A Simple Approach to Intrinsic Correspondence Learning on Unstructured 3D Meshes

Isaak Lim[1], Alexander Dielen[1], Marcel Campen[2], and Leif Kobbelt[1]

[1] Visual Computing Institute, RWTH Aachen University
[2] Osnabrück University

Abstract. The question of representation of 3D geometry is of vital importance when it comes to leveraging the recent advances in the field of machine learning for geometry processing tasks. For common unstructured surface meshes state-of-the-art methods rely on patch-based or mapping-based techniques that introduce resampling operations in order to encode neighborhood information in a structured and regular manner. We investigate whether such resampling can be avoided, and propose a simple and direct encoding approach. It does not only increase processing efficiency due to its simplicity – its direct nature also avoids any loss in data fidelity. To evaluate the proposed method, we perform a number of experiments in the challenging domain of intrinsic, non-rigid shape correspondence estimation. In comparisons to current methods we observe that our approach is able to achieve highly competitive results.

Keywords: shape correspondence estimation, learning on graphs

1 Introduction

The representation of 3D geometry is a key issue in the context of machine learning in general and deep learning in particular. A variety of approaches, from point clouds over voxel sets to range images, have been investigated. When the input geometry is in the common form of a surface mesh, conversion to such representations typically comes with losses in fidelity, accuracy, or conciseness. Hence, techniques have been introduced to more or less directly take such discrete surface data as input to machine learning methods. Examples are graph-based [13,4] and patch-based approaches [17,3,18]. While graph-based techniques rely on fixed mesh connectivity structures, patch-based techniques provide more flexibility. However, they crucially rely on some form of (re)sampling of the input mesh data, so as to achieve consistent, regular neighborhood encodings, similar to the regular pixel structures exploited for learning on image data.

In this paper we consider the question whether such resampling can be avoided, taking the mesh data as input even more directly. The rationale for our interest is twofold: the avoidance of resampling would increase the efficiency of inference (and perhaps training) and could possibly increase precision. The increase in efficiency would be due to not having to perform the (typically non-trivial) resampling (either as a preprocess or online). One could hypothesize an

increase in precision based on the fact that resampling is, in general, accompanied by some loss of data fidelity.

We propose a resampling and conversion free input encoding strategy for local neighborhoods in manifold 3D surface meshes. In contrast to many previous approaches for learning on surface meshes, we then make use of RNNs and fully-connected networks instead of CNNs, so as to be able to deal with the non-uniform, non-regular structure of the input. Though simple, this raw input encoding is rich enough that our networks could, in theory, learn to emulate common patch resampling operators based on it. Nevertheless, hand-crafting such resampling operators and preprocessing the input accordingly, as previously done, could of course be of benefit in practice. Hence it is important to evaluate practical performance experimentally.

We apply and benchmark our technique in the context of *non-rigid shape correspondence estimation* [29]. The computation of such point-to-point (or shape) correspondences is of interest for a variety of downstream shape analysis and processing tasks (e.g. shape interpolation, texture transfer, etc.). The inference of these correspondences, however, is a challenging task and topic of ongoing investigation. Our experiments in this context reveal that the preprocessing efforts can indeed be cut down significantly by our approach without sacrificing precision. In certain scenarios, as hypothesized, precision can even be increased relative to previous resampling-based techniques.

Contribution In this work we propose and investigate a novel form of using either fully-connected layers or LSTMs (Hochreiter and Schmidhuber [9]) for point-to-point correspondence learning on manifold 3D meshes. By serializing the local neighborhood of vertices we are able to encode relevant information in a straightforward manner and with very little preprocessing. We experimentally analyze the practical behavior and find that our approach achieves competitive results and outperforms a number of current methods in the task of shape correspondence prediction.

2 Related Work

Several data- and model-driven approaches for finding correspondences between shapes have been proposed in previous works.

Functional Maps Ovsjanikov et al. [23] approach the problem of finding point-to-point correspondences by formulating a function correspondence problem. They introduce functional maps as a compact representation that can be used for point-to-point maps. Various (model- and data-driven) improvements have been suggested [14,24,10,5,6,25,22,21,8]. Most closely related to our approach, Litany et al. [15] use deep metric learning to optimize input descriptors for the functional maps framework. However, point-to-point correspondence inference in all cases requires the computation of a functional map for each pair of shapes. This possibly costly computation can be avoided with our approach. Once trained, our model can be applied directly for inference.

