Eurographics Symposium on Geometry Processing 2021
K. Crane and J. Digne
(Guest Editors)

Volume 40 (2021), Number 5

Learning Direction Fields for Quad Mesh Generation

Alexander Dielen Isaak Lim Max Lyon Leif Kobbelt

Visual Computing Institute, RWTH Aachen University

Abstract

State of the art quadrangulation methods are able to reliably and robustly convert triangle meshes into quad meshes. Most of these methods rely on a dense direction field that is used to align a parametrization from which a quad mesh can be extracted. In this context, the aforementioned direction field is of particular importance, as it plays a key role in determining the structure of the generated quad mesh. If there are no user-provided directions available, the direction field is usually interpolated from a subset of principal curvature directions. To this end, a number of heuristics that aim to identify significant surface regions have been proposed. Unfortunately, the resulting fields often fail to capture the structure found in meshes created by human experts. This is due to the fact that experienced designers can leverage their domain knowledge in order to optimize a mesh for a specific application. In the context of physics simulation, for example, a designer might prefer an alignment and local refinement that facilitates a more accurate numerical simulation. Similarly, a character artist may prefer an alignment that makes the resulting mesh easier to animate. Crucially, this higher level domain knowledge cannot be easily extracted from local curvature information alone. Motivated by this issue, we propose a data-driven approach to the computation of direction fields that allows us to mimic the structure found in existing meshes, which could originate from human experts or other sources. More specifically, we make use of a neural network that aggregates global and local shape information in order to compute a direction field that can be used to guide a parametrization-based quad meshing method. Our approach is a first step towards addressing this challenging problem with a fully automatic learning-based method. We show that compared to classical techniques our data-driven approach combined with a robust model-driven method, is able to produce results that more closely exhibit the ground truth structure of a synthetic dataset (i.e. a manually designed quad mesh template fitted to a variety of human body types in a set of different poses).

CCS Concepts

• ***Computing methodologies*** → *Shape analysis; Neural networks; Mesh models;*

1. Introduction

Mesh quadrangulation, i.e. the process of converting a given triangle mesh into a quadrilateral mesh, is a fundamental problem in computer graphics and geometry processing with applications in character animation and physics simulation. Unfortunately, quadrangulating a given triangle mesh by hand is both labor-intensive and cumbersome, as it requires a user to manually place individual quads on the surface of the input mesh. Because of this, a number of authors have proposed fully automatic quadrangulation techniques [BZK09; BCE*13; CBK15; JTPS15].

These fully automatic techniques work well on input shapes for which a meaningful alignment of quads can be computed from local curvature information. They do, however, encounter problems when faced with shapes that do not offer strong curvature guidance. The problem is exacerbated by the fact that, depending on the application, a designer might prefer an alignment that is not directly related to principal curvature directions. In the context of character modeling and animation, for example, artists usually place additional edge loops around surface regions that are likely to deform, such as the eyes or mouth of a human character. Furthermore, irregular vertices are often placed in approximately planar regions in order to hide visual artifacts. For numerical simulation on quad meshes, designers align and specify different sizes of quads based on their expert knowledge on how simulation solvers behave. It is unclear how exactly the local curvature information relates to the expert knowledge in these various domains.

To address this issue, a number of authors have proposed methods that incorporate user-guidance into the remeshing process. The method presented in [TPSS13], for example, enables users to define patch layouts using a sketch-based interface. Similarly, the techniques described in [JTPS15; ESCK16] allow users to override the edge flow of an automatically generated quad mesh using brush-strokes. A common drawback of these methods is that querying a user repeatedly for guidance can be somewhat time-consuming.

We present a data-driven approach that does not require any user-guidance. Our method learns the structure present in exist-

ing meshes, which allows for the generation of quad meshes that exhibit many features typically reserved to manually created ones. To this end, we combine a field-guided quadrangulation technique with a neural network that infers direction fields from unstructured triangle meshes. We prefer these continuous outputs instead of directly inferring a quad mesh from a neural network, since this would require the network to make a series of discrete decisions (each of which can have catastrophic effects on the final output). A state-of-the-art quadrangulation method can then robustly compute a quad mesh from the triangle mesh and the network inferred direction field.

Contribution We investigate which type of direction field best captures the structure found in manually created quad meshes. Based on our findings, we propose a neural network that infers frame fields from unstructured triangle meshes. Furthermore, we present a number of loss functions that can be used to train our network. We demonstrate the applicability of our approach on the challenging task of remeshing human characters models and compare our results with those obtained using three existing curvature-based methods. Our experiments show that our method performs favorably. Using an ablation study, we validate our design choices for the neural network and the losses.

2. Related Work

Geometric Deep Learning Neural Networks that work on different 3D shape representations as input are well established. Input representations and corresponding network architectures range from learning on 2D maps of 3D shapes [SMKL15; SBR16; MGA*17], over (sparse) voxel representations [MS15; GEvdM18; WSLT18], and point set methods [ZKR*17; QSMG17; QYSG17; AML18; TQD*19], to techniques that treat 3D shapes as graphs [WSS18; WSL*19; HHF*19] or learn on curved surfaces directly [MBBV15; MBM*17; FLWM18]. We cannot give a complete overview here and refer to [BBL*17; XLZ*20] for more detailed surveys. In this work we encode global and local shape information with architectures based on PointNet [QSMG17] and SpiralNet [LDCK18] respectively.

Quad Meshing Methods The computer aided generation of quad meshes has been an extensively researched topic in the past years. Methods range from fully automatic pipelines to interactive ones that require a user to specify most of the quad mesh geometry and connectivity by hand. A great overview of existing methods can be found in [BLP*13].

Of particular interest are field guided quad meshing algorithms [KNP07; BZK09; BCE*13; CBK15; JTPS15; HZN*18; FBT*18; LCBK19] which yield high quality results by dividing the process into two steps. In the first step a guiding field is generated which specifies the position and degrees of irregular vertices as well as the desired orientation and sizing of the resulting quad elements. The second step computes a parametrization that aligns its gradients with the specified directions of the guiding field and whose integer iso-lines define the edges of the resulting quad mesh. The quality of the results depends largely on the guiding field for which many methods have been proposed [BZK09; KCPS13; CIE*16; PPTS14; JFH*15; DVPS15]. All these methods have in common that the desired alignment is derived from the surface geometry – mainly its curvature. This works well on shapes where alignment to principal curvature is sufficient. For cases where a user may desire different alignment, user input can be considered during field generation. This works particularly well with the methods proposed in [ESCK16; JTPS15] that provide quick results enabling interactive workflows. These methods allow the user to manually place singular vertices, and to specify general edge alignment or even explicit edge loops connecting the singularities. Due to the robustness and reliability of these methods as well as the high quality of the results that can be achieved by them, we make use of such a field guided quad meshing method [CBK15] in this work.

Other interactive quad meshing methods require the user to partition the surface into patches which are then filled with suitable quad grids [NSY09; TPSS13; PBJW14; TPS14]. CAMPEN et al. propose in [CK14] a quad meshing algorithm which requires the user to specify the dual loops of the desired mesh. These interactive methods have in common that, while they do provide detailed and explicit control of the resulting mesh, they also *require* extensive user input which increases the overall time required to generate quad meshes.

Data-driven Remeshing The data-driven quadrangulation method by MARCIAS et al. in [MTP*15] is related to this work in the sense that they also aim to extract domain knowledge encoded in existing quad meshes for the alignment and placements of quads on input shapes. However, they only extract and compare individual patches of quads. Furthermore, their method requires the user to specify the boundaries of these patches manually, which can then be filled with quads automatically. In contrast our method takes the complete shape into account and requires no user interaction. More recently, a number of deep learning based approaches have been presented that guide remeshing processes. In order to learn mesh-based simulation PFAFF et al. learn a sizing field for a given input triangle mesh for adaptive refinement in [PFSB20]. In our work we do not just learn a sizing field but a frame field which incorperates both sizing and directional information for the purpose of quadmeshing. LIU et al. learn the position of new vertices created by a subdivision step in [LKC*20]. Their method only has to consider local shape information for the subdivision scheme. In contrast, our network has to consider both local and global shape information since a correct and coherent alignment and sizing of quads also depends on the global properties (e.g. symmetry).

