DEMEA: Deep Mesh Autoencoders for Non-Rigidly Deforming Objects

Edgar Tretschk[1] Ayush Tewari[1]
Michael Zollhöfer[2] Vladislav Golyanik[1] Christian Theobalt[1]

[1] Max Planck Institute for Informatics, Saarland Informatics Campus
[2] Stanford University

Abstract. Mesh autoencoders are commonly used for dimensionality reduction, sampling and mesh modeling. We propose a general-purpose DEep MEsh Autoencoder (DEMEA) which adds a novel embedded deformation layer to a graph-convolutional mesh autoencoder. The embedded deformation layer (EDL) is a differentiable deformable geometric proxy which explicitly models point displacements of non-rigid deformations in a lower dimensional space and serves as a local rigidity regularizer. DEMEA decouples the parameterization of the deformation from the final mesh resolution since the deformation is defined over a lower dimensional embedded deformation graph. We perform a large-scale study on four different datasets of deformable objects. Reasoning about the local rigidity of meshes using EDL allows us to achieve higher-quality results for highly deformable objects, compared to directly regressing vertex positions. We demonstrate multiple applications of DEMEA, including non-rigid 3D reconstruction from depth and shading cues, non-rigid surface tracking, as well as the transfer of deformations over different meshes.

Keywords: auto-encoding, embedded deformation, non-rigid tracking

1 Introduction

With the increasing volume of datasets of deforming objects enabled by modern 3D acquisition technology, the demand for compact data representations and compression grows. Dimensionality reduction of mesh data has multiple applications in computer graphics and vision, including shape retrieval, generation, interpolation, and completion. Recently, deep convolutional autoencoder networks were shown to produce compact mesh representations [2, 37, 31, 6].

Dynamic real-world objects do not deform arbitrarily. While deforming, they preserve topology, and nearby points are more likely to deform similarly compared to more distant points. Current convolutional mesh autoencoders exploit this coherence by learning the deformation properties of objects directly from data and are already suitable for mesh compression and representation learning. On the other hand, they do not explicitly reason about the deformation field in terms of local rotations and translations. We show that explicitly reasoning about the local rigidity of meshes enables higher-quality results for highly deformable objects, compared to directly regressing vertex positions.

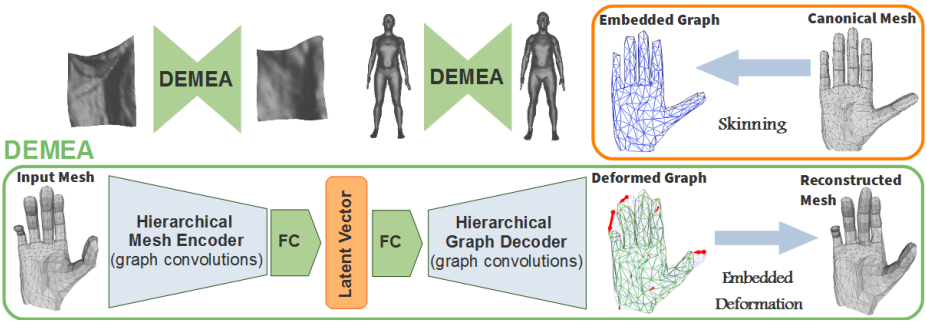


Fig. 1: Pipeline: DEMEA encodes a mesh using graph convolutions on a mesh hierarchy. The graph decoder first maps the latent vector to node features of the coarsest graph level. A number of upsampling and graph convolution modules infer the node translations and rotations of the embedded graph. An embedded deformation layer applies the node translations to a template graph, against which a template mesh is skinned. With the node rotations and the skinning, this deformed graph allows reconstructing a deformed mesh.

At the other end of the spectrum, mesh manipulation techniques such as As-Rigid-As-Possible Deformation [34] and Embedded Deformation [35] only require a single mesh and enforce deformation properties, such as smoothness and local rigidity, based on a set of hand-crafted priors. These hand-crafted priors are effective and work surprisingly well, but since they do not model the real-world deformation behavior of the physical object, they often lead to unrealistic deformations and artifacts in the reconstructions.

In this paper, we propose a general-purpose mesh autoencoder with a model-based deformation layer, combining the best of both worlds, *i.e.,* supervised learning with deformable meshes and a novel *differentiable embedded deformation* layer that models the deformable meshes using lower-dimensional deformation graphs with physically interpretable deformation parameters. While the core of our DEep MEsh Autoencoder (DEMEA) learns the deformation model of objects from data using the state-of-the-art convolutional mesh autoencoder (CoMA) [31], the novel embedded deformation layer decouples the parameterization of object motion from the mesh resolution and introduces local spatial coherence via vertex skinning. DEMEA is trained on mesh datasets of moderate sizes that have recently become available [24, 4, 3, 26]. DEMEA is a general mesh autoencoding approach that can be trained for any deformable object class. We evaluate our approach on datasets of three objects with large deformations like articulated deformations (body, hand) and large non-linear deformations (cloth), and one object with small localized deformations (face). Quantitatively, DEMEA outperforms standard convolutional mesh autoencoder architectures in terms of vertex-to-vertex distance error. Qualitatively, we show that DEMEA produces visually higher fidelity results due to the physically based embedded

deformation layer. We show several applications of DEMEA in computer vision and graphics. Once trained, the decoder of our autoencoders can be used for shape compression, high-quality depth-to-mesh reconstruction of human bodies and hands, and even poorly textured RGB-image-to-mesh reconstruction for deforming cloth. The low-dimensional latent space learned by our approach is meaningful and well-behaved, which we demonstrate by different applications of latent space arithmetic. Thus, DEMEA provides us a well-behaved general-purpose category-specific generative model of highly deformable objects.

