ISSN 2464-4617 (print)
ISSN 2464-4625 (DVD)
Computer Science Research Notes
CSRN 3001
WSCG2020 Proceedings

Memory-Friendly Deep Mesh Registration

François Le Clerc
InterDigital
975 avenue des Champs Blancs
F-35576 Cesson-Sévigné, France
Francois.LeClerc@InterDigital.com

Hao Sun[1]
Tencent
1081C Hongmei Road
200233 Shanghai, China
mikiyasun@tencent.com

ABSTRACT

Processing 3D meshes using convolutional neural networks requires convolutions to operate on features sampled on non-Euclidean manifolds. To this purpose, spatial-domain approaches applicable to meshes with different topologies locally map feature values in vertex neighborhoods to Euclidean 'patches' that provide consistent inputs to the convolution filters around all mesh vertices. This generalization of the convolution operator significantly increases the memory footprint of convolutional layers and sets a practical limit to network depths on the available GPU hardware. We propose a memory-optimized convolution scheme that mitigates the issue and allows more convolutional layers to be included in a network for a given memory budget. The experimental evaluation of mesh registration accuracy on datasets of human face and body scans shows that deeper networks bring substantial performance improvements and demonstrate the benefits of our scheme. Our results outperform the state of art.

Keywords

Geometric deep learning, convolutional neural networks, shape matching, 3D mesh

1 INTRODUCTION

Mesh registration, aka shape matching, is a key stage of a 3D geometry processing pipeline that provides control on the sampling of vertices and brings all processed meshes to a common representation. Effective registration techniques based on non-rigid Iterative Closest Point approaches have been developed, but their computational cost makes them unsuitable for interactive processing. Besides, landmark annotations are needed to drive the convergence of these algorithms. Recent advances in machine learning and particularly deep learning offer promising prospects for improvements in the field. Inference on a deep network is fast thanks to the availability of high-performance GPU hardware, and learning rather than computing the registration removes the need for landmarking. The challenge for learning-based approaches is to maintain the high levels of accuracy achieved by computational geometry algorithms.

[1] Work done while the author was with Technicolor R&D, now InterDigital

Convolutional neural networks were originally applied to signals regularly sampled on a Euclidean domain, for which the implementation of discrete convolution is straightforward. For signals defined on non-Euclidean mesh surfaces, the convolution operator becomes a position-dependent filter [Bro17a]. When the processed meshes have different topologies, the feature values known at the locations of vertices must be locally mapped to Euclidean 'patches' to provide the inputs to the convolution filters. The patch extraction process incurs additional computations and increased memory requirements. In practice, the network depth is limited by the memory capacity of the available GPU hardware: network architectures proposed in [Mas15a, Mon17a, Ver18a] do not contain more than 3 convolutional layers.

In this paper we quantify the storage space needed for implementing convolutions on feature signals sampled on the surface of a 3D mesh. In addition to the Euclidean case requirements, local geometry information around each vertex must be fed into the network for extracting the patches, and storage must be allocated for intermediate tensors in the computation of convolutions. We propose mitigation schemes for both of these issues to reduce the memory footprint of convolutional layers. For a given memory budget, more of these layers can be squeezed into the network. The shape matching performance of our memory-optimized networks is evaluated on human body and human face mesh datasets with varying resolutions. The results show that the registration accuracy greatly benefits from in-

creased network depths. Our approach outperforms the state-of-art and achieves close to perfect registration scores on the FAUST dataset.

2 RELATED WORK

We focus our review of past research on approaches most related to our work, and in particular to registration algorithms that can deal with non-isometric deformations. For a comprehensive review of shape matching, we refer the reader to the survey in [Kai04a].

Non-rigid Iterative Closest Point

Early shape matching approaches [Sum04a, Amb07a] were extensions of the Iterative Closest Point method to non-rigid registration, where the deformation between a source and a target mesh is modelled as a set of local affine transforms. The registration is obtained by globally solving for the deformed source vertex positions, with regularization terms to ensure a spatially smooth mapping. Landmark correspondences, typically obtained by manual annotation, are required to initiate the process. [Zel13a] improves the resilience of these methods to large differences in the geometries of the source and target mesh by initiating the computation of the deformation on geometrically simplified meshes. All of these approaches require solving a bulky sparse linear system dimensioned by the number of vertices and faces in the meshes, making them unsuitable for interactive operation except on small meshes.

Functional Maps

Functional maps [Ovs12a] extend the notion of vertex-to-vertex correspondence to mappings between real-valued functions that encode descriptors of the geometry around each vertex. The mapping between functions is expressed as a linear operator on orthogonal bases taken to be the eigenfunctions of the Laplace-Beltrami operator. Solving a large linear system yields the coefficients of this linear operator. [Lit17a] improves on the original approach by leveraging a deep network to jointly compute the functional map correspondence and optimize the functional descriptors.

Learning descriptor-based correspondences

In early machine learning approaches, shape registration is cast as a classification problem and the correspondence is computed from local mesh geometry descriptors. In [Rod14a] correspondence labels for shape vertices are computed from the outputs of a random forest whose split functions are designed based on the Wave Kernel Signatures of the vertices. In [Wei16a] per-pixel descriptors on depth maps of human bodies are learnt as intermediate features of a shape classification network, and the correspondence problem is solved by a closest neighbor search in descriptor space.