The task of learning a frame field has been investigated by GIRARD et al. in [GSST20]. In their work, GIRARD et al. train a neural network that infers a frame field from satellite images in order to regularize the segmentation of buildings in the images and extraction of 2D polygons that describe their contours. In this work we consider frame fields on 3D shapes, while their method predicts frame fields on a 2D regular grid.

3. Learning Direction Fields

Given an unstructured triangle mesh T, our goal is to generate a quad mesh Q that not only represents the same shape as T, but

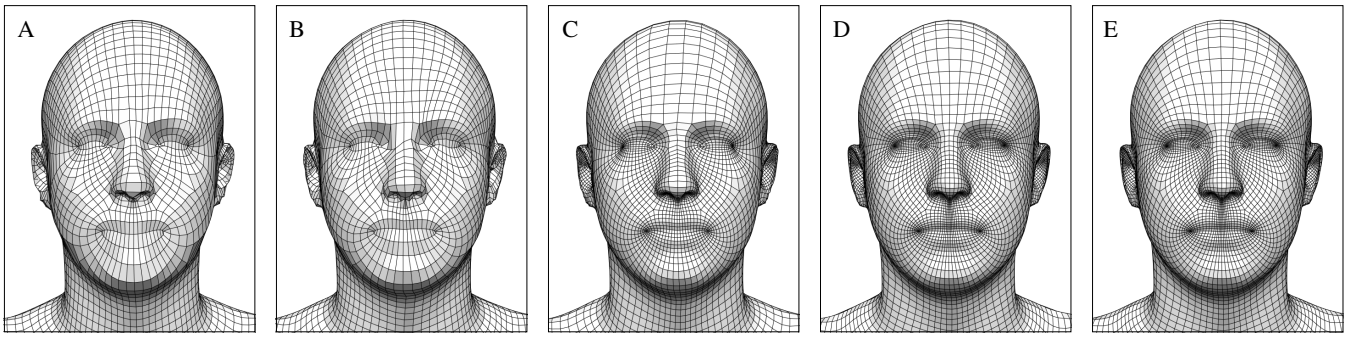

Figure 1: *Quad meshes generated using different types of direction fields. From left to right: unit cross field, unit frame field, cross field, frame field, ground truth.*

also exhibits the structure found in meshes created by domain experts. Directly outputting such a quad mesh is a non-trivial task that would require the network to generate a valid mesh topology which is difficult due to global consistency requirements. As a consequence, we propose using a neural network to compute a direction field that can be fed to an existing field-guided parametrization-based quadrangulation technique (cf. Section 2). For these direction fields no global consistency requirements exist. While the computation of e.g. an *integrable* direction field would be desirable as it would enable the meshing algorithm to compute a perfectly aligned quad mesh, this is not strictly necessary and the algorithm will simply produce a quad mesh that is aligned to the field as much as possible. Thus, this approach allows us to control the structure of the resulting mesh Q using a direction field that can be more easily inferred by a network operating on the input mesh T.

3.1. Representation

The question arises as to which type of direction field should be used to specify the desired size and orientation of the to be generated quads. Most commonly, this is done using a (possibly scaled) cross field [BZK09; KCPS13] or frame field [PPTS14; JFH*15; DVPS15]. To determine which of these fields is most suitable for our task at hand, we quadrangulate a given triangle mesh T using different ground truth fields and compare the resulting quad meshes. More specifically, we use the Skinned Multi-Person Linear Model (SMPL) [LMR*15] as a representative input shape and consider both unit and non-unit cross fields and frame fields. To obtain a mesh Q that contains only quads, we apply one iteration of Catmull-Clark subdivision [CC78] to the quad-dominant SMPL mesh. The corresponding triangle mesh T is generated by splitting each quad of Q into two triangles. To compute two ground truth directions u, v for a given face f of T, we first project the barycenter p of f onto the surface of Q. Let p' be the projection of p onto Q and f' be the face of Q that contains p'. We use p' to compute the shortest distances d_1, d_2, d_3, d_4 of p' to the edges e_1, e_2, e_3, e_4 depicted in Figure 2 (left). The distances d_1, d_2, d_3, d_4 are used to interpolate between the edges of f', i.e. the ground truth frame field directions u and v are obtained as follows:

$$u = \frac{d_3}{d_1+d_3}e_1 + \frac{d_1}{d_1+d_3}e_3, \quad v = \frac{d_4}{d_2+d_4}e_2 + \frac{d_2}{d_2+d_4}e_4$$

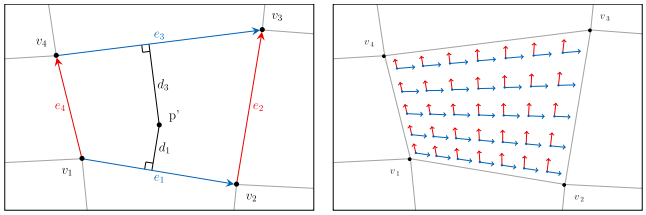

Figure 2: *The vertices and edges used to compute a ground truth frame field (left) and the resulting frame field vectors for different sample points p' (right).*

To obtain a cross field, we orthogonalize the vectors u and v for every face of T. To obtain a unit field, we normalize both u and v.

Quadrangulation results are shown in Figure 1. As expected, the unit direction fields (A, B) lead to meshes with significantly more uniform edge lengths than their non-unit counterparts (C, D). This is of course due to the fact that a unit direction field cannot represent varying magnitudes and therefore forces all generated quads to be approximately the same size. Regarding the results obtained using the two non-unit fields (C, D), we observe that both of these fields lead to meshes that closely resemble the ground truth mesh (E). However, the inability of cross fields (C) to represent anisotropic quad sizes introduces some distortion, which is particularly evident in the quads located on the nose and forehead. As a consequence, we conclude that frame fields (D) are best suited to capture the structure found in quad meshes created by professional artists.

3.2. Network Architecture

Motivated by the results presented in the previous section, our proposed network should infer frame fields from unstructured triangle meshes. Previous work on field-guided quadrangulation techniques has highlighted the importance of aligning direction fields with principal curvature directions and other local surface features such as sharp edges. We therefore believe that our proposed network should have access to the local geometry of the input mesh T. That being said, some of the most noticeable characteristics of

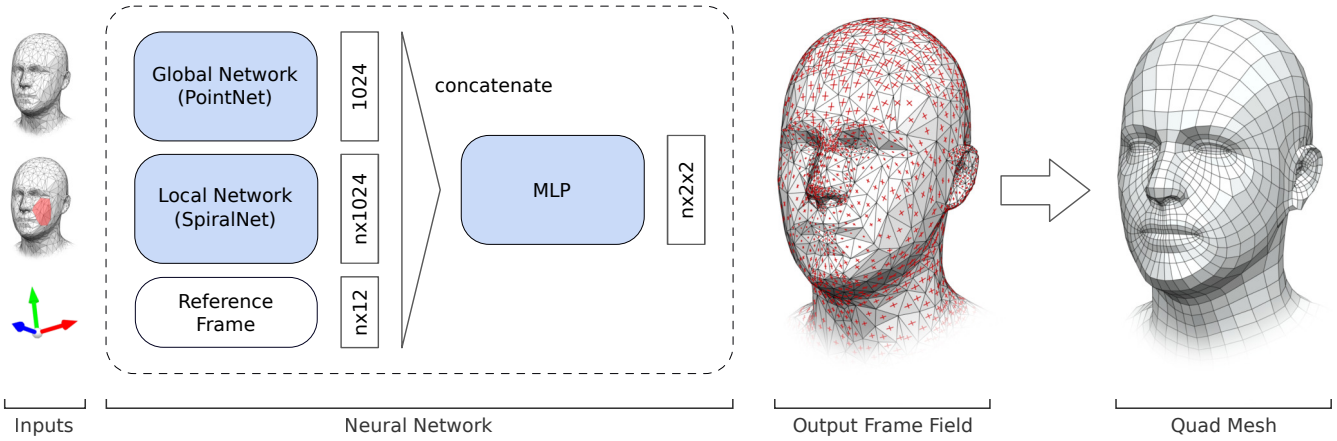

Figure 3: *Our network combines information from three sources to infer a frame field for a given input mesh: a global network that operates on a point cloud representing the entire input shape, a local network that operates on patches centered around individual triangles for which a frame is to be inferred, and a set of reference frames that describe the position and orientation of the aforementioned triangles. In the illustration above, n denotes the number of triangles in a batch and every batch originates from a single input mesh, which allows us to run the global network only once per batch. The network output with shape $n \times 2 \times 2$ is interpreted as a pair of 2D vectors for each of the n input triangles. These vectors are subsequently used to guide the parametrization-based quadrangulation method described in [CBK15].*

manually created quad meshes cannot be explained using local surface properties alone. An example of this are the edge loops placed by character artists around surface regions that are likely to deform.