2 Related Work

Mesh Manipulation and Tracking. Our embedded deformation layer is inspired by as-rigid-as-possible modelling [34] and the method of Sumner *et al.* [35] for mesh editing and manipulation. While these methods have been shown to be very useful for mesh manipulation in computer graphics, to the best of our knowledge, this is the first time a model-based regularizer is used in a mesh autoencoder. Using a template for non-rigid object tracking from depth maps was extensively studied in the model-based setting [22, 41]. Recently, Litany *et al.* [23] demonstrated a neural network-based approach for the completion of human body shapes from a single depth map.

Graph Convolutions. The encoder-decoder approach to dimensionality reduction with neural networks (NNs) for images was introduced in [17]. Deep convolutional neural networks (CNNs) allow to effectively capture contextual information of input data modalities and can be trained for various tasks. Lately, convolutions operating on regular grids have been generalized to more general topologically connected structures such as meshes and two-dimensional manifolds [7, 29], enabling learning of correspondences between shapes, shape retrieval [27, 5, 28], and segmentation [40]. Masci *et al.* [27] proposed geodesic CNNs operating on Riemannian manifolds for shape description, retrieval, and correspondence estimation. Boscani *et al.* [5] introduced spatial weighting functions based on simulated heat propagation and projected anisotropic convolutions. Monti *et al.* [28] extended graph convolutions to variable patches through Gaussian mixture model CNNs. In FeaSTNet [38], the correspondences between filter weights and graph neighborhoods with arbitrary connectivities are established dynamically from the learned features. The localized spectral interpretation of Defferrard *et al.* [9] is based on recursive feature learning with Chebyshev polynomials and has linear evaluation complexity. Focusing on mesh autoencoding, Bouritsas *et al.* [6] exploited the fixed ordering of neighboring vertices.

Learning Mesh-Based 3D Autoencoders. Very recently, several mesh autoencoders with various applications were proposed. A new hierarchical variational mesh autoencoder with fully connected layers for facial geometry parameterization learns an accurate face model from small databases and accomplishes depth-to-mesh fitting tasks [2]. Tan and coworkers [36] introduced a mesh autoencoder with a rotation-invariant mesh representation as a generative model. Their network can generate new meshes by sampling in the latent space and

perform mesh interpolation. To cope with meshes of arbitrary connectivity, they used fully-connected layers and did not explicitly encode neighbor relations. Tan *et al.* [37] trained a network with graph convolutions to extract sparse localized deformation components from meshes. Their method is suitable for large-scale deformations and meshes with irregular connectivity. Gao *et al.* [12] transferred mesh deformations by training a generative adversarial network with a cycle consistency loss to map shapes in the latent space, while a variational mesh autoencoder encodes deformations. The Convolutional facial Mesh Autoencoder (CoMA) of Ranjan *et al.* [31] allows to model and sample stronger deformations compared to previous methods and supports asymmetric facial expressions. The Neural 3DMM of Bouritsas *et al.* [6] improves quantitatively over CoMA due to better training parameters and task-specific graph convolutions. Similar to CoMA [31], our DEMEA uses spectral graph convolutions but additionally employs the embedded deformation layer as a model-based regularizer.

Learning 3D Reconstruction. Several supervised methods reconstruct rigid objects in 3D. Given a depth image, the network of Sinha *et al.* [33] reconstructs the observed surface of non-rigid objects. In its 3D reconstruction mode, their method reconstructs rigid objects from single images. Similarly, Groueix *et al.* [15] reconstructed object surfaces from a point cloud or single monocular image with an atlas parameterization. The approaches of Kurenkov *et al.* [21] and Jack *et al.* [18] deform a predefined object-class template to match the observed object appearance in an image. Similarly, Kanazawa *et al.* [19] deformed a template to match the object appearance but additionally support object texture. The Pixel2Mesh approach of Wang *et al.* [39] reconstructs an accurate mesh of an object in a segmented image. Initializing the 3D reconstruction with an ellipsoid, their method gradually deforms it until the appearance matches the observation. The template-based approaches [21, 18, 19], as well as Pixel2Mesh [39], produce complete 3D meshes.

Learning Monocular Non-Rigid Surface Regression. Only a few supervised learning approaches for 3D reconstruction from monocular images tackle the deformable nature of non-rigid objects. Several methods [30, 14, 32] train networks for deformation models with synthetic thin plates datasets. These approaches can infer non-rigid states of the observed surfaces such as paper sheets or membranes. Still, their accuracy and robustness on real images are limited. Bednařík *et al.* [3] proposed an encoder-decoder network for texture-less surfaces relying on shading cues. They trained on a real dataset and showed an enhanced reconstruction accuracy on real images, but support only trained object classes. Fuentes-Jimenez *et al.* [11] trained a network to deform an object template for depth map recovery. They achieved impressive results on real image sequences but require an accurate 3D model of every object in the scene, which restricts the method's practicality. One of the applications of DEMEA is the recovery of texture-less surfaces from RGB images. Since a depth map as a data modality is closer to images with shaded surfaces, we train DEMEA in the depth-to-mesh mode on images instead of depth maps. As a result, we can regress surface geometry from shading cue.

3 Approach

In this section, we describe the architecture of the proposed DEMEA. We employ an embedded deformation layer to decouple the complexity of the learned deformation field from the actual mesh resolution. The deformation is represented relative to a canonical mesh $\mathcal{M} = (\mathbf{V}, \mathbf{E})$ with N_v vertices $\mathbf{V} = \{\mathbf{v}_i\}_{i=1}^{N_v}$, and edges $\mathbf{E}$. To this end, we define the encoder-decoder on a coarse deformation graph and use the embedded deformation layer to drive the deformation of the final high-resolution mesh, see Fig. 1. Our architecture is based on graph convolutions that are defined on a multi-resolution mesh hierarchy. In the following, we describe all components in more detail. We describe the employed spiral graph convolutions [6] in the supplemental document.