Graph Convolutional Neural Networks

Deep learning approaches formulate mesh registration as a classification problem whose outputs for each source mesh vertex are the probabilities of assignment to each of the target mesh vertices. The generalization of Convolutional Neural Networks (CNNs) to signals sampled on non-Euclidean domains (in the most general setting, graphs) can be achieved by defining the convolution in the graph Fourier domain [Bro17a]. However, these spectral-domain approaches, *e.g.* [Kip17a], are inadequate for processing meshes with different topologies as the graph Fourier transform differs for each mesh. Approaches applicable to shape matching operate in the spatial domain by locally mapping feature values in a neighborhood of each vertex on the mesh surface to a Euclidean domain on which a convolution kernel can be defined consistently for all vertices. In [Mas15a, Bos16a, Mon17a] these mappings target local geodesic polar coordinate systems, while in [Ver18a] they are learnt from the convolutional layer inputs.

The adaptation of the convolution operator in spatial-domain graph-CNN approaches incurs additional computations and increased memory requirements that limit in practice the network depth. Based on the intuition that deeper networks could provide better performance, we propose a memory-optimized convolution scheme that allows more convolutional layers to be included in the network for the same memory budget.

3 TECHNICAL APPROACH

3.1 Problem and notations

We address the shape matching problem for input meshes with N vertices by means of a spatial-domain graph-CNN. For simplicity and without loss of generality, we assume minibatches to enclose the feature descriptors of all the vertices of a mesh. We consider a generic convolutional layer of the network with D input channels, E output channels and convolution kernels of size M. The learnable parameters for the layer are the convolution weights $w_m^{d,e}$ with $1 \leq d \leq D$, $1 \leq e \leq E$ and $1 \leq m \leq M$.

3.2 Patch operator

The convolution can be formulated using a "patch operator" [Mas15a, Bro17a] that interpolates a feature signal f sampled at vertex locations on the mesh surface to consistent inputs to the convolution operator, based on data available in local neighborhoods Γ_i of each vertex V_i:

$$D_m(V_i)f = \sum_{j \in \Gamma_i} c_m[\mathbf{u}(V_i, V_j)].f(V_j). \qquad (1)$$

In this equation, m indexes the values of the convolution kernel, $\mathbf{u}(V_i, V_j)$ is a local parameterization of the mesh

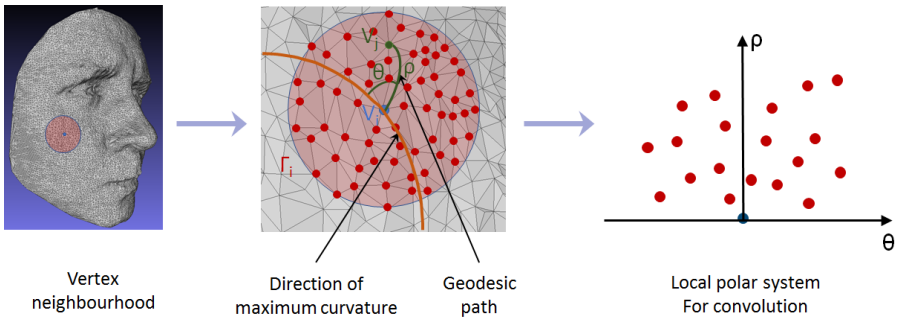


Figure 1: *the patch operator maps vertices in a vertex neighbourhood Γ_i to a Euclidean polar coordinate system*

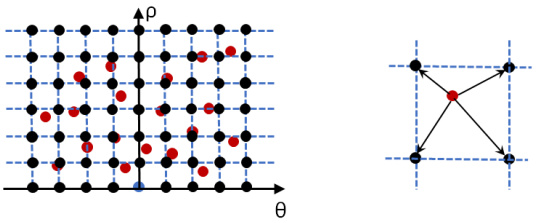


Figure 2: *the convolution kernel support is a uniformly sampled grid in the (ρ,θ) space (black dots). The feature value at each vertex neighbor V_j (red dots) contributes only to the 4 neighboring grid points.*

surface in Γ_i and the c_m weights define the patch operator. The Γ_i can be defined as geodesic, Euclidean or ring neighborhoods. Their size $K_i \stackrel{\text{def}}{=} |\Gamma_i|$ is a dimensioning parameter for the memory footprint of the convolutional layer. We denote by K the average of K_i over the mesh. Following [Mas15a, Mon17a] we define $\mathbf{u}(V_i,V_j) = (\rho_{ij},\theta_{ij})^T$ as local polar intrinsic coordinate systems where ρ_{ij} is the geodesic distance between V_i and V_j and θ_{ij} the angle between the geodesic path at V_i with a reference direction that we set to the maximum curvature direction at V_i [1] (see Figure 1). For notational simplicity we define $c_{ijm} \stackrel{\text{def}}{=} c_m[\mathbf{u}(V_i,V_j)]$ and obtain the e^{th} output component of the convolution as

$$g_i^e = \sum_{d=1}^{D} \sum_{j\in\Gamma_i} \sum_{m=1}^{M} c_{i,j,m} w_m^{d,e} f_j^d. \tag{2}$$

In [Mas15a, Mon17a] the patch operator relies on a mixture of Gaussian kernels :

$$c_m(\mathbf{u}) = exp[-\frac{1}{2}(\mathbf{u}-\mathbf{u}_m)^T \begin{pmatrix} \sigma_\rho^2 & 0 \\ 0 & \sigma_\theta^2 \end{pmatrix}^{-1} (\mathbf{u}-\mathbf{u}_m)] \tag{3}$$

with the difference that σ_ρ, σ_θ and $\mathbf{u}_m$ are fixed in [Mas15a] and learnt in [Mon17a]. [Ver18a] remove the hand-crafted parameterization $\mathbf{u}(V_i,V_j)$ of the mesh surface and directly learn the patch operator as a function of the feature inputs to the current layer:

$$c_m(f_i,f_j) \propto exp[\mathbf{a}^T(f_i-f_j)+\mathbf{b}_m]. \tag{4}$$

We opt for the baseline patch operator of [Mas15a] and define the support of the convolution kernel to be a regular grid of $N_\rho \times N_\theta$ points in the local polar geodesic systems around each vertex. To reduce training times we speed up the computation by bi-linearly interpolating the contribution of the feature value at each V_j to its 4 closest points on the grid (Figure 2). Subject to the corresponding restrictions on $|\rho_{ij}-\rho_m|$ and $|\theta_{ij}-\theta_m|$, letting $\Delta\rho$ and $\Delta\theta$ be the spacings between adjacent grid points along each axis, our patch operator is defined as

$$c_m((\rho_{ij},\theta_{ij})^T) = \left(1-\frac{|\rho_m-\rho_{ij}|}{\Delta\rho}\right)\left(1-\frac{|\theta_m-\theta_{ij}|}{\Delta\theta}\right) \tag{5}$$