Generalized CNNs for 3D Meshes Several data-driven methods that do not rely on functional maps were proposed in recent years. Masci et al. [17] generalize convolution operations in modern deep learning architectures to non-Euclidean domains. To this end they define geodesic disks (patches) around each vertex. Based on a local polar coordinate system the patches can be resampled with a fixed number and fixed pattern of samples (cf. Figure 1a). This predefined sampling pattern allows to construct a convolution operation on these patches by computing weighted sums of features at sample positions. In order to transfer the information (i.e. descriptors) available discretely at the vertices to the continuous setting of the geodesic disks for the purpose of resampling, they are blended by means of appropriate kernels. Boscaini et al. [3] propose to use anisotropic kernels in this context, while aligning the local coordinate systems with the principal curvature directions. Monti et al. [18] generalize the construction of these blending kernels to Gaussian Mixture Models, which avoids the hand-crafting of kernels in favor of learning them.

Ezuz et al. [7] and Maron et al. [16] both propose forms of global (instead of local patch-wise) structured resampling of the surface, which can then be used as input to well-known CNN architectures used in computer vision.

Similar in spirit to our work is the method introduced by Kostrikov et al. [13]. They apply Graph Neural Networks (cf. [27,4,20]) in the domain of 3D meshes. A key difference is that their network's layers see neighborhood information in reduced blended form (via Laplace or Dirac operators) rather than natively like our approach.

In comparison to these approaches we require very little preprocessing, no heavy online computation, and no resampling. Per-vertex descriptors are exploited directly rather than taking blended versions of them as input.

3 Resampling-free Neighborhood Encoding

We assume that the input domain is represented as a manifold triangle mesh $\mathcal{M}$. Some form of input data (e.g. positions, normals, or geometry descriptors) is specified or can be computed at the vertices of $\mathcal{M}$. We denote the information (*feature*) at a vertex v by $\mathrm{f}(v)$. As in previous work [17,3,18], for the task of correspondence estimation, we would like to collect this information f from a local neighborhood around a vertex a. As mentioned above, we intend to encode this relevant information in a very direct manner, essentially by a notion of serialization of the per-vertex features f in local neighborhoods, without any alterations.

3.1 Spiral Operator

To this end we make the observation that, given a center vertex, the surrounding vertices can quite naturally be enumerated by intuitively following a spiral, as illustrated in Figure 1b. The only degrees of freedom are the orientation (clockwise or counter-clockwise) and the choice of 1-ring vertex marking the spiral's

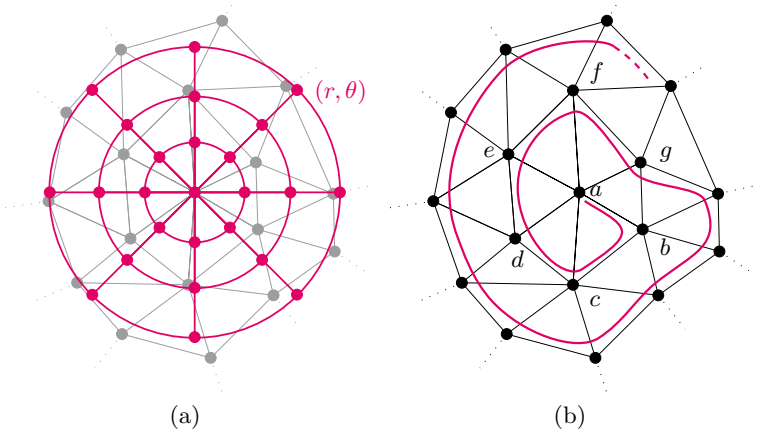


Fig. 1. The black graph represents a patch of a triangle mesh. (a) For generalized CNNs on 3D meshes [17,3,18], we would have to compute a blended $\mathrm{f}(r, \theta)$ for each node of the magenta polar grid in order to provide a fixed number and pattern of samples for a convolution kernel. (b) Instead, we enumerate the neighborhood vertices of a center vertex a by following a spiral pattern (magenta). For a given feature $\mathrm{f}(\cdot)$ we encode the local neighborhood information feeding $[\mathrm{f}(a), \mathrm{f}(b), \mathrm{f}(c), \mathrm{f}(d), \mathrm{f}(e), \mathrm{f}(f), \mathrm{f}(g), \ldots]$ into a LSTM Cell.

starting direction. We fix the orientation to clockwise here. The choice of starting direction is arbitrary, and a different sequence of vertices will be produced by the spiral operator depending on this choice. This rotational ambiguity is a common issue in this context, and has been dealt with, for instance, by max-pooling over multiple choices [17], or by making the choice based on additional, e.g. extrinsic, information [3]. We avoid this by instead making a random choice in each iteration during training, enabling the network to learn to be robust against this ambiguity, assuming a sufficient number of parameters in the network.

Given a starting direction (i.e. a chosen 1-ring vertex), the spiral operator produces a sequence enumerating the center vertex, followed by the 1-ring vertices, followed by the 2-ring vertices, and so forth. Thus, for a given k, it is possible to trace the spiral until we have enumerated all vertices up to and including the k-ring. In Figure 1b this is illustrated for the case $k = 2$, where the sequence reads $[a, b, c, d, e, f, g, \ldots]$. Alternatively, for a given N, we can of course trace until we have enumerated exactly N vertices, thereby producing fixed length sequences – in contrast to the variable length sequences up to ring k.