3.1 Mesh Hierarchy

The up- and downsampling in the convolutional mesh autoencoder is defined over a multi-resolution mesh hierarchy, similar to the CoMA [31] architecture. We compute the mesh hierarchy fully automatically based on quadric edge collapses [13], *i.e.,* each hierarchy level is a simplified version of the input mesh. We employ a hierarchy with five resolution levels, where the finest level is the mesh. Given the multi-resolution mesh hierarchy, we define up- and downsampling operations [31] for feature maps defined on the graph. To this end, during downsampling, we enforce the nodes of the coarser level to be a subset of the nodes of the next finer level. We transfer a feature map to the next coarser level by a similar subsampling operation. The inverse operation, *i.e.,* feature map upsampling, is implemented based on a barycentric interpolation of close features. During edge collapse, we project each collapsed node onto the closest triangle of the coarser level. We use the barycentric coordinates of this closest point with respect to the triangle's vertices to define the interpolation weights.

3.2 Embedded Deformation Layer (EDL)

Given a canonical mesh, we have to pick a corresponding coarse embedded deformation graph. We employ MeshLab's [8] more sophisticated implementation of quadric edge collapse to fully automatically generate the graph. See the supplemental document for details. The deformation graph is used as one of the two levels immediately below the mesh in the mesh hierarchy (depending on the resolution of the graph) of the autoencoder. When generating the mesh hierarchy, we need to enforce the subset relationship between levels. However, the quadric edge collapse algorithm of [31] might delete nodes of the embedded graph when computing intermediate levels between the mesh and the embedded graph. We ensure that those nodes are not removed by setting the cost of removing them from levels that are at least as fine as the embedded graph to infinity.

Our embedded deformation layer models a space deformation that maps the vertices of the canonical template mesh $\mathbf{V}$ to a deformed version $\hat{\mathbf{V}}$. Suppose $\mathcal{G} = (\mathbf{N}, \mathbf{E})$ is the embedded deformation graph [35] with L canonical nodes

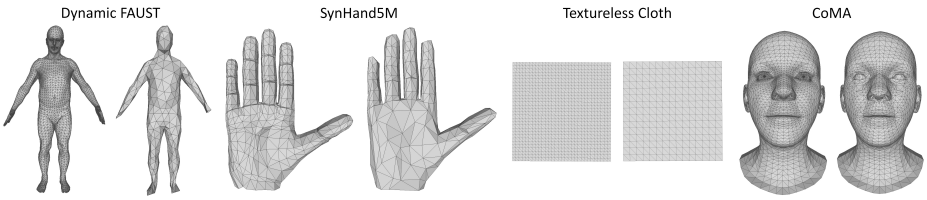


Fig. 2: Template mesh and the corresponding embedded deformation graph pairs automatically generated using [8].

$\mathbf{N} = \{\boldsymbol{g}_l\}_{i=1}^{L}$ and K edges $\mathbf{E}$, with $\boldsymbol{g}_l \in \mathbb{R}^3$. The global space deformation is defined by a set of local, rigid, per-graph node transformations. Each local rigid space transformation is defined by a tuple $\boldsymbol{T}_l = (\boldsymbol{R}_l, \boldsymbol{t}_l)$, with $\boldsymbol{R}_l \in \mathbf{SO}(3)$ being a rotation matrix and $\boldsymbol{t}_l \in \mathbb{R}^3$ being a translation vector. We enforce that $\boldsymbol{R}_l^\mathsf{T} = \boldsymbol{R}_l^{-1}$ and $\det(\boldsymbol{R}_l) = 1$ by parameterizing the rotation matrices based on three Euler angles. Each $\boldsymbol{T}_l$ is anchored at the canonical node position $\boldsymbol{g}_l$ and maps every point $\boldsymbol{p} \in \mathbb{R}^3$ to a new position in the following manner [35]:

$$\boldsymbol{T}_l(\boldsymbol{p}) = \boldsymbol{R}_l[\boldsymbol{p} - \boldsymbol{g}_l] + \boldsymbol{g}_l + \boldsymbol{t}_l. \tag{1}$$

To obtain the final global space deformation $\mathbf{G}$, the local per-node transformations are linearly combined:

$$\mathbf{G}(\boldsymbol{p}) = \sum_{l \in \mathcal{N}_{\boldsymbol{p}}} w_l(\boldsymbol{p}) \cdot \boldsymbol{T}_l(\boldsymbol{p}) \ . \tag{2}$$

Here, $\mathcal{N}_{\boldsymbol{p}}$ is the set of approximate closest deformation nodes. The linear blending weights $w_l(\boldsymbol{p})$ for each position are based on the distance to the respective deformation node [35]. Please refer to the supplemental for more details.

The deformed mesh $\hat{\mathbf{V}} = \mathbf{G}(\mathbf{V})$ is obtained by applying the global space deformation to the canonical template mesh $\mathbf{V}$. The free parameters are the local per-node rotations $\boldsymbol{R}_l$ and translations $\boldsymbol{t}_l$, *i.e.,* $6L$ parameters with L being the number of nodes in the graph. These parameters are input to our deformation layer and are regressed by the graph convolutional decoder.

3.3 Differentiable Space Deformation

Our novel EDL is fully differentiable and can be used during network training to decouple the parameterization of the space deformation from the resolution of the final high-resolution output mesh. This enables us to define the reconstruction loss on the final high-resolution output mesh and backpropagate the errors via the skinning transform to the coarse parameterization of the space deformation. Thus, our approach enables finding the best space deformation by only supervising the final output mesh.

3.4 Training

We train our approach end-to-end in Tensorflow [1] using Adam [20]. As loss we employ a dense geometric per-vertex ℓ_1-loss with respect to the ground-truth mesh. For all experiments, we use a learning rate of 10^{-4} and default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ for Adam. We train for 50 epochs for Dynamic Faust, 30 epochs for SynHand5M, 50 epochs for the CoMA dataset and 300 epochs for the Cloth dataset. We employ a batch size of 8.