We weight the contributions of vertex features $f(V_j)$ in to the patch operator (1) by a term that approximates the Voronoi area of V_j, thereby giving more importance to feature values that represent larger areas on the mesh surface. Specifically, each triangle a vertex belongs to contributes 1/3 of its area to the vertex weight. The weight value is accumulated over all the triangles. Experimentally we found that this refinement stabilizes the convergence of the training process and slightly improves the registration accuracy.

3.3 Memory optimization

Implementing CNNs on non-Euclidean domains using a patch operator (1) incurs extra complexity: a local parameterization $\mathbf{u}(V_i,V_j)$ of the mesh surface around each V_i is needed, and the convolution involves an extra summation level. We show below that this strongly impacts memory consumption.

To obtain orders of magnitude on the memory footprints of convolutional layers, we consider as a typical use case medium resolution meshes with N = 15000 vertices and an average vertex neighborhood size of K = 300 that corresponds to the optimum in our experiments on face meshes. We further assume values of M = 32,

[1] [Mas15a] does not define a reference direction but computes convolutions for N_θ possible rotations of the convolution filter and retains the maximum value as the output.

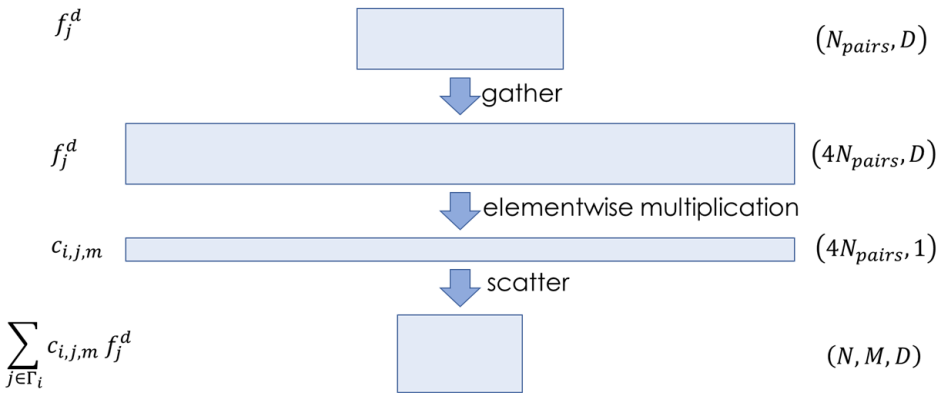


Figure 3: *optimized memory management of variable-sized vertex neighborhoods.*

$D = 128$, $E = 256$ that are typical of the experimental setups used in [Mas15a, Mon17a, Ver18a].

The patch operators used in [Mas15a, Mon17a] and in our approach depend on local polar coordinates $\mathbf{u}(V_i, V_j)$ around each V_i on the mesh surface. This information must be fed to the network for extracting the convolution patches around each vertex through the $c_m(\mathbf{u})$ coefficients and is used in each convolutional layer. These coefficients represent in total NKM floating point values for a minibatch of N vertices. In the approach of [Ver18a] the geometrical data needed to compute the patch operator are the indicator functions of neighborhoods Γ_i, NK integer vertex indices whose memory footprint is negligible.

Assuming a 4-byte floating point representation, NKM values amount to 0.58 GB of data in our numerical example. However, this estimation assumes neighborhoods of fixed size K. In practice K is an average and the count of vertices in each Γ_i varies as a function of the sampling density on the mesh surface. In our experiments the ratio K_{max}/K of max to average neighborhood vertex count lies between 2 and 4. We optimize the memory management of our patch operator coefficients $c_{i,j,m} \stackrel{\text{def}}{=} c_m(\mathbf{u})$ in two ways. First, we reduce their memory footprint from NKM to $4NK$ values by restricting the contribution of each feature at V_j to the 4 neighboring points on the convolution kernel grid, as explained in section 3.2 (see figure 2). Second, to avoid wasting space through padding because of the varying vertex neighborhood size, we map patch extraction coefficients to 1D tensors of $4N_{pairs}$ values, where N_{pairs} is the total number of (V_i, V_j) pairs on the mesh. Each tensor element represents one of the 4 patch extraction coefficients for one such pair. As illustrated on figure 3, the input feature values f_j^d are mapped to this representation using gather operations, and after multiplying the features elementwise with the $c_{i,j,m}$ the result is scattered to an (N, M, D) tensor. Note that the scatter operation accumulates the contributions of the (V_i, V_j) pairs at each patch location m and consequently performs the summation over j indices.

