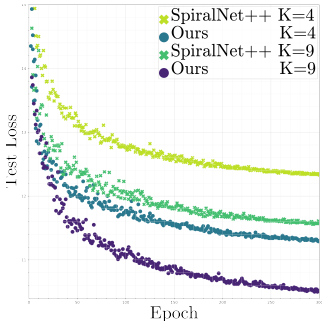

Figure 1: **3DMM on Synhand.** Test curves for kernel size = 4 and 9. Using the same encoder-decoder architecture, a simple drop in replacement of SpiralConv++ layers with our method improve performance by a large margin.

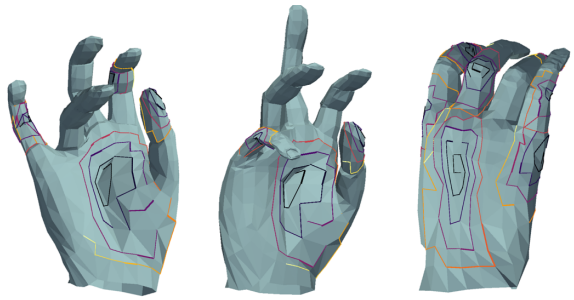

Figure 2: **3DMM on Synhand.** Visualization of receptive field. Our method adapts the size of the receptive field in a vertex-adaptive and mesh-adaptive fashion.

Dynamic FAUST Dynamic FAUST dataset extends the well-known FAUST to a more challenging case, of more than one hundred moving sequences covering a variety of actions. The data 40K+ dynamic human body shapes aligned to a common reference topology containing 6,890 vertices each. We follow [**?**] data split and follow the same experimental setup described in the main paper. We trained our method on various kernel sizes and network sizes, keeping the latent size fixed to 8. We report the performance of our method compared against other 3DMM variants (i.e. CoMa, Neural3DMM) and a 3DMM equipped with SpiralConv++ as a baseline. Results are reported in table 3. As visible, our method is capable of outperforming a standard SpiralNet of 274K parameters reducing by a three-fold its parameter count. Moreover, when compared to a standard PCA explaining 85% of the total variance, our method is capable of reducing the generalization error of 43 mm. When compared to its closest baseline (SpiralConv++) our method is capable of consistently outperforming it by a large margin on all the evaluated settings for just a modest increase in parameter count. Together these results showcase the generalization ability of our method, its high-quality performance-efficiency trade-off in various settings, and the importance of an adaptive processing of meshes for high-quality generation.

Method	K	l	Filter	Params	Error
PCA *	-	-	-	165k	59.30
SpiralNet++	4	8	[16,16,16,16]	15k	33.56
Ours-II	4	8	[16,16,16,16]	20k	30.40
Neural3DMM (small)*	-	8	[16,32,64,128]	41k	28.69
COMA*	9	8	-	32k	28.09
SpiralNet++	9	8	[16,16,16,32]	43k	25.73
Ours-II	9	8	[16,16,16,32]	56k	21.60
Neural3DMM*	-	8	[128,64,32,32,16]	274k	19.77
Ours-III	9	8	[16,16,16,32]	87k	19.14
SpiralNet++	14	8	[16,16,16,32]	59k	22.93
Ours-II	14	8	[16,16,16,32]	77k	19.23
Ours-III	14	8	[16,16,16,32]	126k	16.50

Table 3: **3DMM on Dynamic FAUST**. We report Generalization Error in mm as a metric for performance and parameter count as a proxy for efficiency. l stands for Latent Dimension, K kernel size. The column Filter report the Encoder filter sizes. PCA with 84.8 % Explained Variance. "*" taken from[3] . Our method present the best performance/efficiency trade off in all the evaluated settings.