3.5 Reconstructing Meshes from Images/Depth

The image/depth-to-mesh network consists of an image encoder and a mesh decoder, see Fig. 5a. The mesh decoder is initialized from the corresponding mesh auto-encoder, the image/depth encoder is based on a ResNet-50 [16] architecture, and the latent code is shared between the encoder and decoder. We initialize the ResNet-50 component using pre-trained weights from ImageNet [10]. To obtain training data, we render synthetic depth maps from the meshes. We train with the same settings as for mesh auto-encoding.

3.6 Network Architecture Details

In the following, we provide more details of our encoder-decoder architectures.
Encoding Meshes. Input to the first layer of our mesh encoder is an $N_v \times 3$ tensor that stacks the coordinates of all N_v vertices. We apply four *downsampling modules*. Each module applies a graph convolution and is followed by a downsampling to the next coarser level of the mesh hierarchy. We use spiral graph convolutions [6] and similarly apply an ELU non-linearity after each convolution. Finally, we take the output of the final module and apply a fully connected layer followed by an ELU non-linearity to obtain a latent space embedding.
Encoding Images/Depth. To encode images/depth, we employ a 2D convolutional network to map color/depth input to a latent space embedding. Input to our encoder are images of resolution 256×256 pixels. We modified the ResNet-50 [16] architecture to take single or three-channel input image. We furthermore added two additional convolution layers at the end, which are followed by global average pooling. Finally, a fully connected layer with a subsequent ELU non-linearity maps the activations to the latent space.
Decoding Graphs. The task of the graph decoder is to map from the latent space back to the embedded deformation graph. First, we employ a fully connected layer in combination with reshaping to obtain the input to the graph convolutional *upsampling modules*. We apply a sequence of three or four upsampling modules until the resolution level of the embedded graph is reached. Each upsampling module first up-samples the features to the next finer graph resolution and then performs a graph convolution, which is then followed by an ELU non-linearity. Then, we apply two graph convolutions with ELUs for refinement and a final convolution without an activation function. The resulting tensor is passed to our embedded deformation layer.

4 Experiments

We evaluate DEMEA quantitatively and qualitatively on several challenging datasets and demonstrate state-of-the-art results for mesh auto-encoding. In Sec. 5, we show reconstruction from RGB images and depth maps and that the learned latent space enables well-behaved latent arithmetic. We use Tensorflow 1.5.0 [1] on Debian with an NVIDIA Tesla V100 GPU.

Datasets. We demonstrate DEMEA's generality on experiments with body (Dynamic Faust, DFaust [4]), hand (SynHand5M [26]), textureless cloth (Cloth [3]), and face (CoMA [31]) datasets. Table 1 gives the number of graph nodes used

	Mesh	1st	2nd	3rd	4th
DFaust [4]	6890	1723	**431**	108	27
CoMA [31]	5023	**1256**	314	79	20
SynHand5M [26]	1193	**299**	75	19	5
Cloth [3]	961	**256**	100	36	16

Table 1: Number of vertices on each level of the mesh hierarchy. Bold levels denote the embedded graph. Note that except for Cloth these values were computed automatically based on [31].

	DFaust		SynHand5M		Cloth		CoMA	
	8	32	8	32	8	32	8	32
CA	6.35	**2.07**	8.12	2.60	**11.21**	6.50	**1.17**	0.72
MCA	**6.21**	2.13	**8.11**	2.67	11.64	6.59	1.20	0.71
Ours	6.69	2.23	8.12	**2.51**	11.28	6.40	1.23	0.81
FCA	6.51	2.17	15.10	2.95	15.63	5.99	1.77	**0.67**
FCED	6.26	2.14	14.61	2.75	15.87	**5.94**	1.81	0.73

Table 2: Average per-vertex errors on the test sets of DFaust (cm), SynHand5M (mm), textureless cloth (mm) and CoMA (mm) for 8 and 32 latent dimensions.

on each level of our hierarchical architecture. All meshes live in metric space.

DFaust [4]. The training set consists of 28,294 meshes. For the tests, we split off two identities (female 50004, male 50002) and two dynamic performances, *i.e.,* *one-leg jump* and *chicken wings*. Overall, this results in a test set with $12,926$ elements. For the depth-to-mesh results, we found the synthetic depth maps from the DFaust training set to be insufficient for generalization, *i.e.,* the test error was high. Thus, we add more pose variety to DFaust for the depth-to-mesh experiments. Specifically, we add $28k$ randomly sampled poses from the CMU Mocap[1] dataset to the training data, where the identities are randomly sampled from the SMPL [25] model ($14k$ female, $14k$ male). We also add $12k$ such samples to the test set ($6k$ female, $6k$ male).

Textureless Cloth [3]. For evaluating our approach on general non-rigidly deforming surfaces, we use the *textureless cloth* data set of Bednařík *et al.* [3]. It contains real depth maps and images of a white deformable sheet — observed in different states and differently shaded — as well as ground-truth meshes. In total, we select 3,861 meshes with consistent edge lengths. 3,167 meshes are used for training and 700 meshes are reserved for evaluation. Since the canonical mesh is a perfectly flat sheet, it lacks geometric features, which causes downsampling methods like [13], [31] and [8] to introduce severe artifacts. Hence, we generate the entire mesh hierarchy for this dataset, see the supplemental. This hierarchy

[1] mocap.cs.cmu.edu

is also used to train the other methods for the performed comparisons.
SynHand5M [26]. For the experiments with hands, we take $100k$ random meshes from the synthetic *SynHand5M* dataset of Malik *et al.* [26]. We render the corresponding depth maps. The training set is comprised of $90k$ meshes, and the remaining $10k$ meshes are used for evaluation.
CoMA [31]. The training set contains 17,794 meshes of the human face in various expressions [31]. For tests, we select two challenging expressions, *i.e., high smile* and *mouth extreme*. Thus, our test set contains 2,671 meshes in total.