The computation of the convolution as defined by equation (2) involves a 3-level summation over tensors $\{c_{ijm}\}$ of shape (N, K, M), $\{w_m^{d,e}\}$ of shape (M, D, E) and $\{f_j^d\}$ of shape (N, D). In practice, the values of the f_j^d have to be duplicated for each Γ_i to compute the convolution, resulting in a tensor of shape (N, K, D). The 3 reduction operations in (2) must be performed one after the other[2], generating intermediate tensors among which the output of the first reduction is the largest. Thus, the ordering of the summations impacts memory usage. There are three options:

- over j first: $\sum_{j\in\Gamma_i} c_{i,j,m} f_j^d$ has shape (N, M, D)
- over m first: $\sum_{m=1}^{M} c_{i,j,m} w_m^{d,e}$ has shape (N, K, D, E)
- over d first: $\sum_{d=1}^{D} w_m^{d,e} f_j^d$ has shape (N, K, M, E)

Summing over j first allows considerable memory savings as it outputs an order 3 tensor while the other options generate intermediate tensors of order 4. A numerical evaluation on our typical use case illustrates how important the summation order is:

- j first: $N \times M \times D \approx 61.4\,10^6$, or 246 MB
- m first: $N \times K \times D \times E \approx 147\,10^9$, or 590 GB
- d first: $N \times K \times M \times E \approx 36.9\,10^9$, or 147 GB.

Using today's GPU hardware, any option other than summing over j first precludes the implementation of a large number of convolutional layers in mesh registration networks.

[2] Deep learning frameworks offer the possibility of combining several reduction operations using an Einstein summation, but experiments on TensorFlow and PyTorch show that the underlying implementations compute the sums sequentially without optimizing their order for memory consumption.

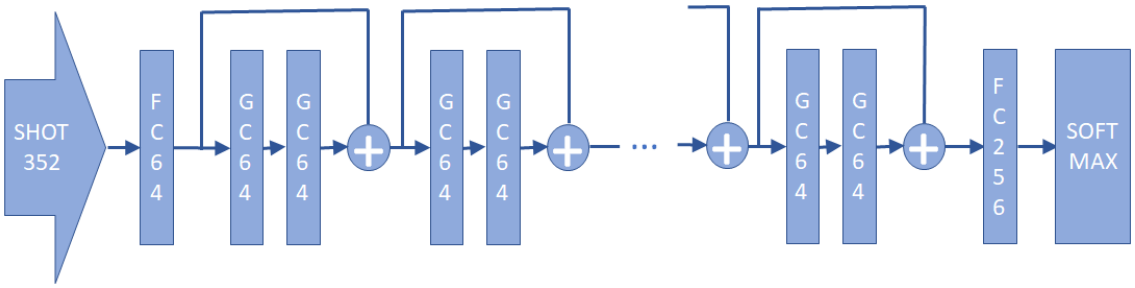


Figure 4: *architecture of our single-resolution deep mesh registration network.*

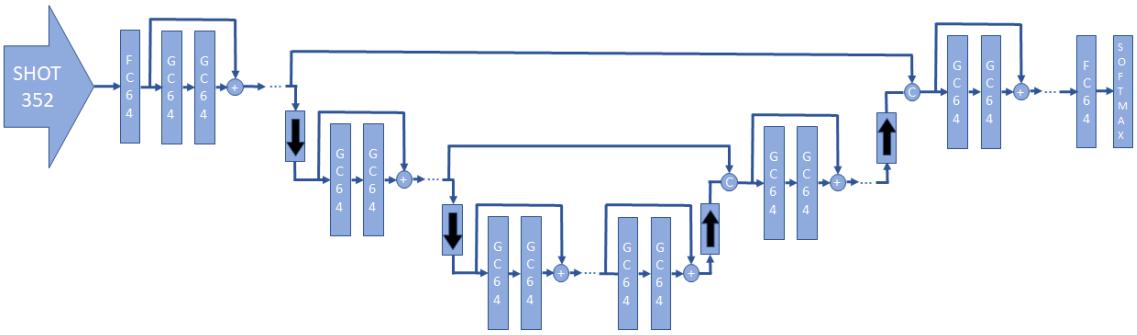


Figure 5: *architecture of our multi-resolution deep mesh registration network. Arrow blocks represent pooling and unpooling layers, C blocks concatenation.*

3.4 Baseline network architecture

The architecture of our baseline network is shown on figure 4. The network is fed with minibatches of SHOT descriptors [Sal14a] for all the vertices of the processed mesh. The registration task is cast as a labeling problem: for each source mesh vertex, the network outputs the probabilities of assignment to each target mesh vertex, and its correpondence is computed as the *argmax* of this vector. The network core consists of memory-optimized geodesic convolutional layers, as described above, each of depth 64. Each pair is set-up as a residual block [HeK16a] by-passed by a skip connection. The number of such blocks is maximized to fit into the available hardware memory.

3.5 Multi-resolution network architecture

In a spatial graph-CNN each vertex is processed independently from its neighbors at inference time. A multi-resolution network as proposed in [Ver18a] introduces some amount of spatial regularization on the outputs, and was shown to improve mesh registration performance on the FAUST dataset. We base our multi-resolution network on the same U-Net architecture but replace the convolutional layers by residual blocks of our memory-optimized layers (figure 5), resulting in a deeper netwrk.

The pooling approach in [Ver18a] is generic for all types of graphs. Following [Ran18a], we specialize it to benefit from the spatial vertex layout information available on a mesh. Pooling in our multi-resolution network relies on the edge collapsing scheme of [Gar97a]. Collapsing an edge maps its two end vertices to one vertex in the decimated mesh, thereby halving the number of vertices. We choose as decimated vertex the edge end that has the smaller collapsing cost.

A pooling layer in our network generates a feature value for each decimated mesh vertex by copying the feature value of the corresponding vertex in the original mesh. Letting N_{orig} be the vertex count of the original mesh, a pooling layer is implemented as a fixed sparse $N_{orig}/2 \times N_{orig}$ matrix that is precomputed from the mesh decimation results. Each row has a unique non-zero value equal to 1 at the location of the collapsed vertex index in the original mesh.