While the definition and practical enumeration of a spiral's vertices is really simple locally, some care must be taken to support the general setting, in particular with large k or large N (when k-rings are not necessarily simple loops anymore) or on meshes with boundary (where k-rings can be partial, maybe consisting of multiple components). The following concise definition of the spiral operator handles also such cases.

Let k-ring and k-disk be defined as follows:

$$\begin{aligned} 0\text{-ring}(v) &= \{v\}, \\ (k{+}1)\text{-ring}(v) &= N(k\text{-ring}(v)) \setminus k\text{-disk}(v), \\ k\text{-disk}(v) &= \cup_{i=0\ldots k}\, i\text{-ring}(v), \end{aligned}$$

where $N(V)$ is the set of all vertices adjacent to any vertex in set V.

The spiral(v, k) is defined simply as the concatenation of the *ordered* rings:

$$\text{spiral}(v, k) = (0\text{-ring}(v) \ \ldots \ k\text{-ring}(v)).$$

The fixed-length spiral(v, N) is obtained by truncation to a total of N vertices.

The required order $<$ on the vertices of a k-ring is defined as follows: The 1-ring vertices are ordered clockwise, starting at a random position. The ordering of the $(k{+}1)$-ring vertices is induced by their k-ring neighbors in the sense that vertices v_1 and v_2 in the $(k{+}1)$-ring being adjacent to a common vertex v^* in the k-ring are ordered clockwise around v^*, while vertices v_1 and v_2 having no common k-ring neighbor are sorted in the same order as (any of) their k-ring neighbors.

3.2 Learning

With the (either variable length or fixed length) vertex sequence $[a, b, c, d, e, f, g, \ldots]$ produced for a given center vertex, one easily serializes the neighborhood features as the sequence $[\mathrm{f}(a), \mathrm{f}(b), \mathrm{f}(c), \mathrm{f}(d), \mathrm{f}(e), \mathrm{f}(f), \mathrm{f}(g), ...]$.

For the purpose of correspondence estimation our goal is to learn a compact high-level representation of these sequences. This can be done in a straightforward and intuitive way using recurrent neural networks. More specifically, we feed our vertex sequences into an LSTM cell as proposed by Hochreiter and Schmidhuber [9] and use the last cell output as representation. This representation is thus computed using the following equations:

$$\begin{aligned} f_t &= \sigma(W_f \cdot [x_t, h_{t-1}] + b_f), \\ i_t &= \sigma(W_i \cdot [x_t, h_{t-1}] + b_i), \\ o_t &= \sigma(W_o \cdot [x_t, h_{t-1}] + b_o), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_c \cdot [x_t, h_{t-1}] + b_c), \\ h_t &= o_t \odot \tanh(c_t), \end{aligned}$$

where the learnable parameters are the matrices W_f, W_i, W_o, W_c with their respective biases b_f, b_i, b_o, b_c. $[x_t, h_{t-1}]$ is the concatenation of the input x_t (e.g. $\mathrm{f}(a)$) and the previous hidden state h_{t-1}, while c_t and h_t are the current cell- and hidden-state respectively. We denote the Hadamard product as $\odot$.

This generation of a representation of the local neighborhood of a vertex via a LSTM cell is, in an abstract sense, comparable to the generalized convolution operation of previous patch-based approaches. However, the resampling of

neighborhoods and computation of blended features $\mathrm{f}(r,\theta)$ for each sample (r,θ) (see Figure 1a) is avoided by our approach. Here r and θ are geodesic polar coordinates of some local coordinate system located at each center vertex. $\mathrm{f}(r,\theta)$ is then computed based on a weighted combination of f at nearby vertices (e.g. $\mathrm{f}(r,\theta) = w_c\mathrm{f}(c) + w_d\mathrm{f}(d) + \cdots$). Depending on the nature of f this linear blending can be lossy.

For the case of a fixed length serialization, the use of an RNN supporting variable length input is not necessary. A fully-connected layer (combined with some non-linearity) can be used instead. Naturally, we apply these neighborhood encoding operations repeatedly in multiple layers in a neural network to facilitate the mapping of input features to a higher level feature representation. This is detailed in the following section.

Tessellation Dependence Our simple method of encoding the neighborhood obviously is not independent of the tessellation of the input. By augmenting the features f with metric information (i.e. by appending length and angle information), we can mitigate this and essentially enable the network to possibly *learn* to be independent. In Section 4.1 we investigate the effects of this.

Concretely, we concatenate to the input feature $\mathrm{f}(c)$ the distance of the current vertex c to the center vertex a as well as the angle at a between the previous vertex b and c.