As a consequence, our proposed network follows a dual strategy that is based on the idea of combining both local and global shape information. More specifically, we combine the outputs of a local feature network L and a global feature network G using a Multi-Layer Perceptron (MLP) P. As can be seen in Figure 3, we also provide P with a reference frame that describes the position and orientation of every input triangle. The primary purpose of these frames is to provide the network with a set of local coordinate systems in which the output directions should be expressed (cf. [VCD*16, §5.2]). Specifically, the reference frame for a face f consists of the barycenter $p \in \mathbb{R}^3$ (in absolute coordinates) and the unit vectors $x_f, y_f, n_f \in \mathbb{R}^3$ of f, where x_f corresponds to one of the edges of f, n_f is the normal of the supporting plane of f and $y_f = n_f \times x_f$. The output of P is a frame field represented using two vectors $u_f, v_f \in \mathbb{R}^2$ for every input triangle f, where both u_f and v_f are interpreted w.r.t. the local tangent frame defined by x_f and y_f. Since we assume a dataset of aligned shapes, we forego special consideration of rotational invariance for our reference frames and networks.

The global network G is based on the PointNet architecture presented by Qi et al. in [QSMG17]. Following Qi et al., we use a sequence of five fully-connected network layers with ReLU [NH10] activation functions. However, unlike Qi et al., we do not use either of the two alignment networks described in [QSMG17], as they do not appear to have a measurable effect on the performance of our network due to the alignment of training shapes. The input to G consists of the 32 000 vertices of the input mesh T, where each vertex is represented using its 3D position and normal.

The local network L is based on the SpiralNet architecture that was originally presented by Lim et al. in [LDCK18]. In their work, Lim et al. propose to encode the neighborhood of a given vertex v_0 using a sequence of vertices $[v_0, v_1, \ldots, v_{k-1}]$ that extends outwards from v_0 in a spiral manner until a predetermined number of vertices k have been enumerated. The obtained sequences are then mapped to a new representation using either an LSTM [HS97] cell or a fully-connected network layer. We process sequences using fully-connected layers, since the resulting network can be trained significantly faster and is only marginally less powerful. More specifically, the network L consists of four spiral layers. Each layer takes a sequence of $k = 20$ vertices as input and produces an intermediate representation consisting of 16, 256, 512 and 1024 features respectively. The first three layers use the spiral sequences to compute feature vectors for each mesh vertex. Since our goal is to compute a feature vector for every triangle of the input mesh (in order to output a frame per face), the last layer operates on sequences that are centered around the triangles. To center a sequence around a given triangle, we set the first three sequence elements to the vertices that make up the triangle in question and then extend this sequence using the method described in [LDCK18]. Analogous to the global network G, we represent each input vertex using its 3D position and normal, i.e. the first layer of L operates on sequences of 6-dimensional point features.

Furthermore, we subtract the center element of every sequence from all remaining sequence elements. Thus, given a sequence $[v_0, v_1, \ldots, v_{k-1}]$ of length k that is centered around a vertex v_0, we instead use the modified sequence $[v_0, v_1 - v_0, \ldots, v_{k-1} - v_0]$. If the vertices in a sequence are represented using their positions and normals, as in the case of the vertices in the bottommost spirals, we apply the centering only to the positions and leave the normal vectors as they are. For sequences that are centered around a triangle f, we subtract the barycenter $p = \frac{1}{3}(v_0 + v_1 + v_2)$, where v_0, v_1, v_2 are the three vertices that make up f, from all sequence elements.

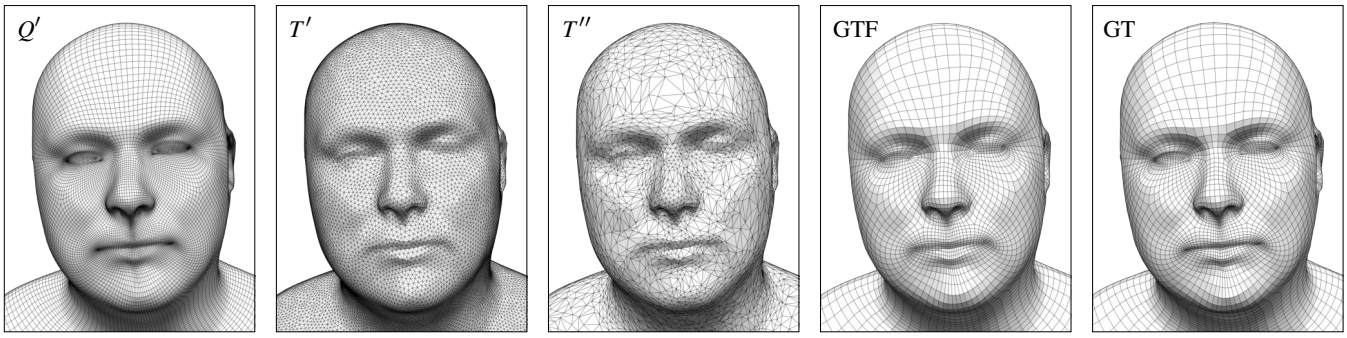

Figure 4: *An example of our remeshing procedure that ensures that the edges of a mesh fed into our network are not aligned with the corresponding ground truth quad mesh. Also shown is a visualization of the expected reconstruction error caused by our data generation process. From left to right: the twice subdivided quad mesh Q', the isotropically remeshed triangle mesh T', the decimated triangle mesh T'' that is fed into our network, a quad mesh that was reconstructed from T'' using a ground truth frame field (GTF), the actual ground truth mesh (GT).*

3.3. Loss Functions

To ensure that our loss function $\mathcal{L}$ penalizes both direction and size errors in a balanced manner, we split $\mathcal{L}$ into a direction loss $\mathcal{L}_d$ and a size loss $\mathcal{L}_s$:

$$\mathcal{L}(u,v,\hat{u},\hat{v}) = \mathcal{L}_d(u,v,\hat{u},\hat{v}) + \mathcal{L}_s(u,v,\hat{u},\hat{v}) \tag{1}$$

Our direction loss is based on the von Mises distribution [MJ09] that approximates a normal distribution wrapped around a unit circle. In [BHL15], BEYER et al. use the von Mises distribution to formulate a loss for angle-based outputs θ and targets $\hat{\theta}$ that is invariant w.r.t. rotations of 2π radians:

$$\mathcal{L}_{\text{vm}}(\theta,\hat{\theta}) = 1 - e^{\kappa(\cos(\theta-\hat{\theta})-1)} \tag{2}$$

Since a frame can be seen as two independent directions with π symmetry each, we modify the loss in Equation (2) by doubling the frequency of the cosine term. This makes the resulting loss invariant w.r.t. rotations of π radians:

$$\mathcal{L}^{2x}_{\text{vm}}(\theta,\hat{\theta}) = 1 - e^{\kappa(\cos(2(\theta-\hat{\theta}))-1)} \tag{3}$$

Furthermore, we adapt $\mathcal{L}^{2x}_{\text{vm}}$ to vector-based network outputs and targets using the double angle formula:

$$\mathcal{L}^{2x}_{\text{vm}}(y,\hat{y}) = 1 - e^{\kappa(2(y\cdot\hat{y})^2-2)}, \tag{4}$$

where both y and $\hat{y}$ are assumed to have unit length. We formulate a per-triangle loss using the sum of two evaluations of $\mathcal{L}^{2x}_{\text{vm}}$:

$$\mathcal{L}^{2x}_{\text{vm}}(u,v,\hat{u},\hat{v}) = \mathcal{L}^{2x}_{\text{vm}}(u,\hat{u}) + \mathcal{L}^{2x}_{\text{vm}}(v,\hat{v}) \tag{5}$$