B.4. Effect of Non-Linearities

In the main paper, we describe our method as a non-linear extension of SpiralConvolution. In this section, we quantitatively show the ability of our method to capture non-linear dependencies even without the direct use of non-linear activation functions. Table 4 shows the results of an ablation study investigating the effect of the ELU activation function on the performance of Ours-IV 3DMM when compared to a SpiralConvolution 3DMM baseline. We use the experimental setup as described in Sec. 5.3 of the main paper and replace the ELU activations with Identities, evaluating both methods using Mean Error and Median Error as performance metrics. As expected, the performance drop for SpiralConv is considerable. Changing the ELU activations with an Identities brings a 50.31% drop in performance (from 0.320 to 0.481 median error). Differently, it can be clearly observed that our method is not impacted at all by the absence of non-linearities. As visible, it achieves slightly better results using an Identity activation function when compared to the use of ELU, with a mean error decreasing from 0.439 to 0.435. Together these results highlight the capacity of our method to work as a non-linear function of the input.

Method	Activation	Dataset	Mean Error	Median Error	Params
SpiralConv	ELU	COMA	0.554 ± 0.674	0.320	92K
SpiralConv	Identity	COMA	0.857 ± 1.111	0.481	92K
Ours-IV	ELU	COMA	0.439 ± 0.599	0.244	160K
Ours-IV	Identity	COMA	0.435 ± 0.595	0.240	160K

Table 4: **Ablation on the use of an activation function**. Our operator acts as a non-linear generalization of SpiralConv. Therefore, is able to capture non-linear correlations between meshes without the need to use an ELU. Experiments use 3DMM with $K = 4$ trained and tested on the COMA dataset.

B.5. Effect of R

As discussed in Sec. 4 of the main paper, our approach involves decomposing the dense weight tensor W_{inmcd} into a set of smaller factor matrices using CP decomposition. The construction of these matrices depends on the number of components employed to approximate the original tensor, denoted as R. In the experiments reported in Sec. 5.3 of the main paper, we assumed that the number of input and output channels, denoted as C and D, respectively, are equal and that the rank of the CP decomposition is equal to the input/output dimensions, i.e., R = C. In this section, we extend the experimental results of the main paper by investigating the sensibility of our method to the choice of R, by reporting how the overall performance is affected when we vary the number of components used by the factor matrices. Table 5 presents the results of an ablation study on the CP decomposition rank R for the 3D Morphable Model (3DMM) Ours-II with kernel size $K = 4$, trained and tested on the COMA dataset. The table shows the results for different values of R and R ratio, which is the ratio of the number of components used with respect to the input/output dimension. As expected, results show how decreasing R results in higher approximation error and lower performance. The method performs best when R is 128 and worst when R is 8, with median errors going from 0.270 to 0.413. It also highlights the importance of an appropriate choice for the number of R components used to approximate the weight tensor. In fact, a reduction in performance does not always result in a decrease in parameter count. For example, choosing a ratio of 0.25 compared to 0.5 implies a decrease in parameter count of 0.98% (from 102K to 101K parameters) but a 17% reduction in performance, (from 0.353 to 0.413 median error). Differently, reducing the ratio from 4.0 to 2.0 show an opposite trend, with the number of parameters cut by 47.3%(from 264K to 139K) but an increase in the median error of 5.6%(from 0.270 to0.285). Overall, these results showcase how the trade-off between model complexity and performance cannot be always justified solely by parameter count, highlighting the impact of the characteristics of the method used.

Method	C,D	R	R ratio	Mean Error	Median Error	Params
Ours	32	128	× 4.0	0.466 ± 0.602	0.270	264K
Ours	32	64	× 2.0	0.492 ± 0.632	0.285	139K
Ours	32	32	× 1.0	0.512 ± 0.665	0.296	106K
Ours	32	16	× 0.5	0.604 ± 0.764	0.353	102K
Ours	32	8	× 0.25	0.697 ± 0.852	0.413	101K

Table 5: **Ablation on the CP decomposition rank R**. We use the 3DMM Ours-II K=4 trained and tested on the COMA dataset. Decreasing R results in higher approximation error and lower performance. R ratio describes the ratio of components used with respect to the value of input and output dimensions.

C. Additional Qualitative Results

In this section, we provide additional qualitative comparisons and visualizations for the model described in Sec. 5 of the main paper.