3.3 Architecture Details

To evaluate and compare our proposed methods (with variable or fixed length sequences) in the context of shape correspondence estimation, we construct our network architectures in a manner similar to the GCNN3 model proposed by Masci et al. [17]. We replace the convolution layers in GCNN3 by the ones presented above, as detailed below. For the sake of comparability, we use the SHOT descriptor proposed by Salti et al. [26] with 544 dimensions and default parameter settings computed at each vertex as input, following [3,18].

The original GCNN3 [17] network is constructed as FC16 + GC32 + GC64 + GC128 + FC256 + FC6890. FCx refers to a fully connected layer with output size x, which is applied to each vertex separately. GCx is the geodesic convolution operation followed by angular max-pooling, producing x-dimensional feature vectors for every vertex.

LSTM-NET Our network (LSTM-NET) for sequences with varying length replaces the GC layers and is constructed as FC16 + LSTM150 + LSTM200 + LSTM250 + FC256 + FC6890. LSTMx is the application of a LSTM cell to a sequence consisting of the input vertex and its neighborhood. In this manner we compute a new feature vector with dimensionality x (encoding neighborhood information) for every vertex, similar to a convolution operation.

Table 1. Number of parameters used in the different network architectures. FCS-NET (20) refers to FCS-NET applied to sequences with length 20, while GCNN3 is our implementation of GCNN3 [17] with the SHOT descriptor

Network	Number of Parameters
GCNN3 (SHOT)	2,672,634
LSTM-NET	2,675,706
FCS-NET (20)	2,763,356

FCS-NET For fixed-length sequences we make use of a network (FCS-NET) constructed as FC16 + FCS100 + FCS150 + FCS200 + FC256 + FC6890. FCSx refers to a fully-connected layer, which takes the concatenated features of a sequence as input and produces a x-dimensional output for every vertex, analogously to the LSTMx operation above.

We apply ReLU [19] to all layer outputs except for the output of the final layer to which we apply softmax. As regularization we apply dropout [28] with $p = 0.3$ after FC16 and FC256. For fair comparison, the layers of our LSTM-NET and FCS-NET were chosen such that the total number of learnable parameters is roughly equal to that of GCNN3 (cf. Table 1). Our networks are implemented with TensorFlow [1].

4 Experiments

For our experiments we used the FAUST dataset (consisting of 100 shapes) [2]. This allows for comparisons to related previous methods, which have commonly been evaluated on this dataset. Following common procedure, for training we used the first 80 shapes (10 of which were used for validation). All experiment results were computed on the last 20 shapes (our test set). We optimized all networks with Adam [12] ($lr = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$), where each batch consisted of the vertices of one mesh.

In order to evaluate the performance of our LSTM-NET we restrict ourself to sequences of fixed length as input (even though it would be capable of dealing with variable length input). This is because the mesh connectivity is the same over all meshes of the dataset. For varying length sequences (e.g. the 1- and 2-ring of each vertex) the network would potentially be able to learn the valence distribution and use connectivity information as an (unfair) prediction help.

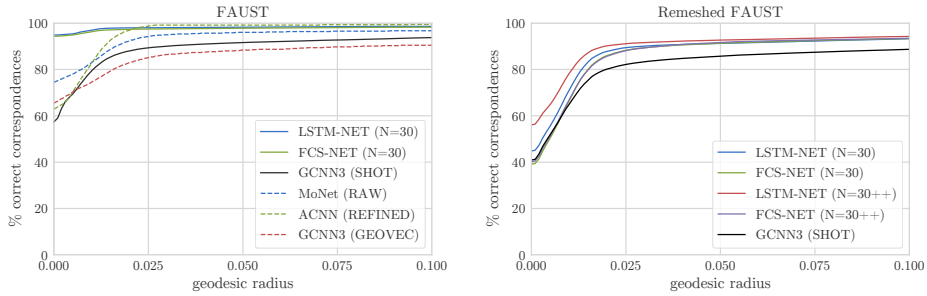
Following Kim et al. [11] we compute point-to-point correspondences and plot the percentage of correct correspondences found within given geodesic radii. For the evaluation no symmetry information is taken into account. We compare to the results from [17,3,18]. In addition we also implemented GCNN3 (using the SHOT instead of the GEOVEC descriptor as input) after Masci et al. [17] and evaluated the method in our setting. We used the parameters and loss proposed in the original paper. As shown in Figure 2 (a) our method outperforms current patch-based approaches with both LSTM-NET and FCS-NET for a sequence length of 30. Note that, by contrast, the average number of interpolated vertices

in a patch for GCNN3 is 80. Furthermore, we do not perform any post-processing or refinement on the network predictions. An evaluation of the effect of different sequence lengths is visualized in Figure 3 (a-b). Even with shorter sequence lengths (15) our method achieves competitive results. Qualitative results are visualized in Figure 6. We show the geodesic distance to the ground truth target vertices on four shapes from the test set. Correspondence errors of relative geodesic distance > 0.2 are clamped for an informative color coding.