As an alternative, we can also formulate a direction loss that is based on the representation of frame fields using unit complex numbers [KCPS13; DVPS15]. Interpreting a network output y and its corresponding ground truth target $\hat{y}$ as unit complex numbers has the advantage that squaring both y and $\hat{y}$ removes any ambiguity w.r.t. their orientation, i.e. $y^2 = (-y)^2$ and similarly $\hat{y}^2 = (-\hat{y})^2$. We use this property to formulate a *complex cosine* loss that measures the cosine similarity of y^2 and $\hat{y}^2$:

$$\mathcal{L}_{\text{cc}}(y,\hat{y}) = 1 - (y^2 \cdot \hat{y}^2), \tag{6}$$

where both y and $\hat{y}$ are assumed to be normalized. Analogous to Equation (5), the complex cosine loss for a given triangle can then be expressed as follows:

$$\mathcal{L}_{\text{cc}}(u,v,\hat{u},\hat{v}) = \mathcal{L}_{\text{cc}}(u,\hat{u}) + \mathcal{L}_{\text{cc}}(v,\hat{v}), \tag{7}$$

For our size loss $\mathcal{L}_s$, we compute the absolute difference between the magnitudes of the network outputs u,v and the target vectors $\hat{u},\hat{v}$:

$$\mathcal{L}_{\text{abs}}(u,v,\hat{u},\hat{v}) = |\|u\| - \|\hat{u}\|| + |\|v\| - \|\hat{v}\|| \tag{8}$$

Alternatively, we also consider a relative size loss that expresses the same quantity w.r.t. the magnitudes of the target vectors $\hat{u},\hat{v}$:

$$\mathcal{L}_{\text{rel}}(u,v,\hat{u},\hat{v}) = \left|\frac{\|u\| - \|\hat{u}\|}{\|\hat{u}\|}\right| + \left|\frac{\|v\| - \|\hat{v}\|}{\|\hat{v}\|}\right| \tag{9}$$

During training, we match the network outputs u,v and ground truth targets $\hat{u},\hat{v}$ using $\min(\mathcal{L}(u,v,\hat{u},\hat{v}),\mathcal{L}(v,u,\hat{u},\hat{v}))$. This is necessary because the network cannot know the expected order of its outputs. Furthermore, we compute the loss for an entire batch of triangles using the mean of the loss function defined above.

4. Dataset

We train our network on the meshes of the Dynamic FAUST (DFAUST) [BRPB17] dataset that consists of 10 human subjects performing a variety of motions. Strictly speaking, the DFAUST dataset consists of triangle meshes. However, each of these triangle meshes was created by registering the quad-dominant SMPL [LMR*15] mesh with a given 3D scan and then splitting each quad into two triangles. As a consequence, we can recover the original quad meshes by combining the topology of the SMPL mesh with the vertex coordinates of the DFAUST meshes. Subsequently applying one iteration of Catmull–Clark subdivision [CC78] to the recoverd meshes gives us a set of more than 40 000 meshes that contain only quads.

To ensure that our network does not simply learn to align its outputs with the edges of the input mesh, we remesh every input

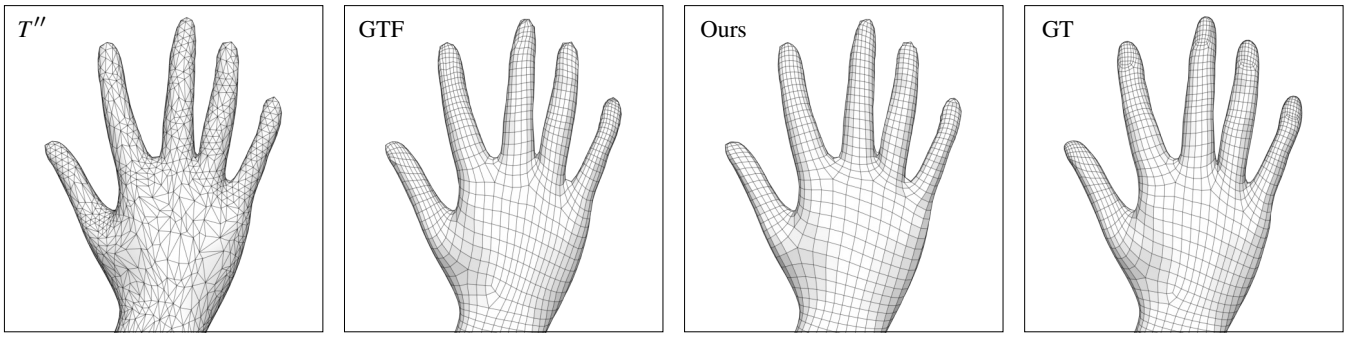

Figure 5: *An illustration of the reconstruction error incurred by our method as a result of our data generation process. From left to right: the remeshed network input (T''), a quad mesh reconstructed for T'' using the ground truth frame field (GTF), a quad mesh reconstructed for T'' using our inferred frame field (Ours), the actual ground truth quad mesh (GT). As can be seen, some of the more intricate features such as the edge loops around the fingernails cannot be reconstructed using our input triangle mesh and ground truth frame field. Despite this, our network is able to infer similar sizes and directions.*

mesh as follows: Given a ground truth quad mesh Q, we first apply two iterations of Catmull–Clark subdivision [CC78] to Q in order to generate a higher resolution mesh Q'. Next, we triangulate Q' to obtain a triangle mesh T. The mesh T is then remeshed using the isotropic remeshing method described in [BK04] and a sufficently small target edge length (0.004 in our experiments). As illustrated in Figure 4 (left, center left), remeshing T ensures that the resulting mesh T' no longer contains any edge of Q or Q'. To reduce the computational costs associated with processing T', we decimate T' to 64 000 triangles using the incremental decimation approach presented in [KCS98]. The resulting mesh T'' is shown in Figure 4 (center) and the corresponding ground truth frame field is computed using the quad mesh Q and the method described in Section 3.1. Note, that remeshing Q and then projecting the barycenters of T'' back onto Q can reduce the quality of the generated ground truth frame field. This can be due to a coarser triangle mesh T'' (offering fewer degrees of freedom) or slight projection inaccuracies. However, as shown in Figures 4 and 5, the quad meshes generated using these fields still exhibit most of their original structure.

We split the DFAUST dataset into a training, validation and test set such that the test set does not contain any subject that is also included in the training and/or validation set. This addresses the issue that many sequences include meshes where the subject assumes an approximately neutral pose. As a consequence, the inclusion of the same subject in both the test and training set, would lead to a significant overlap that is likely to distort our evaluation. Furthermore, the test set should contain both a male and a female subject in order to be approximately representative of both the original dataset and the range of expected network inputs. However, withholding two subjects for testing purposes reduces the amount of available training meshes significantly. Because of this, we select the male and female subjects with the smallest number of meshes, namely the subjects with the IDs `50009` and `50020`, to make up our test set. For our validation set we opt to withhold a subset of sequences of the remaining subjects. The withheld sequences should include a sufficiently large range of motions, but should not restrict the variety of poses in the remaining meshes too much. We find the `light_hopping_stiff` and `one_leg_loose` sequences to fulfill these requirements and therefore select them to form our validation set.

5. Results

To evaluate our proposed approach, we trained the network described in Section 3.2 for 130 epochs on the approximately 31 000 training examples of the DFAUST dataset. More specifically, we used the Adam [KB15] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a learning rate of 6.25×10^{-5}. After each epoch, we reduced the learning rate by a factor of 0.95. Furthermore, we inferred all frame fields using the network weights that performed best on our validation set and generated the quad meshes shown throughout this section using the inferred fields and the quadrangulation technique described in [CBK15].

As can be seen in Figure 6, our approach produces meshes that capture many semantic features. The mouth, eyes and ears, for example, are all clearly defined and easily recognizable across a wide variety of input shapes, which is of course due to the fact that the meshes used to train our network exhibit these characteristics as well. As a consequence, our approach also generates meaningful topology for other input regions, such as the upper bodies and backs of human characters. For this to work, the frame fields inferred using our network have to induce singular vertices in appropriate locations on the respective input shapes. We find that this works particularly well in regions where singularities are located far apart, but could be improved in regions where many singularities are located in close proximity.

We also evaluate our method on the FAUST [BRLB14] dataset that contains a number of subjects not included in the larger DFAUST dataset. To this end, we first remesh the FAUST meshes using the method described in Section 4 and then generate frame fields and quad meshes using our network trained on the DFAUST dataset. Results are shown in Figure 7. As can be seen, our network generalizes quite well to these inputs.