In the same way, the unpooling layer is implemented as a precomputed sparse $N_{orig} \times N_{orig}/2$ matrix. Each row represents a vertex in the upsampled mesh and holds a unique non-zero value equal to 1 at the location of the corresponding vertex in the decimated mesh, computed as the closest vertex.

4 EXPERIMENTAL DATASETS

We validate our approach on two complementary datasets with very different features.

4.1 FAUST [Bog14a]

We experiment with the low-resolution version of the MPI FAUST dataset, built from 100 human body scans with varying poses and morphology. The meshes have 6890 vertices each and are completely free of geometrical artefacts. All meshes have been pre-registered using

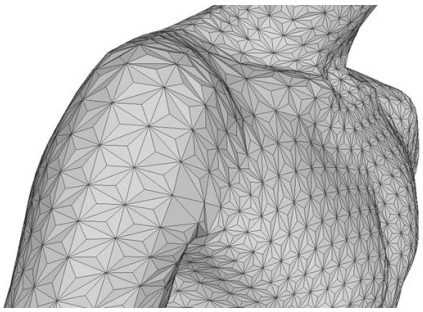


Figure 6: *example FAUST mesh after up-sampling to 15K vertices (detail)*

landmark annotations on the bodies to provide accurate ground-truth correspondence. We follow the protocol of [Mon17a] and split the dataset into 80 meshes for training and 20 for testing. The first mesh in the training dataset is used as the registration target.

To experiment with memory limitations, we generate a higher-resolution version FAUST-15K of the dataset by upsampling the meshes to 15000 vertices (Figure 6). To this purpose we fit a cubic surface patch around each vertex following the algorithm of [Gol04a] and interpolate on this patch the locations of the centroids on each triangle adjacent to the vertex. The centroid for each triangle is computed as the average of these locations. We add centroids to a reference original mesh in the FAUST dataset up to the desired target vertex count, in decreasing order of triangle area. We then select the same set of centroids for all other meshes in order to preserve the ground-truth per-vertex correspondence of the original dataset.

4.2 Bosphorus [Sav08a]

We also validate our approach on Bosphorus, a dataset of higher-resolution, noisy expressive facial scans that is more representative of the 3D data processed in the industry. The scans come as point clouds with holes and sometimes noticeable artifacts, which we pre-process as shown on Figure 7. After removing background points by depth thresholding, the point clouds are meshed using 2D Delaunay triangulation in a plane perpendicular to the scanning axis (Figure 7b). Next, we fill holes and remove small connected components (Figure 7c). Finally, we apply quadric edge decimation to downsample all meshes to 15K vertices (Figure 7d). The target mesh is set to a neutral expression scan that belongs neither to the training set nor to the test set, and is pre-processed in the same way. Ground truth correspondences to this target are obtained using the deformation transfer algorithm of [Sum04a], leveraging the landmark annotations provided with the Bosphorus dataset. We validate the excellent registration quality of this method through high-resolution texture transfer. 150 scans are selected for training and 50 for testing.

5 EXPERIMENTAL RESULTS

5.1 Methodology

We assess the performance of our registration approach using the Princeton benchmark protocol [Kim11a], and compare it with two recent approaches [Mon17a, Ver18a] for which the authors provide their code. The registration accuracy is plotted as the proportion of correct correspondences over the test set (vertical axis) as a function of the tolerance on the geodesic error (horizontal axis), expressed as a ratio of the mesh diameter. To allow for a fair comparison we optimize the network and training parameters separately for all approaches given a fixed GPU memory budget. We run all our experiments on an NVidia Tesla P100 GPU with 16 GB memory. We use as network inputs for all datasets 352-dimensional SHOT descriptors [Sal14a]. Minibatches consist of the descriptors for all the vertices of a mesh, except for [Mon17a] where they are restricted to mesh regions to avoid running out of memory. In this case the processing of sub-meshes is managed in the authors code.

Besides the number of convolutional layers in the network, the parameters most impacting the memory requirements are the geometrical extent ρ_{max} of the vertex neighborhoods and the size M of the convolution kernel. In our approach, kernels are sampled on 8 by 8 regular grids in the local geodesic polar systems around each vertex, hence M is set to 64. For [Mon17a, Ver18a] we use the values of M advocated in the papers. For each approach we optimize the remaining parameters by performing a grid search, selecting the values that yield the best registration accuracy while not running out of memory. These parameters are ρ_{max} for [Ver18a], ρ_{max} and the count of vertices in each minibatch for [Mon17a], and for our approach ρ_{max} and the number of convolutional layers.

The network is trained by minimizing the standard cross-entropy loss by means of an Adam optimizer, with an initial learning rate of 10^{-4} and over 800 epochs. We use batch normalization but no dropout.

5.2 Single-resolution performance

Figure 8 shows the compared accuracy of our approach against MoNet [Mon17a] and the single resolution version of FeaStNet [Ver18a]. For completeness we also plot the performance curves of older shape matching approaches on FAUST, reproduced from [Lit17a] and [Mas15a].