4.1 Tessellation Dependence

An important, but often overlooked detail is the fact that the shapes in the FAUST dataset are meshed compatibly, i.e. the mesh connectivity is identical across shapes, and identical vertices are at corresponding points. Unless a correspondence estimation method is truly tessellation-oblivious, this naturally has the potential to incur a beneficial bias in this artifical benchmark, as in any realistic correspondence estimation application scenario, the tessellation will of course be incompatible. We thus repeat our experiments with a remeshed version of the FAUST dataset (see Figure 4), where each shape was remeshed individually and incompatibly.

Quantitative results are shown in Figure 2 (b). Here (++) denotes the additional relative information that we concatenate to the SHOT descriptor vectors.



(a) (b)

Fig. 2. Here the percentage of correct point-to-point correspondence predictions included in varying geodesic radii is shown. (a) shows a comparison of our approaches (FCS-NET, LSTM-NET) on sequences of length N=30 to current approaches. Dashed lines refer to results reported in previous work. For GCNN3 [17] we compare against the original version that uses the GEOVEC descriptor (dashed) as well as our implementation of GCNN3 (black), which takes the more advanced SHOT descriptor as input. ACNN [3] shows the results after a correspondence map refinement step. For the sake of fair comparison we show the raw (w/o refinement) performance of MoNet [18], as we do not perform any refinement for the output of FCS- and LSTM-NET either. (b) visualizes the results on the remeshed FAUST dataset (cf. Sec. 4.1). As expected, the addition of relative angles and distances (++) is beneficial.

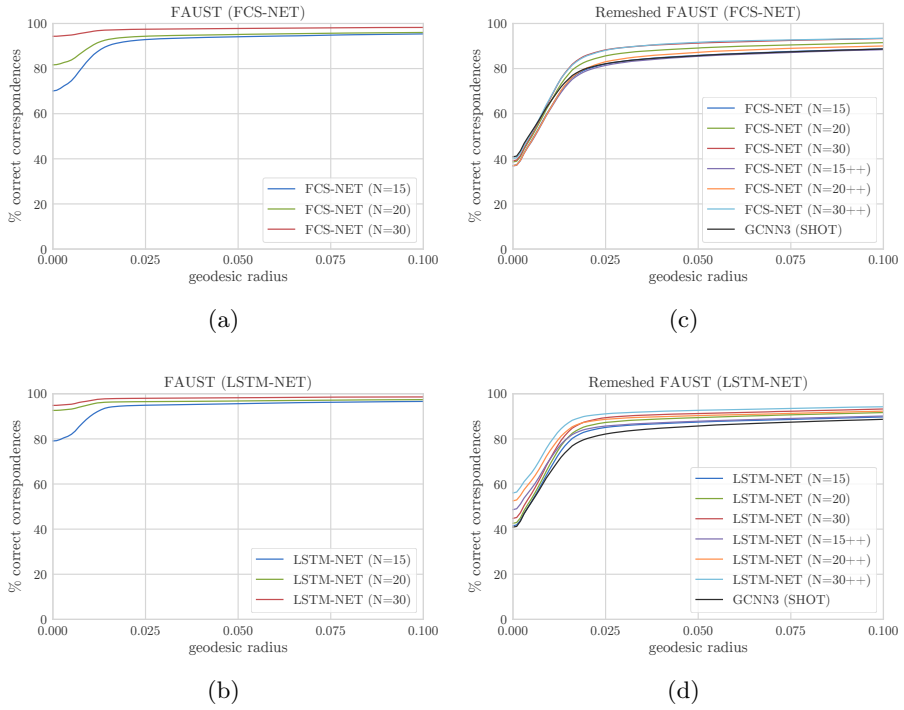


Fig. 3. Here the percentage of correct point-to-point correspondence predictions included in varying geodesic radii is shown. (a-b) show the effect of different sequence lengths (N=15,20,30) for the FAUST dataset. Even with relatively short sequences (15) we achieve competitive results. (c-d) visualize the results on the remeshed FAUST dataset. For comparison we also show the performance of the GCNN3 [17] network with the SHOT descriptor. (++) denotes the usage of additional metric information.

On this more challenging dataset we likewise achieve competitive results. Especially the additional information (++) enables our networks to encode less tessellation-dependent representations of neighborhoods for better performance. The effect of different sequence lengths is shown for this dataset in Figure 3 (c-d). For the sake of comparison to the performance of FCS-NET we also restrict LSTM-NET to sequences of fixed length. See Figure 7 for qualitative results.

Furthermore, we test the robustness of our network predictions to random starting points after the center vertex in our sequences (random rotations of the spiral). To this end we perform 100 predictions with different random rotations on the remeshed FAUST dataset with both FCS-NET and LSTM-NET. As shown in Figure 5 our networks are highly robust to these random orientations, such that the curves of separate predictions are not discernible.