4.1 Baseline Architectures

We compare DEMEA to a number of strong baselines.
Convolutional Baseline. We consider a version of our proposed architecture, *convolutional ablation (CA)*, where the ED layer is replaced by learned upsampling modules that upsample to the mesh resolution. In this case, the extra refinement convolutions occur on the level of the embedded graph. We also consider *modified CA (MCA)*, an architecture where the refinement convolutions are moved to the end of the network, such that they operate on mesh resolution.
Fully-Connected Baseline. We also consider an almost-linear baseline, *FC ablation (FCA)*. The input is given to a fully-connected layer, after which an ELU is applied. The resulting latent vector is decoded using another FC layer that maps to the output space. Finally, we also consider an *FCED* network where the fully-connected decoder maps to the deformation graph, which the embedded deformation layer (EDL) in turn maps to the full-resolution mesh.

4.2 Evaluation Settings

	DFaust		SynHand5M		Cloth		CoMA	
	8	32	8	32	8	32	8	32
w/ GL	8.92	2.75	9.02	2.95	**11.26**	6.45	1.38	0.99
w/ LP	7.71	**2.22**	**8.00**	2.52	11.46	7.96	1.25	**0.79**
Ours	**6.69**	2.23	8.12	**2.51**	11.28	**6.40**	**1.23**	0.81

Table 3: Average per-vertex errors on the test sets of DFaust (*cm*), SynHand5M (*mm*), textureless cloth (*mm*) and CoMA (*mm*) for 8 and 32 latent dimensions.

	DFaust		SynHand5M		Cloth		CoMA	
	8	32	8	32	8	32	8	32
N. 3DMM	7.09	**1.99**	8.50	2.58	12.64	6.49	1.34	**0.71**
Ours	**6.69**	2.23	**8.12**	**2.51**	**11.28**	**6.40**	**1.23**	0.81

Table 4: Average per-vertex errors on the test sets of DFaust (in *cm*), SynHand5M (in *mm*), textureless cloth (in *mm*) and CoMA (in *mm*) for 8 and 32 latent dimensions, compared with Neural 3DMM [6].

We first determine how to integrate the EDL into the training. Our proposed architecture regresses node positions and rotations and then uses the EDL to obtain the deformed mesh, on which the reconstruction loss is applied.

As an alternative, we consider the *graph loss (GL)* with the ℓ_1 reconstruction loss directly on the graph node positions (where the vertex positions of the

input mesh that correspond to the graph nodes are used as ground-truth). The GL setting uses the EDL only at test time to map to the full mesh, but not for training. Although the trained network predicts graph node positions $\boldsymbol{t}_l$ at test time, it does not regress graph node rotations $\boldsymbol{R}_l$ which are necessary for the EDL. We compute the missing rotation for each graph node l as follows: assuming that each node's neighborhood transforms roughly rigidly, we solve a small Procrustes problem that computes the rigid rotation between the 1-ring neighborhoods of l in the template graph and in the regressed network output. We directly use this rotation as $\boldsymbol{R}_l$.

We also consider the alternative of estimating the local Procrustes rotation *inside the network during training (LP)*. We add a reconstruction loss on the deformed mesh as computed by the EDL. Here, we do not back-propagate through the rotation computation to avoid training instabilities.

Table 3 shows the quantitative results using the average per-vertex Euclidean error. Using the EDL during training leads to better quantitative results, as the network is aware of the skinning function and can move the graph nodes accordingly. In addition to being an order of magnitude faster than LP, regressing rotations either gives the best results or is close to the best. We use the EDL with regressed rotation parameters during training in all further experiments.

We use spiral graph convolutions [6], but show in the supplemental document that spectral graph convolutions [9] also give similar results.

4.3 Evaluations of the Autoencoder

Qualitative Evaluations. Our architecture significantly outperforms the baselines qualitatively on the DFaust and SynHand5M datasets, as seen in Figs. 3 and 4. Convolutional architectures without an embedded graph produce strong artifacts in the hand, feet and face regions in the presence of large deformations. Since EDL explicitly models deformations, we preserve fine details of the template under strong non-linear deformations and articulations of extremities.

Quantitative Evaluations. We compare the proposed DEMEA to the baselines on the autoencoding task, see Table 2. While the fully-connected baselines are competitive for larger dimensions of the latent space, their memory demand increases drastically. On the other hand, they perform significantly worse for low dimensions on all datasets, except for DFaust. In this work, we are interested in low latent dimensions, *e.g.* less than 32, as we want to learn mesh representations that are as compact as possible. We also observe that adding EDL to the fully-connected baselines maintains their performance. Furthermore, the lower test errors of FCED on Cloth indicate that network capacity (and not EDL) limits the quantitative results.

On SynHand5M, Cloth and CoMA, the convolutional baselines perform on par with DEMEA. On DFaust, our performance is slightly worse, perhaps because other architectures can also fit to the high-frequency details and noise. EDL regularizes deformations to avoid artifacts, which also prevents fitting to high-frequency or small details. Thus, explicitly modelling deformations via the EDL and thereby avoiding artifacts has no negative impact on the quantitative

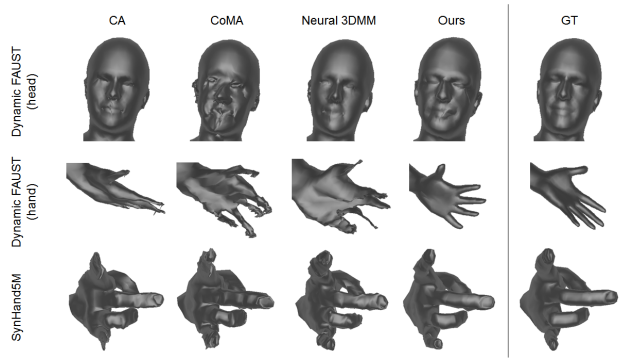


Fig. 3: In contrast to graph-convolutional networks that directly regress vertex positions, our embedded graph layer does not show artifacts. These results use a latent dimension of 32.

performance. Since CoMA mainly contains small and local deformations, DEMEA does not lead to any quantitative improvement. This is more evident in the case of latent dimension 32, as the baselines can better reproduce noise and other high-frequency deformations.