For shapes that are increasingly different from our training set, we show quad meshes computed based on the inferred frame fields

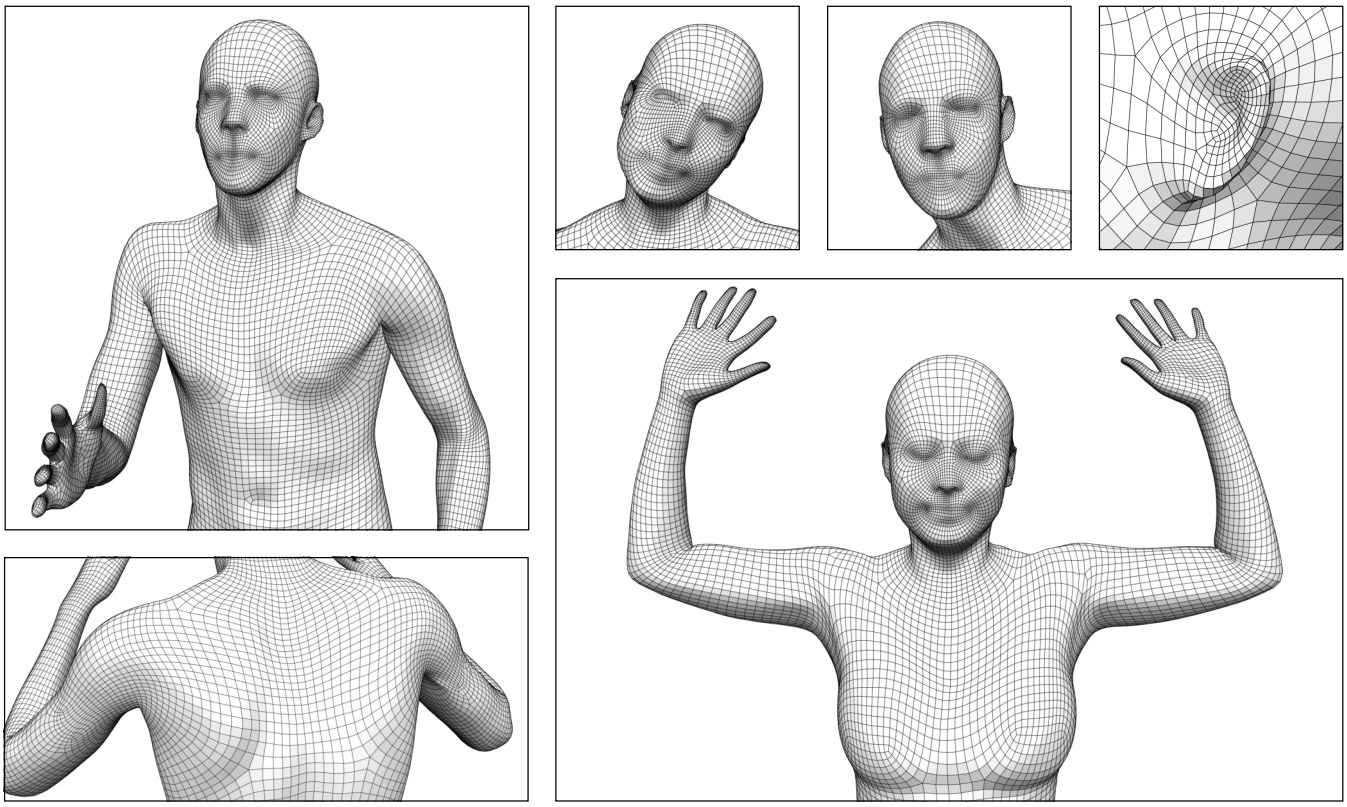

Figure 6: *Examples of quad meshes generated using frame fields inferred by our network. As can be seen in the images above, the quad meshes generated using our approach capture many semantic features, such as the eyes, ears, and mouths of the depicted subjects.*

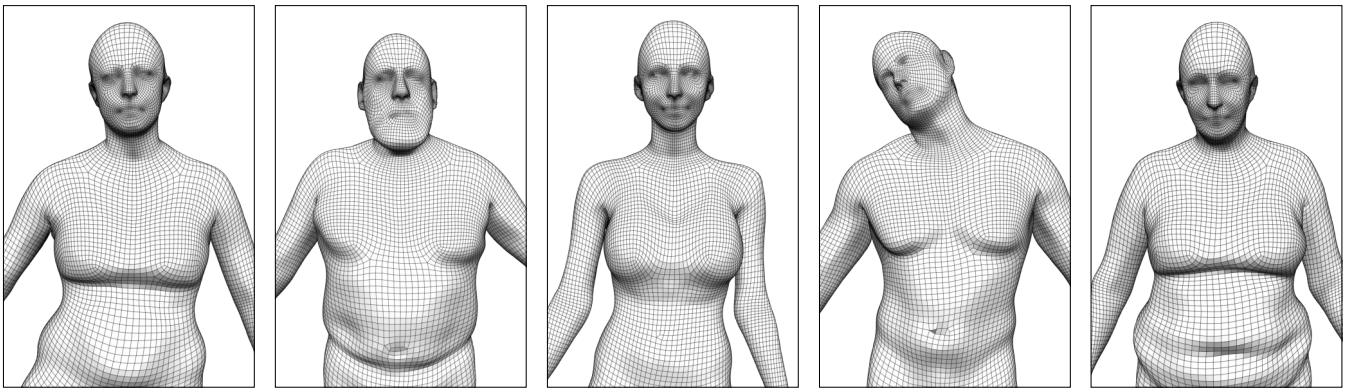

Figure 7: *Examples of quad meshes generated using our approach for triangle meshes from the FAUST [BRLB14] dataset. Just as in Figure 6, our network has never seen these subjects during training.*

in Figure 8. On the left we show our result on a realistic model of a child (note that our training set consists of adults only). Here the resulting quad mesh is of a similar quality to the ones obtained on our test set. For the more stylized character (center left) with fairly exaggerated proportions, the quad mesh still resembles one an artist might produce. For an even more unrealistic humanoid cartoon character (center right) the quads on the arms, legs, and upper body are well aligned. However, due to the very different facial proportions compared to the realistic faces in DFAUST, the edge flow in that area is less pleasing. On the right a completely different shape is shown in the form of a mechanical component. Here the inferred frame field is least likely to align with one a user might design. Nevertheless, in certain regions the network does produce frames aligned with principal curvature directions. This suggests that curvature information (among other shape properties) is taken into account by the network. Of course, the network is not expected

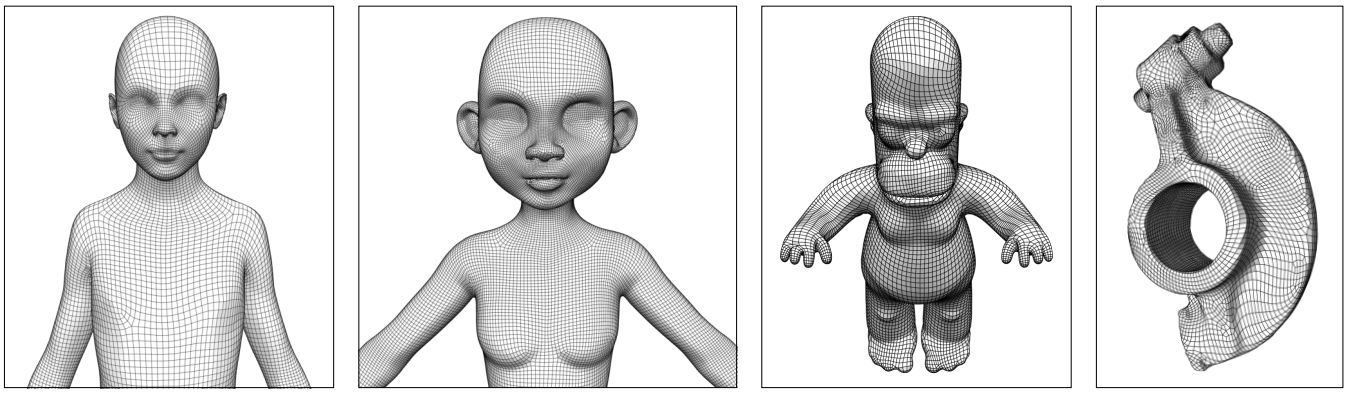

Figure 8: *Examples of quad meshes generated using our approach for triangle meshes with varying difference from the training set. From left to right: a child, a stylized character, a cartoon character and a mechanical component.*