C.1. Interpolation in the Latent Space

We extend the results on 3DMM of the main paper by providing visualizations for the length of the spirals estimated by the three layers in our network that include our operator. We showcase the ability of our model to generate new realistic representations, transitioning from one expression to another via interpolation. In practice, we choose two samples from our test set, encoding them both in the latent space as Z^1_{mnd}, Z^2_{mnd}. Then we proceed by generating their intermediate encodings by sampling the line that connects them in the latent space as $Z_{mnd} = aZ^1_{mnd} + (1-a)Z^2_{mnd}$, where $a \in (0, 1)$. Visualizations are reported in Figure 3.

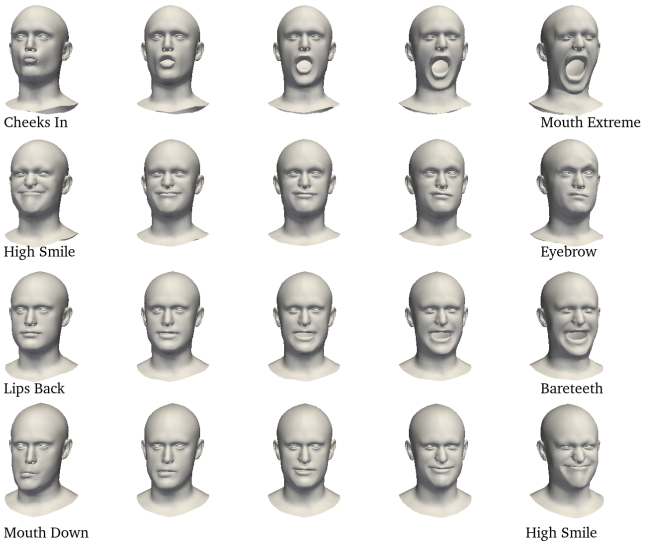

Figure 3: **Interpolation** on the COMA dataset using Ours-III network $d = 16$. We visualize results for $a = [0, 0.25, 0.5, 0.75, 1]$. Our model is able to generate realistic new samples from the distribution.

C.2. 3D Correspondence Adaptive Receptive Fields

We extend Sec. 5.2 of the main paper by providing additional visualization of the receptive field learned by our mesh operator in Figure 4. We show the length of the spirals estimated by the three layers in our network which include our operator. We illustrate spirals for various vertices in three different examples of the FAUST test set. In all cases, the dynamic receptive fields of the three layers are presented sequentially from left to right. As clearly visible, even in absence of any direct supervision, our mesh operator is capable of learning spirals of progressively longer length, hence demonstrating the importance of adapting the receptive field throughout the different layers of the network to

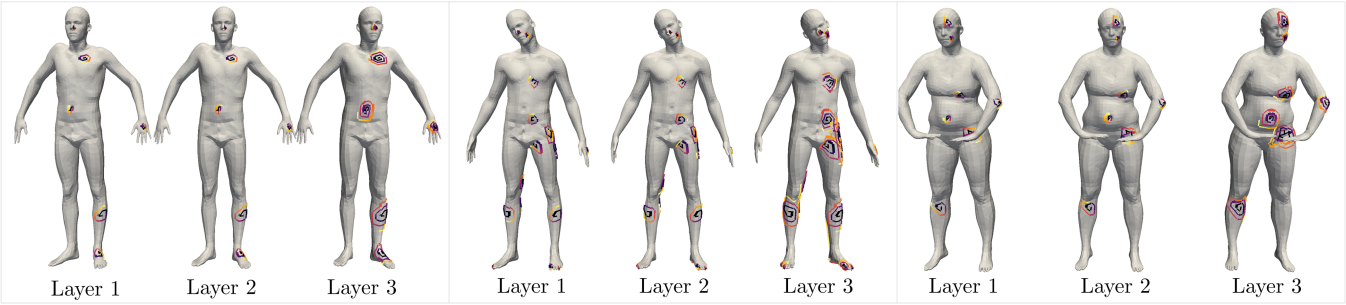

Figure 4: **Receptive field** predicted by our operator for three consecutive layers of the same network trained on the FAUST dataset for the task of 3D shape correspondence. Note that the architecture deploys no pooling layers. Our method predicts the length of the spiral for each mesh and vertex, progressively enlarging the length of the spiral to learn hierarchical representations.

build consistent hierarchical representation.