Our scheme outperforms all previous single-resolution approaches on the 3 considered datasets. Unsurprisingly, the accuracy on Bosphorus for all methods is considerably less than on FAUST, a consequence of the large amount of geometrical noise of this dataset. However, for the same quality of geometry (FAUST and

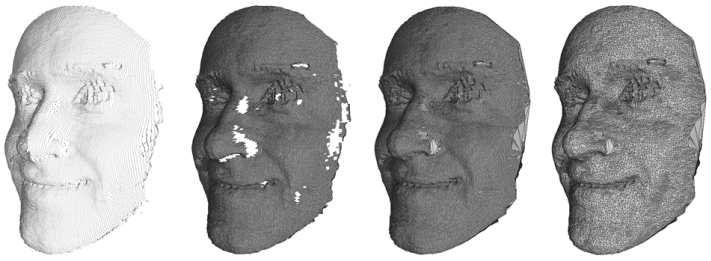


Figure 7: *Pre-processing pipeline of the Bosphorus dataset point clouds*

Algo - Database	Num conv. layers	ρ_{max}	M	Batch size
[Mon17a] - FAUST	3	0.60 IOD	80	75 vertices
[Mon17a] - FAUST 15K	3	0.30 IOD	80	150 vertices
[Mon17a] - Bosphorus	3	0.06 IOD	80	150 vertices
[Ver18a] - FAUST	3	1-ring	32	1 mesh
[Ver18a] - FAUST 15K	3	1-ring	32	1 mesh
[Ver18a] - Bosphorus	3	1-ring	32	1 mesh
Ours single-res - FAUST	30	1.20 IOD	64	1 mesh
Ours single-res - FAUST 15K	30	1.20 IOD	64	1 mesh
Ours single-res - Bosphorus	14	0.25 IOD	64	1 mesh
Ours 2 res levels - FAUST	10+10+10	1.20 IOD	64	1 mesh
Ours 2 res levels - FAUST 15K	10+10+10	0.90 IOD	64	1 mesh
Ours 2 res levels - Bosphorus	10+10+10	0.18 IOD	64	1 mesh

Table 1: *Optimal experimental settings of memory-dimensioning parameters (IOD: Inter-Ocular Distance)*

FAUST-15K), our approach maintains its performance level when the vertex count is increased, while the accuracy of [Mon17a] and [Ver18a] drops significantly.

Figure 9 a) and b) demonstrates the improvement of the registration accuracy as more convolutional layers are added to the network. On Bosphorus performance keeps improving up to the depth limit set by the hardware memory capacity.

5.3 Multi-resolution performance

Table 2 summarizes the performance of our multi-resolution network on the FAUST and FAUST-15K datasets. Our architecture has 2 resolution levels and 10 convolutional layers in each of the sections delimited by the pooling and unpooling layers. Adding more layers again improves performance, but the gain becomes less significant as our registration scores approach perfection. On FAUST our performance is superior to the results reported in [Ver18a], the only point of comparison in this case.

On Bosphorus the multi-resolution accuracy improves with network depth but remains lower than using a single-resolution network (Figure 9 c). We hypothesize that the noisier geometry on this dataset hinders the convergence of the network towards stable vertex correspondence patterns and thereby cancels the benefits of the multi-resolution analysis of the meshes.

Database	[Ver18a]	Ours(10+10+10)
FAUST	0.9860	**0.99997**
FAUST 15K	N/A	**0.9984**

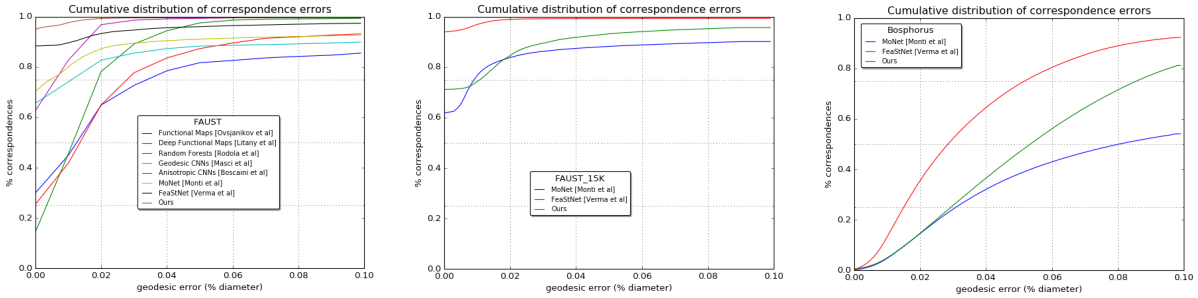
Table 2: *multi-resolution registration accuracy at zero geodesic error for [Ver18a] and the optimal configuration of our multi-resolution network)*

5.4 Qualitative experimental results

Figure 11 shows qualitative registration results on samples from the FAUST and Bosphorus datasets. Geodesic registration errors are translated into cm assuming an interocular distance of 6.3 cm.

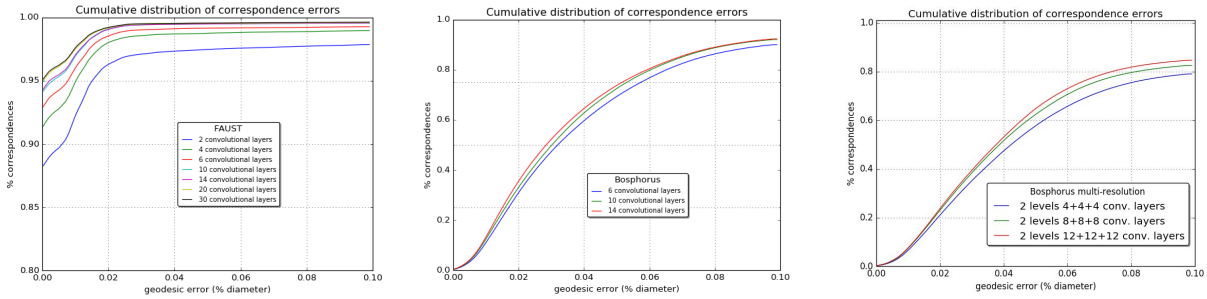
On FAUST most of the registration errors are localized on the torso, which lacks distinctive geometrical features for shape matching. For the same reason, the largest registration errors on Bosphorus occur on the peripheral areas of the meshes, away from the main facial features. The approach of [Ver18a] tends to map neighboring source vertices to the same target location, creating colocated vertices that give the morphed mesh a lower-resolution appearance.