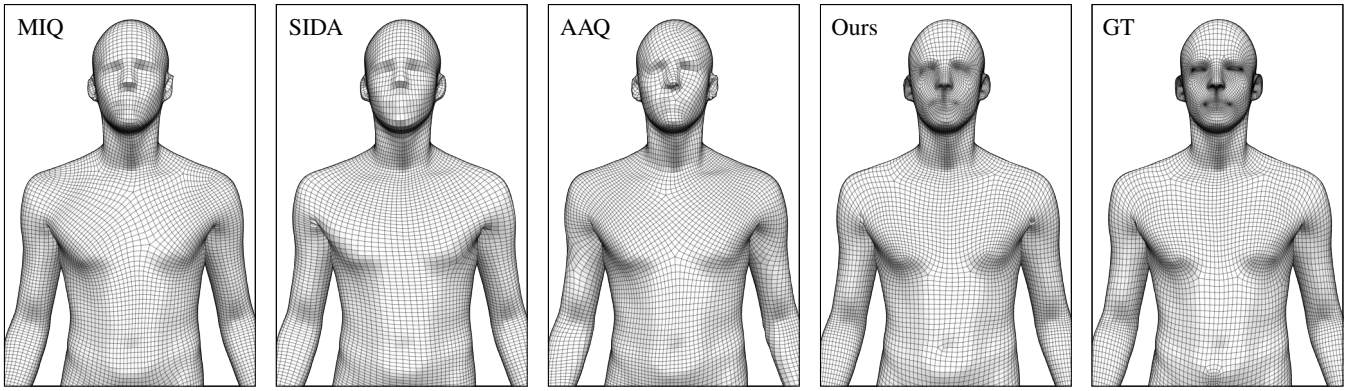

Figure 9: *A comparison between the methods presented by* BOMMES *et al. [BZK09] (MIQ),* CAMPEN *et al. [CIE*16] (SIDA),* MARCIAS *et al. [MPP*13] (AAQ), our approach and the corresponding ground truth mesh.*

to outperform classical frame field generation algorithms on such mechanical parts, since it was trained on human shapes only.

To assess our method in the context of other fully-automatic quadrangulation techniques, we compare the results generated using our approach to those obtained using the curvature-based field synthesis methods presented by BOMMES et al. [BZK09], CAMPEN et al. [CIE*16] and MARCIAS et al. [MPP*13]. For our comparison, we use the implementations and default parameters which were kindly provided by the respective authors. As can be seen in Figure 9, our method generates singular vertices that are substantially better placed w.r.t. the global structure present in the ground truth data. This leads to quadrangulations that are both cleaner and more symmetrical. The perhaps most striking difference in overall fidelity can be observed in the faces of the depicted subjects, where our method manages to generate edge flows that are significantly more similar to those found in manually designed meshes.

A quantitative comparison between the approach presented by CAMPEN et al. [CIE*16] and our method can be found in Figure 10 (left), where we visualize the distribution of angle errors incurred by the frame fields generated using both techniques. As

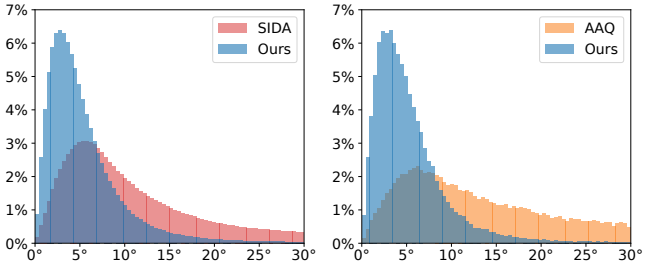

Figure 10: *Left: A comparison between the angle errors incurred by the method presented by* CAMPEN *et al. [CIE*16] (SIDA) and our method. Right: The same comparison for the method presented by* MARCIAS *et al. [MPP*13] (AAQ). As can be seen, our method generates frame fields that are significantly better aligned w.r.t. the corresponding ground truth fields.*

can be seen, our method generates frame fields that are significantly better aligned with the corresponding ground truth fields. Specifically, more than 50 % of all frames generated using our method incur an angle error below 4.35° and more than 90 % of all frames

Method	Angle	Relative Size	Absolute Size
Complete Network	**5.727** ± **5.295**	0.125 ± 0.085	0.00106 ± 0.00069
No Global Network	6.251 ± 5.749	**0.120** ± **0.087**	**0.00101** ± **0.00068**
No Local Network	8.106 ± 7.642	0.158 ± 0.138	0.00125 ± 0.00080
Global + HKS	7.990 ± 7.557	0.155 ± 0.134	0.00123 ± 0.00080
VM + Relative	**5.727** ± **5.295**	0.125 ± 0.085	0.00106 ± 0.00069
VM + Absolute	5.763 ± 5.319	0.127 ± 0.091	0.00106 ± 0.00069
CC + Relative	5.943 ± 5.437	**0.123** ± **0.085**	**0.00104** ± **0.00068**
CC + Absolute	5.897 ± 5.458	0.127 ± 0.091	0.00107 ± 0.00069

Table 1: *Mean errors and corresponding standard deviations on the DFAUST test set for different network variants (top) and loss functions (bottom). Angles are given in degrees. VM and CC denote the von Mises and Complex Cosine loss.*

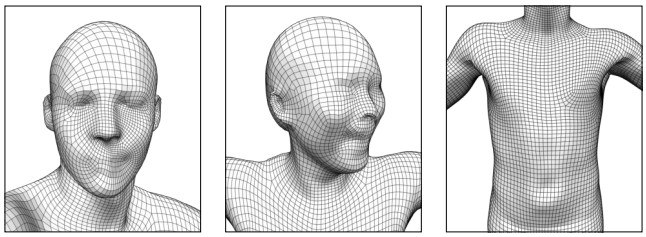

Figure 11: *Our network sometimes fails to account for head rotations, which leads to meshes that capture fewer facial features (left/center). In addition, the singularities around the navel are almost never inferred correctly (right).*

incur an angle error below 10.97°. In comparison, the method presented by CAMPEN et al. only manages to generate 17.07 % and 56.71 % of all frames within the same error bounds. A comparison with the method presented by BOMMES et al. led to similar results.

For the animation-aware method presented by MARCIAS et al., we used the `jumping_jacks` sequence of the test subject with the ID `50009` to compute a direction field for the first frame in the sequence. The resulting quad mesh and error distributions are shown in Figure 9 (center) and Figure 10 (right) respectively. As can be seen, the results produced by our method are closer to the ground truth.

5.1. Ablation Study

To verify the validity of our network architecture, we perform an ablation study in which we remove different network components and retrain the resulting networks. Specifically, we consider the following network variants: a complete network that is identical to the one presented in Section 3.2, two partial networks that do not contain the local and global feature networks respectively and a network that uses Heat Kernel Signatures (HKS) [SOG09] as a drop-in replacement for the features computed by our local network. To evaluate the performance of each of these networks, we use the mean angle, mean relative size and mean absolute size errors of the inferred frame fields on the DFAUST test set. As can be seen in Table 1 (top), the complete network produces the lowest overall angle error and outperforms the two partial networks by 9.1 % and 41.6 %. The network based on the HKS descriptor produces angles that are approximately 40 % less accurate than those predicted by our complete network. Interestingly, the network that does not rely on global shape information achieves a mean size error that is approximately 4 % lower than the error incurred by the complete network. This 4 % advantage, however, is more than offset by the angles inferred by this network, which are approximately 9 % less accurate. As a consequence, we consider the complete network that takes both global and local information into account to be overall superior.

As can be seen in Table 1 (bottom), the von Mises loss performs slightly better than the Complex Cosine loss. Similarly, the relative size loss marginally outperforms its absolute counterpart. As a consequence, the results presented in the previous sections are all based on a network trained using the von Mises direction loss and the relative size loss, even though we consider all of the aforementioned losses to be viable options.

5.2. Failure Cases

For some of the more extreme poses in the FAUST and DFAUST datasets, we sometimes observe frame fields that fail to account for the rotation of the head. As a result, the edges of the generated quad meshes appear to be aligned with one of the global coordinate axes. Two particularly bad examples, where large parts of the generated faces are affected, are shown in Figure 11 (left, center). However, the same problem can, to a lesser extend, also be seen in the two tilted heads shown in Figure 6 (top right). Fortunately, this issue does not occur too often and, if it does occur, is largely contained to the head of the generated mesh. A possible way to mitigate this problem may be to make our network invariant to such rotations.