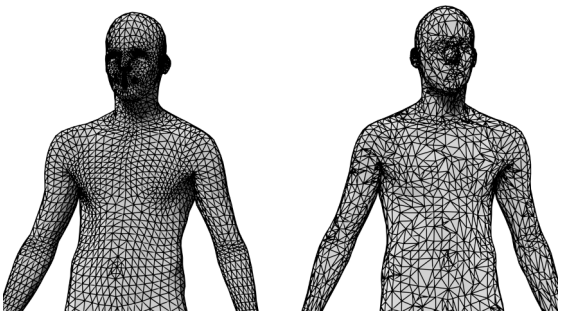
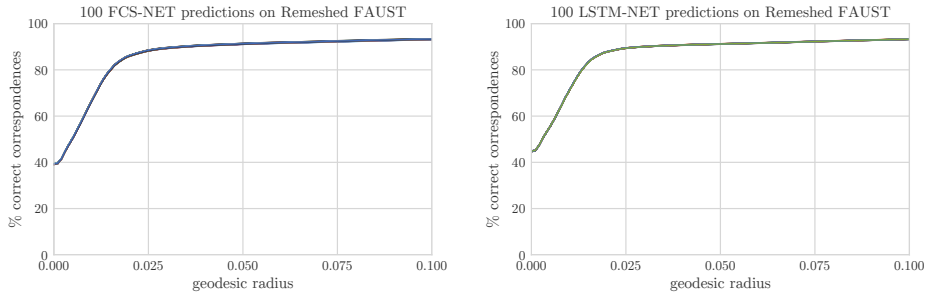


Fig. 4. Left: triangulation of a shape from the original FAUST dataset. Right: independently remeshed version.



(a) (b)

Fig. 5. (a-b) show the robustness of our approach to random rotations of the spirals. We perform 100 inference runs on the test set of the remeshed FAUST dataset with varying random rotations. The 100 different resulting curves plotted here are not distinguishable due to the robustness of our trained networks.

5 Conclusion

In this paper we presented a simple resampling free input encoding strategy for local neighborhoods in 3D surface meshes. Previous approaches rely on forms of resampling of input features in neighborhood patches, which incurs additional computational and implementational costs and can have negative effects on input data fidelity. Our experiments show that our approach, despite its simple and efficient nature, is able to achieve competitive results for the challenging task of shape correspondence estimation.

Limitations and Future Work Although the introduction of metric information aims to make our method less sensitive to tessellation, it is nevertheless affected by it; this, however, is true to some extent in any practical setting for previous patch-based approaches as well. The design of truly tessellation-oblivious encoding strategies is a relevant challenge for future work, as it would