C.3. 3D Reconstruction Adaptive Receptive Fields

We extend Sec. 5.3 of the main paper providing additional visualization of the receptive field learned by our mesh operator in Figure 5. We illustrate the length of the spirals estimated by the last layer of our 3DMM for various vertices in six different examples of the COMA test set. In all cases, dynamic receptive fields from our method are presented on the right meshes, whereas fixed (K=9) receptive fields from SpiralConvolution are on the left ones. As clearly illustrated by these comparisons, our mesh operator is capable of learning spirals of markedly different lengths, hence demonstrating the importance of tailoring the receptive field to the local structure of the input mesh centered around the reference vertex. Interestingly, our method in some cases generates spirals of length similar to the baseline (*e.g.* $7, 12, 16, 9$), whereas in other cases it learns to exploit long-range dependencies by constructing much longer spirals (*e.g.* $45, 61, 77, 90$). Our implementation based on Summed-Area-Table makes the complexity of such dynamic spatial reasoning independent from the number of vertices used, hence allowing the proposed method to adapt to the underlying structure of the mesh, eventually producing higher-quality outputs at no additional cost in complexity.

C.4. 3D Reconstruction Error

Our method acts as a non-linear extension of the SpiralConvolution layer and has three key features: firstly, it is adaptable to the spatial characteristics of the mesh; secondly, it is dynamic in nature, meaning that its weight tensor reacts to each mesh in a distinctive manner; and thirdly, it can expand its receptive field through learnable pooling, which has the potential to cover all vertices in the mesh. In Sec. 5.3.2 of the main paper, we evaluate the impact of each characteristic on the performance of SpiralNet++, showcasing the contribution of each inductive bias on the final reconstruction loss. Here, we extend these experimental results with qualitative comparisons among different layers. Figure 6 shows the visualization of the per-vertex Euclidean error with respect to the ground truth for a SpiralNet++ baseline of 32 channels (first row) and four 3DMM created by adding layers with different inductive biases to SpiralNet++, as described in the main paper. As clearly visible from the figure, the use of W_k does not provide any particular advantage over the baseline, since the information contained in W_k is already included in the baseline tensor W_{kcd}. As expected, the other operations increasingly improve performance. Observing each example from the top row to the last row highlights the contribution of each new characteristic to the output generated by the 3DMM. The last row shows the results of the layer which integrates non-local reasoning, vertex-wise adaptivity (*i.e.*, spatially adaptive), and instance-wise adaptivity (*i.e.*, dynamic). The proposed method achieves the lowest reconstruction error, outperforming all other layers and the baseline configuration, demonstrating the importance of adaptive processing with dynamic receptive fields.