Another issue arises in connection with the topology corresponding to the navel, where our network appears to be unable to infer the correct number of singularities. An example is shown in Figure 11 (right). This problem is particularly surprising given the fact that these singularities are somewhat isolated and should also be comparatively easy to locate since they are centered around an umbilical point (the navel). However, a closer inspection of the meshes in our training set reveals that the singular vertices in question are often particularly bad aligned with the underlying geometry (as in the ground truth mesh in Figure 9), which may explain why our network is unable to correctly infer these features.

6. Conclusion

We presented a novel approach for the generation of quad meshes that is based on the idea of combining a neural network with a field-guided quadrangulation technique. On the one hand, a purely data-driven approach to the direct generation of quad meshes is difficult to achieve due to topological consistency constraints that need to be upheld. On the other hand, automatic model-driven methods struggle to compute quadrangulations that exhibit similar characteristics to manually created quad meshes without proper guidance. By following a hybrid approach we are able to benefit from both worlds. Our network is able to infer frame fields that resemble the alignment of quads found in our dataset. This rich guidance information can then in turn be used in robust and mature model-driven methods

that are able to guarantee the generation of correct and high-quality quad meshes. Specifically, we demonstrated that our approach is able to infer many topological features that cannot be easily generated using other techniques. The reason for this is that it is not clear how the characteristics of manually-created quad meshes can be derived based on a mathematical model from local surface properties. Our method overcomes this problem by automatically learning a map from local and global surface features to the desired orientation and sizing fields. While our experiments are focused on meshes representing human characters, given suitable training data, we expect that our method can be adapted to other input types. As a consequence, we believe that our work represents a first step towards the goal of automatically generating meshes that more closely resemble their manually-created counterparts.

Acknowledgements

The authors thank Jan Möbius for creating and maintaining the geometry processing framework OpenFlipper [MK12]. This work was partially supported by the Gottfried-Wilhelm-Leibniz Programme of the Deutsche Forschungsgemeinschaft (DFG) and by the Deutsche Forschungsgemeinschaft (DFG) - 392037563.

References

[AML18] ATZMON, MATAN, MARON, HAGGAI, and LIPMAN, YARON. "Point convolutional neural networks by extension operators". *ACM Transactions on Graphics* 37.4 (2018), 71:1–71:12 2.

[BBL*17] BRONSTEIN, MICHAEL M., BRUNA, JOAN, LECUN, YANN, et al. "Geometric Deep Learning: Going beyond Euclidean data". *IEEE Signal Process. Mag.* 34.4 (2017), 18–42 2.

[BCE*13] BOMMES, DAVID, CAMPEN, MARCEL, EBKE, HANS-CHRISTIAN, et al. "Integer-Grid Maps for Reliable Quad Meshing". *ACM Transactions on Graphics* 32.4 (2013), 1–12 1, 2.

[BHL15] BEYER, LUCAS, HERMANS, ALEXANDER, and LEIBE, BASTIAN. "Biternion Nets: Continuous Head Pose Regression from Discrete Training Labels". *German Conference on Pattern Recognition*. Springer. 2015, 157–168 5.

[BK04] BOTSCH, MARIO and KOBBELT, LEIF. "A Remeshing Approach to Multiresolution Modeling". *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. 2004, 185–192 6.

[BLP*13] BOMMES, DAVID, LÉVY, BRUNO, PIETRONI, NICO, et al. "Quad-Mesh Generation and Processing: A Survey". *Computer Graphics Forum* 32.6 (2013), 51–76 2.

[BRLB14] BOGO, FEDERICA, ROMERO, JAVIER, LOPER, MATTHEW, and BLACK, MICHAEL J. "FAUST: Dataset and Evaluation for 3D Mesh Registration". *IEEE Conf. on Computer Vision and Pattern Recognition*. 2014, 3794–3801 6, 7.

[BRPB17] BOGO, FEDERICA, ROMERO, JAVIER, PONS-MOLL, GERARD, and BLACK, MICHAEL J. "Dynamic FAUST: Registering Human Bodies in Motion". *IEEE Conf. on Computer Vision and Pattern Recognition*. 2017, 6233–6242 5.

[BZK09] BOMMES, DAVID, ZIMMER, HENRIK, and KOBBELT, LEIF. "Mixed-Integer Quadrangulation". *ACM Transactions on Graphics* 28.3 (2009), 1–10 1–3, 8, 9.

[CBK15] CAMPEN, MARCEL, BOMMES, DAVID, and KOBBELT, LEIF. "Quantized Global Parametrization". *ACM Transactions on Graphics* 34.6 (2015), 1–12 1, 2, 4, 6.

[CC78] CATMULL, EDWIN and CLARK, JAMES. "Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes". *Computer-Aided Design* 10.6 (1978), 350–355 3, 5, 6.

[CIE*16] CAMPEN, MARCEL, IBING, MORITZ, EBKE, HANS-CHRISTIAN, et al. "Scale-Invariant Directional Alignment of Surface Parametrizations". *Computer Graphics Forum*. Vol. 35. 5. Wiley Online Library. 2016, 1–10 2, 8, 9.

[CK14] CAMPEN, MARCEL and KOBBELT, LEIF. "Dual Strip Weaving: Interactive Design of Quad Layouts using Elastica Strips". *ACM Transactions on Graphics* 33.6 (2014), 183:1–183:10 2.

[DVPS15] DIAMANTI, OLGA, VAXMAN, AMIR, PANOZZO, DANIELE, and SORKINE-HORNUNG, OLGA. "Integrable PolyVector Fields". *ACM Transactions on Graphics* 34.4 (2015), 1–12 2, 3, 5.

[ESCK16] EBKE, HANS-CHRISTIAN, SCHMIDT, PATRICK, CAMPEN, MARCEL, and KOBBELT, LEIF. "Interactively Controlled Quad Remeshing of High Resolution 3D Models". *ACM Transactions on Graphics* 35.6 (Nov. 2016), 218:1–218:13 1, 2.

[FBT*18] FANG, XIANZHONG, BAO, HUJUN, TONG, YIYING, et al. "Quadrangulation through Morse-Parameterization Hybridization". *ACM Transactions on Graphics* 37.4 (July 2018) 2.

[FLWM18] FEY, MATTHIAS, LENSSEN, JAN ERIC, WEICHERT, FRANK, and MÜLLER, HEINRICH. "SplineCNN: Fast Geometric Deep Learning With Continuous B-Spline Kernels". *IEEE Conf. on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2018, 869–877 2.

[GEvdM18] GRAHAM, BENJAMIN, ENGELCKE, MARTIN, and van der MAATEN, LAURENS. "3D Semantic Segmentation With Submanifold Sparse Convolutional Networks". *IEEE Conf. on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2018, 9224–9232 2.

[GSST20] GIRARD, N., SMIRNOV, D., SOLOMON, J., and TARABALKA, Y. "Regularized Building Segmentation by Frame Field Learning". *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*. 2020, 1805–1808 2.

[HHF*19] HANOCKA, RANA, HERTZ, AMIR, FISH, NOA, et al. "MeshCNN: a network with an edge". *ACM Transactions on Graphics* 38.4 (2019), 90:1–90:12 2.

[HS97] HOCHREITER, SEPP and SCHMIDHUBER, JÜRGEN. "Long Short-Term Memory". *Neural Computation* 9.8 (1997), 1735–1780 4.

[HZN*18] HUANG, JINGWEI, ZHOU, YICHAO, NIESSNER, MATTHIAS, et al. "QuadriFlow: A Scalable and Robust Method for Quadrangulation". *Computer Graphics Forum* (2018). ISSN: 1467-8659 2.

[JFH*15] JIANG, TENGFEI, FANG, XIANZHONG, HUANG, JIN, et al. "Frame Field Generation through Metric Customization". *ACM Transactions on Graphics* 34.4 (2015), 1–11 2, 3.

[JTPS15] JAKOB, WENZEL, TARINI, MARCO, PANOZZO, DANIELE, and SORKINE-HORNUNG, OLGA. "Instant field-aligned meshes". *ACM Transactions on Graphics* 34.6 (2015), 189 1, 2.

[KB15] KINGMA, DIEDERIK P. and BA, JIMMY. "Adam: A Method for Stochastic Optimization". *International Conference on Learning Representations*. Ed. by BENGIO, YOSHUA and LECUN, YANN. 2015 6.