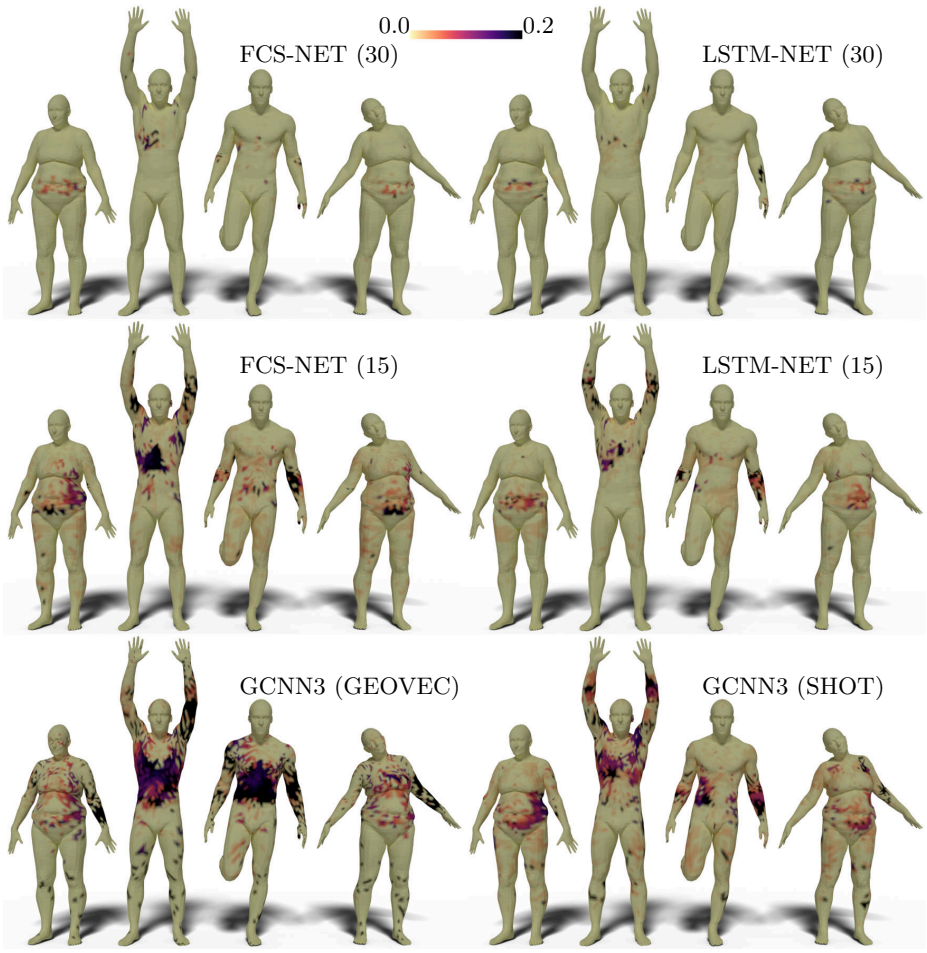


Fig. 6. Geodesic error for 4 shapes from the test set of the FAUST dataset.

relieve the training process from having to *learn* tessellation independence, as required for optimal performance.

Furthermore, high resolution meshes require longer sequences to encode relevant neighborhood information. In the case of FCS-NET this also means an increase in the number of parameters required to learn, which can lead to memory issues. An interesting avenue for future work thus is the investigation of sub-sampled (but not resampled) serialization.

A related issue is that the training of RNNs tends to be slower than that of CNNs. A possible solution to this problem could be the application of 1D convolutions instead of LSTM cells or fully connected layers. An investigation into feature learning, given only raw input data (e.g. lengths, angles, or positions

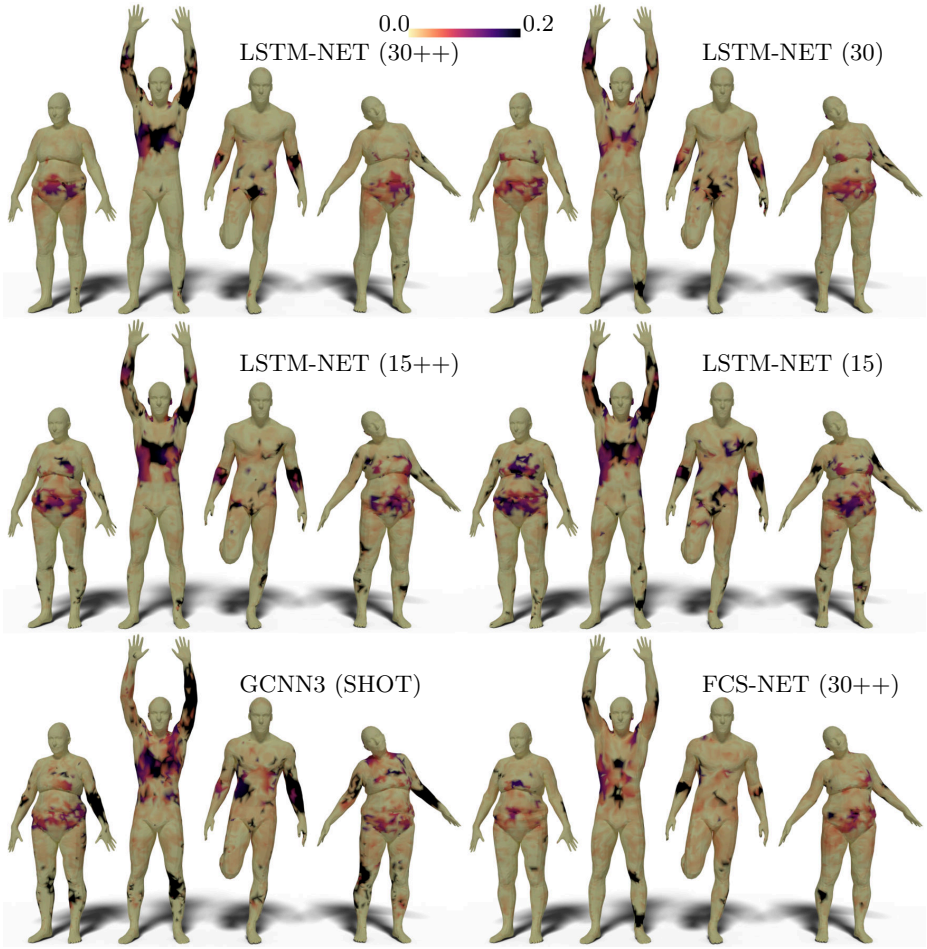


Fig. 7. Geodesic error for 4 shapes from the test set of the remeshed FAUST dataset.

of mesh elements) instead of preprocessed information like the SHOT descriptor will also be of interest.