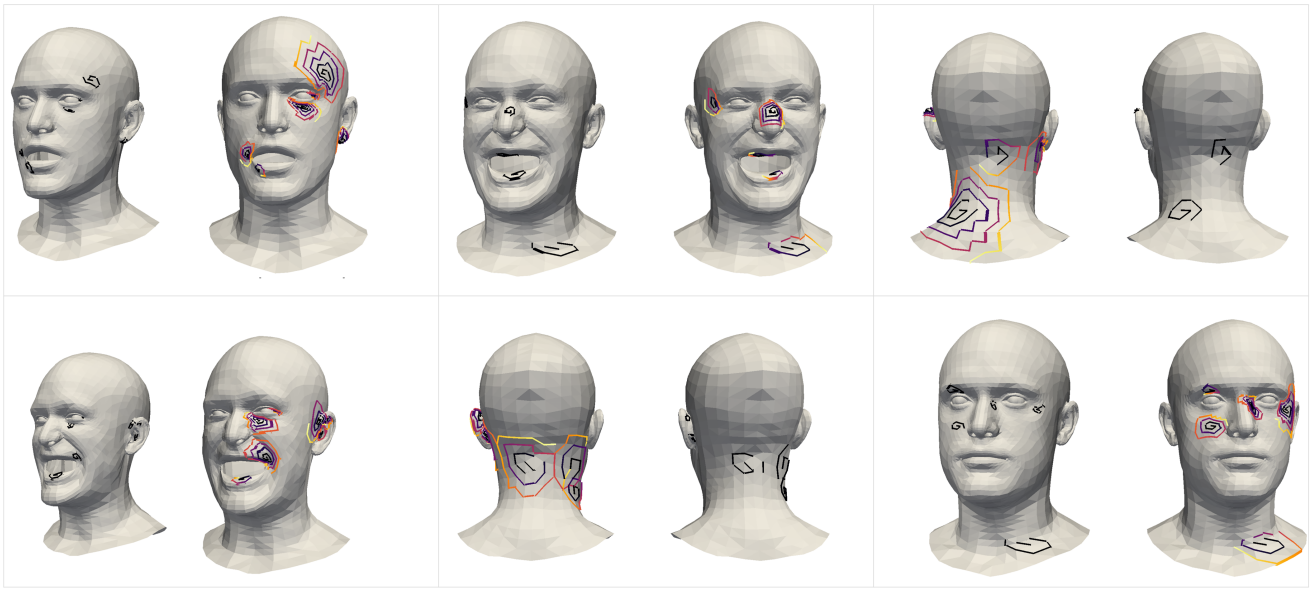

Figure 5: **Receptive field of SpiralConvolution (left) and our operator (right).** Our method predicts the length of the spiral for each mesh and vertex. Compared to a static receptive field of length 9, is capable to adapt its response, obtaining sequences ranging from length 3 to length 90. Due to its efficient implementation, the size of the spiral considered does not impact the complexity of the model.

Figure 6: **Qualitative comparison of 3DMM with different inductive biases.** Visualization of the per-vertex euclidean distance from ground truth (mm) for a SpiralNet++ baseline (ch=32) augmented with different layers. Errors are saturated at 10mm. Each layer incorporates a different characteristic in the model. Observing each example from the top row to the last row highlights the contribution of each new characteristic to the 3DMM. The last row shows how the integration of non-local reasoning, vertex-wise adaptivity (*i.e.* spatially-adaptive), and instance-wise adaptivity (*i.e.* dynamic) achieves the lowest reconstruction error.

References

[1] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3794–3801, 2014. 1

[2] Federica Bogo, Javier Romero, Gerard Pons-Moll, and Michael J Black. Dynamic faust: Registering human bodies in motion. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6233–6242, 2017. 2

[3] Giorgos Bouritsas, Sergiy Bokhnyak, Stylianos Ploumpis, Michael Bronstein, and Stefanos Zafeiriou. Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7213–7222, 2019. 1, 3

[4] Zhixiang Chen and Tae-Kyun Kim. Learning feature aggregation for deep 3d morphable models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13164–13173, 2021. 2

[5] Zhongpai Gao, Junchi Yan, Guangtao Zhai, Juyong Zhang, and Xiaokang Yang. Robust mesh representation learning via efficient local structure-aware anisotropic convolution. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 2

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010. 1

[7] Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. Spiralnet++: A fast and highly efficient mesh convolution operator. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*, pages 0–0, 2019. 1

[8] Ltd. Huawei Technologies Co. Huawei mindspore ai development framework. In *Artificial Intelligence Technology*, pages 137–162. Springer, 2022. 1

[9] Jameel Malik, Ahmed Elhayek, Fabrizio Nunnari, Kiran Varanasi, Kiarash Tamaddon, Alexis Heloir, and Didier Stricker. Deephps: End-to-end estimation of 3d hand pose and shape by learning from synthetic depth. In *2018 International Conference on 3D Vision (3DV)*, pages 110–119. IEEE, 2018. 2

[10] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015. 1

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 1

[12] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European conference on computer vision (ECCV)*, pages 704–720, 2018. 1

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 1

[14] Edgar Tretschk, Ayush Tewari, Michael Zollhöfer, Vladislav Golyanik, and Christian Theobalt. Demea: Deep mesh autoencoders for non-rigidly deforming objects. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 601–617. Springer, 2020. 2