The morphs of the source meshes to the geometry of the target mesh on Figure 11 should be interpreted with care. Most conspicuous in these morphs are the large



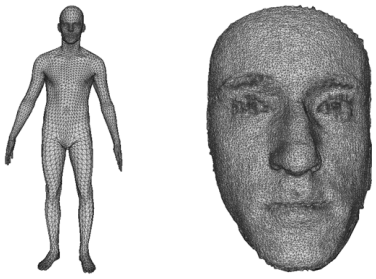
(a) (b) (c)

Figure 8: *registration accuracy benchmark on FAUST (a), FAUST 15K (b) and Bosphorus (c) datasets*



(a) (b) (c)

Figure 9: *influence of the number of convolutional layers on registration accuracy*
(a) Faust single-resolution, (b) Bosphorus single-resolution, (c) Bosphorus multi-resolution



FAUST Bosphorus

Figure 10: *FAUST and Bosphorus registration targets.*

triangles resulting from often isolated wrong vertex correspondences. It should be borne in mind that such triangles account only for a small proportion of the registration errors, as evidenced by the color encoding representations.

6 CONCLUSION

When processing 3D meshes using deep networks, increased memory requirements resulting from the generalization of convolution operators to non-Euclidean domains set a limit to the number of convolutional layers that can be placed in the network. The memory-friendly convolution scheme we propose mitigates this issue and allows deeper networks to be implemented for a given memory budget. On the mesh registration task we showed that deeper network architectures result in substantial performance gains.

Using a multi-resolution network featuring pooling and unpooling operators specialized for 3D mesh processing, we obtain close to perfect registration scores on the FAUST dataset. However, on noisy Bosphorus meshes the registration quality leaves much room for improvement. We believe FAUST is a toy dataset that has been useful for benchmarking mesh registration results so far, but should now been abandoned in favor of collections of meshes that are more representative of the noisy and higher-resolution scans used in the industry. In this respect, Bosphorus is arguably a worst case. The progress of 3D capture technology and photogrammetry in recent years should allow the constitution of datasets of better geometrical quality.

Further issues remain with the proposed registration scheme. First, like all competing approaches it takes as inputs meshes with different topologies and registers them to a common template. In practical use cases the reverse correspondence is needed to transfer the reference topology in a source mesh to a set of captured scans. Second, the local intrinsic mapping of vertex neighborhoods on the mesh surface to Euclidean convolution patches incurs heavy pre-processing as geodesic paths need to be computed in every vertex neighborhood. Simplifying this mapping would be desirable in order to better benefit from the efficiency of deep learning inference on GPU hardware.