Comparisons. In extensive comparisons with several competitive baselines, we have demonstrated the usefulness of our approach for autoencoding strong non-linear deformations and articulated motion. Next, we compare DEMEA to the existing state-of-the-art CoMA approach [31]. We train their architecture on all mentioned datasets with a latent dimension of 8, which is also used in [31]. We outperform their method quantitatively on DFaust ($6.7cm$ *vs.* $8.4cm$), on SynHand5M ($8.12mm$ *vs.* $8.93mm$), on Cloth ($1.13cm$ *vs.* $1.44cm$), and even on CoMA ($1.23mm$ *vs.* $1.42mm$), where the deformations are not large. We also compare to Neural 3DMM [6] on latent dimensions 8 and 32, similarly to [31] on their proposed hierarchy. See Table 4 for the results. DEMEA performs better than Neural 3DMM in almost all cases. In Fig. 3, we show that DEMEA avoids many of the artifacts present in the case of [31], [6] and other baselines.

5 Applications

5.1 RGB to Mesh

On the Cloth [3] dataset, we show that DEMEA can reconstruct meshes from RGB images. See Fig. 5b for qualitative examples with a latent dimension of 32.

On our test set, our proposed architecture achieves RGB-to-mesh reconstruction errors of $16.1mm$ and $14.5mm$ for latent dimensions 8 and 32, respectively. Bednařík *et al.* [3], who use a different split than us, report an error of $21.48mm$. The authors of IsMo-GAN [32] report results on their own split for IsMo-GAN

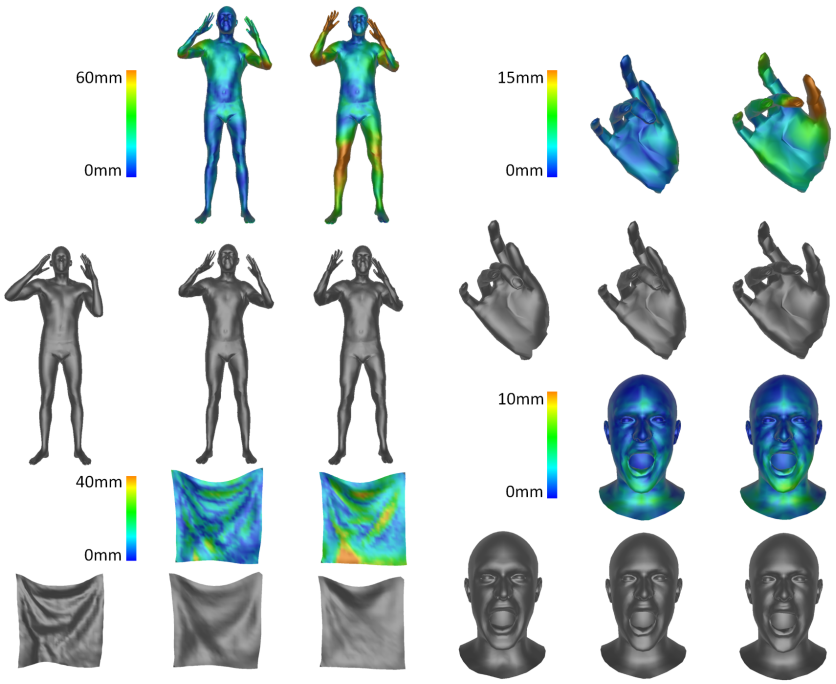
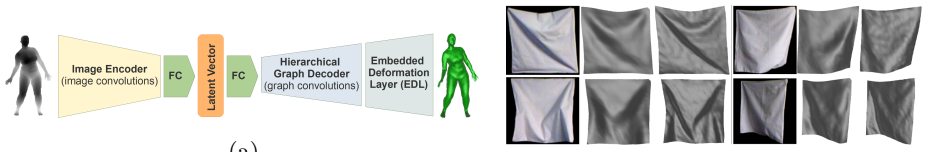


Fig. 4: Auto-encoding results on all four datasets. From left to right: ground-truth, ours with latent dimension 32, ours with latent dimension 8.



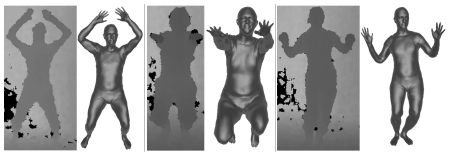
(a) (b)

Fig. 5: (left) Image/depth-to-mesh pipeline: To train an image/depth-to-mesh reconstruction network, we employ a convolutional image encoder and initialize the decoder to a pre-trained graph decoder. (right) RGB-to-mesh results on our test set. From left to right: real RGB image, our reconstruction, ground-truth.

and the Hybrid Deformation Model Network (HDM-net) [14]. On their split, HDM-Net achieves an error of 17.65mm after training for 100 epochs using a batch size of 4. IsMo-GAN obtains an error of 15.79mm. Under the same settings as HDM-Net, we re-train our approach without pre-training the mesh decoder. Our approach achieves test errors of 16.6mm and 13.8mm using latent dimensions of 8 and 32, respectively.

5.2 Depth to Mesh

Bodies. We train a network with a latent space dimension of 32. Quantitatively, we obtain an error of 2.3cm on un-augmented synthetic data. Besides, we also apply our approach to real data, see Fig. 6a. To this end, we found it necessary to



(a) DEMEA (right) on real Kinect depth images (left) with latent dimension 32.