Acknowledgements

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement n° [340884]. We would like to thank the authors of related work [17,3] for making their implementations available, as well as the reviewers for their insightful comments.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `https://www.tensorflow.org/`, software available from tensorflow.org
2. Bogo, F., Romero, J., Loper, M., Black, M.J.: FAUST: Dataset and evaluation for 3D mesh registration. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). IEEE, Piscataway, NJ, USA (Jun 2014)
3. Boscaini, D., Masci, J., Rodolà, E., Bronstein, M.: Learning shape correspondence with anisotropic convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 3189–3197 (2016)
4. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems. pp. 3844–3852 (2016)
5. Eynard, D., Kovnatsky, A., Bronstein, M.M., Glashoff, K., Bronstein, A.M.: Multimodal manifold analysis by simultaneous diagonalization of laplacians. IEEE transactions on pattern analysis and machine intelligence **37**(12), 2505–2517 (2015)
6. Eynard, D., Rodola, E., Glashoff, K., Bronstein, M.M.: Coupled functional maps. In: 3D Vision (3DV), 2016 Fourth International Conference on. pp. 399–407. IEEE (2016)
7. Ezuz, D., Solomon, J., Kim, V.G., Ben-Chen, M.: Gwcnn: A metric alignment layer for deep shape analysis. In: Computer Graphics Forum. vol. 36, pp. 49–57. Wiley Online Library (2017)
8. Gehre, A., Bronstein, M., Kobbelt, L., Solomon, J.: Interactive curve constrained functional maps. Computer Graphics Forum **37**(5) (2018)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
10. Huang, Q., Wang, F., Guibas, L.: Functional map networks for analyzing and exploring large shape collections. ACM Transactions on Graphics (TOG) **33**(4), 36 (2014)
11. Kim, V.G., Lipman, Y., Funkhouser, T.: Blended intrinsic maps. In: ACM Transactions on Graphics (TOG). vol. 30, p. 79. ACM (2011)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Kostrikov, I., Jiang, Z., Panozzo, D., Zorin, D., Bruna, J.: Surface networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
14. Kovnatsky, A., Bronstein, M.M., Bronstein, A.M., Glashoff, K., Kimmel, R.: Coupled quasi-harmonic bases. In: Computer Graphics Forum. vol. 32, pp. 439–448. Wiley Online Library (2013)
15. Litany, O., Remez, T., Rodola, E., Bronstein, A.M., Bronstein, M.M.: Deep functional maps: Structured prediction for dense shape correspondence. In: Proc. ICCV. vol. 2, p. 8 (2017)
16. Maron, H., Galun, M., Aigerman, N., Trope, M., Dym, N., Yumer, E., Kim, V.G., Lipman, Y.: Convolutional neural networks on surfaces via seamless toric covers. ACM Trans. Graph **36**(4), 71 (2017)

17. Masci, J., Boscaini, D., Bronstein, M., Vandergheynst, P.: Geodesic convolutional neural networks on riemannian manifolds. In: Proceedings of the IEEE international conference on computer vision workshops. pp. 37–45 (2015)
18. Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns. In: Proc. CVPR. vol. 1, p. 3 (2017)
19. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). pp. 807–814 (2010)
20. Niepert, M., Ahmed, M., Kutzkov, K.: Learning convolutional neural networks for graphs. In: International conference on machine learning. pp. 2014–2023 (2016)
21. Nogneng, D., Melzi, S., Rodolà, E., Castellani, U., Bronstein, M., Ovsjanikov, M.: Improved functional mappings via product preservation. In: Computer Graphics Forum. vol. 37, pp. 179–190. Wiley Online Library (2018)
22. Nogneng, D., Ovsjanikov, M.: Informative descriptor preservation via commutativity for shape matching. In: Computer Graphics Forum. vol. 36, pp. 259–267. Wiley Online Library (2017)
23. Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A., Guibas, L.: Functional maps: a flexible representation of maps between shapes. ACM Transactions on Graphics (TOG) **31**(4), 30 (2012)
24. Pokrass, J., Bronstein, A.M., Bronstein, M.M., Sprechmann, P., Sapiro, G.: Sparse modeling of intrinsic correspondences. In: Computer Graphics Forum. vol. 32, pp. 459–468. Wiley Online Library (2013)
25. Rodolà, E., Cosmo, L., Bronstein, M.M., Torsello, A., Cremers, D.: Partial functional correspondence. In: Computer Graphics Forum. vol. 36, pp. 222–236. Wiley Online Library (2017)
26. Salti, S., Tombari, F., Di Stefano, L.: Shot: Unique signatures of histograms for surface and texture description. Computer Vision and Image Understanding **125**, 251–264 (2014)
27. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks **20**(1), 61–80 (2009)
28. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
29. Van Kaick, O., Zhang, H., Hamarneh, G., Cohen-Or, D.: A survey on shape correspondence. In: Computer Graphics Forum. vol. 30, pp. 1681–1707. Wiley Online Library (2011)