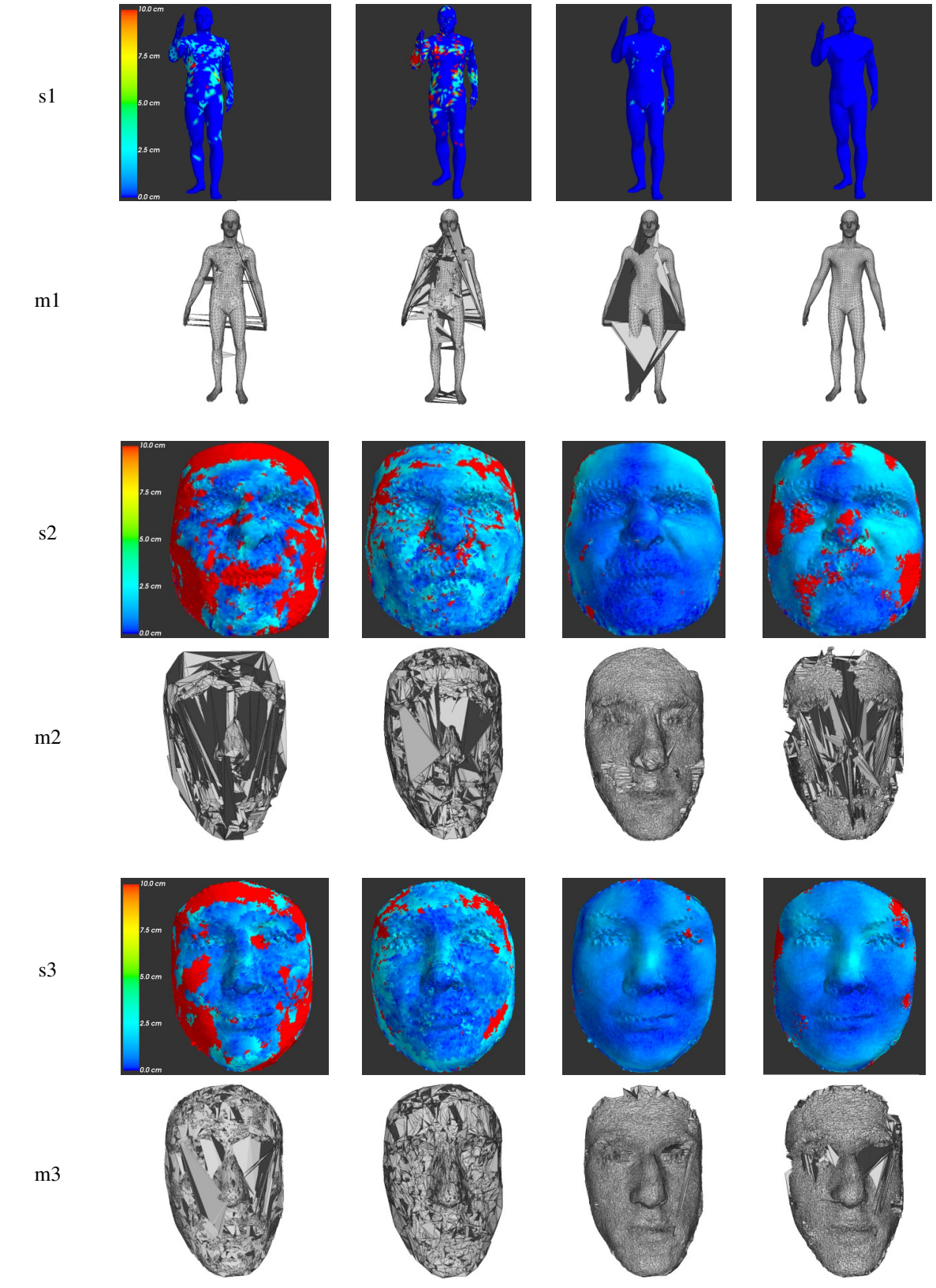


Figure 11: *per-vertex registration errors for FAUST (s1) and Bosphorus (s2, s3) samples. Rows s1, s2, s3 display a color encoding of the geodesic correspondence errors on the source meshes. Rows m1, m2, m3 show the morphs of the source meshes to the target, based on correspondence results. Best viewed in color.*

7 REFERENCES

[Amb07a] Amberg, B., Romdhani, S. and Vetter, T., Optimal Step Nonrigid ICP Algorithms for Surface Registration, in Conf. Proc. CVPR'07, 2007.

[Bog14a] Bogo, F., Romero, J., Loper, M., and Black, M.; FAUST: dataset and evaluation for 3D mesh registration, in Conf. Proc. CVPR'14, 2014.

[Bos16a] Boscaini, D., Masci, J., Rodola, E., and Bronstein, M., Learning shape correspondence with anisotropic convolutional neural networks, in Conf. Proc. NIPS'16, 2016.

[Bro17a] Bronstein, M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P., Geometric deep learning: going beyond Euclidean data, IEEE Sig. Proc. Magazine, vol. 34, no. 4, pp. 18-42, 2017.

[Gar97a] Garland, M., and Heckbert, P.S., Surface Simplification Using Quadric Error Metrics, in Conf. Proc. SIGGRAPH'97, 1997.

[Gol04a] Goldfeather, J., and Interrante, V., A Novel Cubic-Order Algorithm for Approximating Principal Direction Vectors, ACM Trans. on Graphics Vol.23, No.1, pp.45-63, 2004.

[HeK16a] He, K., Zhang, X., Ren, S., and Sun, J., Deep Residual Learning for Image Recognition, in Conf. Proc. CVPR'16, 2016.

[Kai04a] van Kaick, O., Zhang, H., Hamarneh G., and Cohen-Or D., A Survey on Shape Correspondence, Computer Graphics Forum, Vol.30, No.6, p.1681-1707, 2011.

[Kim11a] Kim, V., Lipman, Y., and Funkhouser, T., Blended Intrinsic Maps, ACM Trans. on Graphics Vol.30, No.4, pp.79:1-79:12, 2011.

[Kip17a] Kipf, T., and Welling, M., Semi-Supervised Classification with Graph Convolution Networks, in Conf. Proc. ICLR'17, 2017.

[Lit17a] Litany, O., Remez, T., Rodola, T. and Bronstein, A., Deep Functional Maps: Structured Prediction for Dense Shape Correspondence, in Conf. Proc. ICCV'17, 2017.

[Mas15a] Masci, J., Boscaini, D., Bronstein, M., and Vandergheynst, P., Geodesic Convolutional Neural Networks on Riemaniann Manifolds, in Conf. Proc. ICCV'15 Workshop, 2015.

[Mon17a] Monti, P., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M., Geometric deep learning on graphs and manifolds using mixture model CNNs, in Conf. Proc. CVPR'17, 2017.

[Ovs12a] Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A. and Guibas, L., Functional Maps: a Flexible Representation of Maps between Shapes, ACM Trans. on Graphics, Vol.31, No.4, 2012.

[Ran18a] Ranjan, A., Bolkart, T., Sanyal, S., and Black, M.J., Generating 3D Faces using Convolutional Mesh Autoencoders, in Conf. Proc. ECCV'18, 2018.

[Rod14a] Rodolà, E., Bulo, S., Windheuser, T., Vestner, M. and Cremers, D., Dense Non-rigid Shape Correspondence using Random Forests, in Conf. Proc. CVPR'14, 2014.

[Sal14a] Salti, S., Tombari, F., and Di Stefano, L., SHOT: Unique Signatures of Histograms for Surface and Texture Description, Comp. Vis. and Im. Underst., Vol 125, pp.251-264, 2014.

[Sav08a] Savran, N., Alyüz, N., Dibeklioglu, H., Celiktutan, O., Gökberk, B., Sankur, B., and Akarun, L., Bosphorus Database for 3D Face Analysis, in Conf. Proc. COST 2101 Workshop on Biometrics and Identity Management (BIOID 2008), 2008.

[Sum04a] Sumner, R., and Popovic, J., Deformation Transfer for Triangle Meshes, ACM Trans. on Graphics Vol.23, No.3, pp.339-405, 2004.

[Ver18a] Verma, N., Boyer, E., and Verbeek, J., FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis, in Conf. Proc. CVPR'18, 2018.

[Wei16a] Wei, L., Huang, Q., Ceylan, D., Vouga E., and Li, H., Dense Human Body Correspondences using Convolutional Networks, in Conf. Proc. CVPR'16, 2016.

[Zel13a] Zell E., and Botsch, M., ElastiFace: Matching and Blending Texture Faces, in Conf. Proc. ACM Symposium on Non-Photorealistic Animation and Rendering, 2013.