(b) Reconstruction results from synthetic depth images of hands using a latent dimension of 32. From left to right: depth, our reconstruction, ground-truth.

Fig. 6: Reconstruction from a single depth image.

augment the depth images with artificial noise to lessen the domain gap. Video results are included in the supplementary.

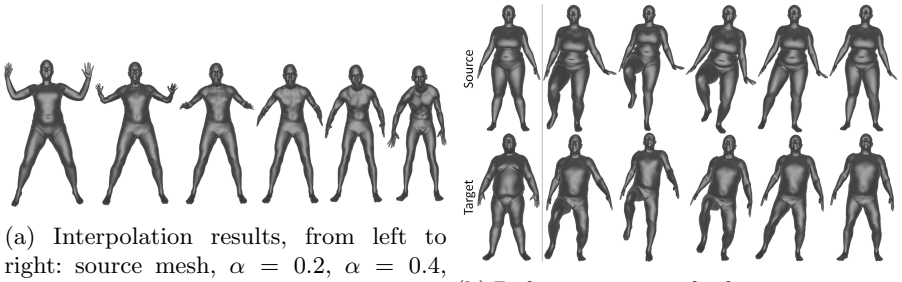
Hands. DEMEA can reconstruct hands from depth as well, see Fig. 6b. We achieve a reconstruction error of 6.73mm for a latent dimension of 32. Malik *et al.* [26] report an error of 11.8 mm. Our test set is composed of a random sample of fully randomly generated hands from the dataset, which is very challenging. We use 256×256, whereas [26] use images of size 96×96.

5.3 Latent Space Arithmetic

Although we do not employ any regularization on the latent space, we found empirically that the network learns a well-behaved latent space. As we show in the supplemental document and video, this allows DEMEA to temporally smooth tracked meshes from a depth stream.

Latent Interpolation. We can linearly interpolate the latent vectors $\mathcal{S}$ and $\mathcal{T}$ of a source and a target mesh: $\mathcal{I}(\alpha) = (1-\alpha)\mathcal{S} + \alpha\mathcal{T}$. Even for highly different poses and identities, these $\mathcal{I}(\alpha)$ yield plausible in-between meshes, see Fig. 7a.

Deformation Transfer. The learned latent space allows to transfer poses between different identities on DFaust. Let a sequence of source meshes $\mathbf{S} = \{\mathbf{M}_i\}_i$ of person A and a target mesh $\mathbf{M}'_0$ of person B be given, where w.l.o.g. $\mathbf{M}_0$ and $\mathbf{M}'_0$ correspond to the same pose. We now seek a sequence of target meshes



(a) Interpolation results, from left to right: source mesh, $\alpha = 0.2$, $\alpha = 0.4$, $\alpha = 0.6$, $\alpha = 0.8$, target mesh.

(b) Deformation transfer from a source sequence to a target identity. The first column shows $\mathbf{M}_0$ and $\mathbf{M}'_0$.

Fig. 7: Latent space arithmetic.

$\mathbf{S}' = \{\mathbf{M}'_i\}_i$ of person B performing the same poses as person A in $\mathbf{S}$. We encode $\mathbf{S}$ and M'_0 into the latent space of the mesh auto-encoder, yielding the corresponding latent vectors $\{\mathcal{M}_i\}_i$ and $\mathcal{M}'_0$. We define the identity difference $d = \mathcal{M}'_0 - \mathcal{M}_0$ and set $\mathcal{M}'_i = \mathcal{M}_i + d$ for $i > 0$. Decoding $\{\mathcal{M}'_i\}_i$ using the mesh decoder than yields $\mathbf{S}'$. See Fig. 7b and the supplement for qualitative results.

6 Limitations

While the embedded deformation graph excels on highly articulated, non-rigid motions, it has difficulties accounting for very subtle actions. Since the faces in the CoMA [31] dataset do not undergo large deformations, our EDL-based architecture does not offer a significant advantage. Similar to all other 3D deep learning techniques, our approach also requires reasonably sized mesh datasets for supervised training, which might be difficult to capture or model. We train our network in an object-specific manner. Generalizing our approach across different object categories is an interesting direction for future work.

7 Conclusion

We proposed DEMEA — the first deep mesh autoencoder for highly deformable and articulated scenes, such as human bodies, hands, and deformable surfaces, that builds on a new differentiable embedded deformation layer. The deformation layer reasons about local rigidity of the mesh and allows us to achieve higher quality autoencoding results compared to several baselines and existing approaches. We have shown multiple applications of our architecture including non-rigid reconstruction from real depth maps and 3D reconstruction of textureless surfaces from images.

Acknowledgments. This work was supported by the ERC Consolidator Grant 4DReply (770784), the Max Planck Center for Visual Computing and Communications (MPC-VCC), and an Oculus research grant.

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/
2. Bagautdinov, T., Wu, C., Saragih, J., Sheikh, Y., Fua, P.: Modeling facial geometry using compositional vaes (2018)
3. Bednařík, J., Fua, P., Salzmann, M.: Learning to reconstruct texture-less deformable surfaces. In: International Conference on 3D Vision (3DV) (2018)
4. Bogo, F., Romero, J., Pons-Moll, G., Black, M.J.: Dynamic FAUST: Registering human bodies in motion. In: Computer Vision and Pattern Recognition (CVPR) (2017)
5. Boscaini, D., Masci, J., Rodoià, E., Bronstein, M.: Learning shape correspondence with anisotropic convolutional neural networks. In: International Conference on Neural Information Processing Systems (NIPS) (2016)
6. Bouritsas, G., Bokhnyak, S., Ploumpis, S., Bronstein, M., Zafeiriou, S.: Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In: International Conference on Computer Vision (ICCV) (2019)
7. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. CoRR **abs/1312.6203** (2013)
8. Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G.: MeshLab: an Open-Source Mesh Processing Tool. In: Scarano, V., Chiara, R.D., Erra, U. (eds.) Eurographics Italian Chapter Conference. The Eurographics Association (2008). https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136
9. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: International Conference on Neural Information Processing Systems (NIPS) (2016)
10. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: Computer Vision and Pattern Recognition (CVPR) (2009)
11. Fuentes-Jimenez, D., Casillas-Perez, D., Pizarro, D., Collins, T., Bartoli, A.: Deep Shape-from-Template: Wide-Baseline, Dense and Fast Registration and Deformable Reconstruction from a Single Image. arXiv e-prints (2018)
12. Gao, L., Yang, J., Qiao, Y.L., Lai, Y.K., Rosin, P.L., Xu, W., Xia, S.: Automatic unpaired shape deformation transfer. ACM Transactions on Graphics (TOG) (2018)
13. Garland, M., Heckbert, P.S.: Surface simplification using quadric error metrics. In: ACM SIGGRAPH (1997)
14. Golyanik, V., Shimada, S., Varanasi, K., Stricker, D.: Hdm-net: Monocular non-rigid 3d reconstruction with learned deformation model. In: International Conference on Virtual Reality and Augmented Reality (EuroVR) (2018)
15. Groueix, T., Fisher, M., Kim, V.G., Russell, B., Aubry, M.: AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In: Computer Vision and Pattern Recognition (CVPR) (2018)

16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Computer Vision and Pattern Recognition (CVPR) (2016)
17. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science (2006)
18. Jack, D., Pontes, J.K., Sridharan, S., Fookes, C., Shirazi, S., Maire, F., Eriksson, A.: Learning free-form deformations for 3d object reconstruction. In: Asian Conference on Computer Vision (ACCV) (2018)
19. Kanazawa, A., Tulsiani, S., Efros, A.A., Malik, J.: Learning category-specific mesh reconstruction from image collections. In: European Conference on Computer Vision (ECCV) (2018)
20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (ICLR) (2015)
21. Kurenkov, A., Ji, J., Garg, A., Mehta, V., Gwak, J., Choy, C., Savarese, S.: Deformnet: Free-form deformation network for 3d shape reconstruction from a single image. In: Winter Conference on Applications of Computer Vision (WACV) (2018)
22. Li, H., Adams, B., Guibas, L.J., Pauly, M.: Robust single-view geometry and motion reconstruction. In: ACM SIGGRAPH Asia (2009)
23. Litany, O., Bronstein, A., Bronstein, M., Makadia, A.: Deformable shape completion with graph convolutional autoencoders. In: Computer Vision and Pattern Recognition (CVPR) (2018)
24. Loper, M., Mahmood, N., Black, M.J.: Mosh: Motion and shape capture from sparse markers. ACM Transactions on Graphics (TOG) (2014)
25. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: Smpl: A skinned multi-person linear model. ACM Transactions on Graphics (TOG) (2015)
26. Malik, J., Elhayek, A., Nunnari, F., Varanasi, K., Tamaddon, K., Héloir, A., Stricker, D.: Deephps: End-to-end estimation of 3d hand pose and shape by learning from synthetic depth. International Conference on 3D Vision (3DV) (2018)
27. Masci, J., Boscaini, D., Bronstein, M.M., Vandergheynst, P.: Geodesic convolutional neural networks on riemannian manifolds. In: International Conference on Computer Vision Workshop (ICCVW) (2015)
28. 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: Computer Vision and Pattern Recognition (CVPR) (2017)
29. Niepert, M., Ahmed, M., Kutzkov, K.: Learning convolutional neural networks for graphs. In: International Conference on Machine Learning (ICML) (2016)
30. Pumarola, A., Agudo, A., Porzi, L., Sanfeliu, A., Lepetit, V., Moreno-Noguer, F.: Geometry-aware network for non-rigid shape prediction from a single view. In: Computer Vision and Pattern Recognition (CVPR) (2018)
31. Ranjan, A., Bolkart, T., Sanyal, S., Black, M.J.: Generating 3D faces using convolutional mesh autoencoders. In: European Conference on Computer Vision (ECCV) (2018)
32. Shimada, S., Golyanik, V., Theobalt, C., Stricker, D.: Ismo-gan: Adversarial learning for monocular non-rigid 3d reconstruction. In: Computer Vision and Pattern Recognition Workshops (CVPRW) (2019)
33. Sinha, A., Unmesh, A., Huang, Q., Ramani, K.: Surfnet: Generating 3d shape surfaces using deep residual networks. In: Computer Vision and Pattern Recognition (CVPR) (2017)
34. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Eurographics Symposium on Geometry Processing (SGP) (2007)
35. Sumner, R.W., Schmid, J., Pauly, M.: Embedded deformation for shape manipulation. In: ACM SIGGRAPH (2007)

36. Tan, Q., Gao, L., Lai, Y.K., Xia, S.: Variational autoencoders for deforming 3d mesh models. In: Computer Vision and Pattern Recognition (CVPR) (2018)
37. Tan, Q., Gao, L., Lai, Y.K., Yang, J., Xia, S.: Mesh-based autoencoders for localized deformation component analysis. In: AAAI Conference on Artificial Intelligence (AAAI) (2018)
38. Verma, N., Boyer, E., Verbeek, J.: FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis. In: Computer Vision and Pattern Recognition (CVPR) (2018)
39. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single rgb images. In: European Conference on Computer Vision (ECCV) (2018)
40. Yi, L., Su, H., Guo, X., Guibas, L.: Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In: Computer Vision and Pattern Recognition (CVPR) (2017)
41. Zollhöfer, M., Nießner, M., Izadi, S., Rhemann, C., Zach, C., Fisher, M., Wu, C., Fitzgibbon, A., Loop, C., Theobalt, C., Stamminger, M.: Real-time non-rigid reconstruction using an rgb-d camera. ACM Transactions on Graphics (TOG) (2014)