[KCPS13] KNÖPPEL, FELIX, CRANE, KEENAN, PINKALL, ULRICH, and SCHRÖDER, PETER. "Globally Optimal Direction Fields". *ACM Transactions on Graphics* 32.4 (2013), 1–10 2, 3, 5.

[KCS98] KOBBELT, LEIF, CAMPAGNA, SWEN, and SEIDEL, HANS-PETER. "A General Framework for Mesh Decimation". *Graphics Interface*. Vol. 98. 1998, 43–50 6.

[KNP07] KÄLBERER, FELIX, NIESER, MATTHIAS, and POLTHIER, KONRAD. "QuadCover – Surface Parameterization using Branched Coverings". *Computer Graphics Forum* 26.3 (2007) 2.

[LCBK19] LYON, MAX, CAMPEN, MARCEL, BOMMES, DAVID, and KOBBELT, LEIF. "Parametrization Quantization with Free Boundaries for Trimmed Quad Meshing". *ACM Transactions on Graphics* 38.4 (2019), 1–14 2.

[LDCK18] LIM, ISAAK, DIELEN, ALEXANDER, CAMPEN, MARCEL, and KOBBELT, LEIF. "A Simple Approach to Intrinsic Correspondence Learning on Unstructured 3D Meshes". *European Conference on Computer Vision*. 2018 2, 4.

[LKC*20] LIU, HSUEH-TI DEREK, KIM, VLADIMIR G., CHAUDHURI, SIDDHARTHA, et al. "Neural subdivision". *ACM Transactions on Graphics* 39.4 (2020), 124 2.

[LMR*15] LOPER, MATTHEW, MAHMOOD, NAUREEN, ROMERO, JAVIER, et al. "SMPL: A Skinned Multi-Person Linear Model". *ACM Transactions on Graphics* 34.6 (2015), 1–16 3, 5.

[MBBV15] MASCI, JONATHAN, BOSCAINI, DAVIDE, BRONSTEIN, MICHAEL M., and VANDERGHEYNST, PIERRE. "Geodesic Convolutional Neural Networks on Riemannian Manifolds". *IEEE International Conference on Computer Vision*. IEEE Computer Society, 2015, 832–840 2.

[MBM*17] MONTI, FEDERICO, BOSCAINI, DAVIDE, MASCI, JONATHAN, et al. "Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs". *IEEE Conf. on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2017, 5425–5434 2.

[MGA*17] MARON, HAGGAI, GALUN, MEIRAV, AIGERMAN, NOAM, et al. "Convolutional neural networks on surfaces via seamless toric covers". *ACM Transactions on Graphics* 36.4 (2017), 71:1–71:10 2.

[MJ09] MARDIA, KANTI V and JUPP, PETER E. *Directional Statistics*. Vol. 494. John Wiley & Sons, 2009 5.

[MK12] MÖBIUS, JAN and KOBBELT, LEIF. "OpenFlipper: An Open Source Geometry Processing and Rendering Framework". *Curves and Surfaces*. Vol. 6920. Lecture Notes in Computer Science. 2012 10.

[MPP*13] MARCIAS, GIORGIO, PIETRONI, NICO, PANOZZO, DANIELE, et al. "Animation-Aware Quadrangulation". *Computer Graphics Forum*. Vol. 32. 5. Wiley Online Library. 2013, 167–175 8, 9.

[MS15] MATURANA, DANIEL and SCHERER, SEBASTIAN A. "VoxNet: A 3D Convolutional Neural Network for real-time object recognition". *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*. IEEE, 2015, 922–928 2.

[MTP*15] MARCIAS, GIORGIO, TAKAYAMA, KENSHI, PIETRONI, NICO, et al. "Data-Driven Interactive Quadrangulation". *ACM Transactions on Graphics* 34.4 (2015), 1–10 2.

[NH10] NAIR, VINOD and HINTON, GEOFFREY E. "Rectified Linear Units Improve Restricted Boltzmann Machines". *International Conference on International Conference on Machine Learning*. 2010 4.

[NSY09] NASRI, A, SABIN, MALCOM, and YASSEEN, ZAHRAA. "Filling N-Sided Regions by Quad Meshes for Subdivision Surfaces". *Computer Graphics Forum*. Vol. 28. 6. Wiley Online Library. 2009, 1644–1658 2.

[PBJW14] PENG, CHI-HAN, BARTON, MICHAEL, JIANG, CAIGUI, and WONKA, PETER. "Exploring quadrangulations". *ACM Transactions on Graphics* 33.1 (2014), 1–13 2.

[PFSB20] PFAFF, TOBIAS, FORTUNATO, MEIRE, SANCHEZ-GONZALEZ, ALVARO, and BATTAGLIA, PETER W. "Learning Mesh-Based Simulation with Graph Networks". *CoRR* abs/2010.03409 (2020). arXiv: 2010.03409 2.

[PPTS14] PANOZZO, DANIELE, PUPPO, ENRICO, TARINI, MARCO, and SORKINE-HORNUNG, OLGA. "Frame Fields: Anisotropic and Non-orthogonal Cross Fields". *ACM Transactions on Graphics* 33.4 (2014), 1–11 2, 3.

[QSMG17] QI, CHARLES R, SU, HAO, MO, KAICHUN, and GUIBAS, LEONIDAS J. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". *IEEE Conf. on Computer Vision and Pattern Recognition*. 2017, 652–660 2, 4.

[QYSG17] QI, CHARLES RUIZHONGTAI, YI, LI, SU, HAO, and GUIBAS, LEONIDAS J. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". *Advances in Neural Information Processing Systems*. Ed. by GUYON, ISABELLE, von LUXBURG, ULRIKE, BENGIO, SAMY, et al. 2017, 5099–5108 2.

[SBR16] SINHA, AYAN, BAI, JING, and RAMANI, KARTHIK. "Deep Learning 3D Shape Surfaces Using Geometry Images". *European Conference on Computer Vision*. Ed. by LEIBE, BASTIAN, MATAS, JIRI, SEBE, NICU, and WELLING, MAX. Vol. 9910. Lecture Notes in Computer Science. Springer, 2016, 223–240 2.

[SMKL15] SU, HANG, MAJI, SUBHRANSU, KALOGERAKIS, EVANGELOS, and LEARNED-MILLER, ERIK G. "Multi-view Convolutional Neural Networks for 3D Shape Recognition". *IEEE International Conference on Computer Vision*. IEEE Computer Society, 2015, 945–953 2.

[SOG09] SUN, JIAN, OVSJANIKOV, MAKS, and GUIBAS, LEONIDAS. "A Concise and Provably Informative Multi-Scale Signature based on Heat Diffusion". *Computer Graphics Forum*. Vol. 28. 5. Wiley Online Library. 2009, 1383–1392 9.

[TPS14] TAKAYAMA, KENSHI, PANOZZO, DANIELE, and SORKINE-HORNUNG, OLGA. "Pattern-Based Quadrangulation for N-Sided Patches". *Computer Graphics Forum*. Vol. 33. 5. Wiley Online Library. 2014, 177–184 2.

[TPSS13] TAKAYAMA, KENSHI, PANOZZO, DANIELE, SORKINE-HORNUNG, ALEXANDER, and SORKINE-HORNUNG, OLGA. "Sketch-Based Generation and Editing of Quad Meshes". *ACM Transactions on Graphics* 32.4 (2013), 1–8 1, 2.

[TQD*19] THOMAS, HUGUES, QI, CHARLES R., DESCHAUD, JEAN-EMMANUEL, et al. "KPConv: Flexible and Deformable Convolution for Point Clouds". *IEEE International Conference on Computer Vision*. IEEE, 2019, 6410–6419 2.

[VCD*16] VAXMAN, AMIR, CAMPEN, MARCEL, DIAMANTI, OLGA, et al. "Directional Field Synthesis, Design, and Processing". *Computer Graphics Forum*. Vol. 35. 2. Wiley Online Library. 2016, 545–572 4.

[WSL*19] WANG, YUE, SUN, YONGBIN, LIU, ZIWEI, et al. "Dynamic Graph CNN for Learning on Point Clouds". *ACM Transactions on Graphics* 38.5 (2019), 146:1–146:12 2.

[WSLT18] WANG, PENG-SHUAI, SUN, CHUN-YU, LIU, YANG, and TONG, XIN. "Adaptive O-CNN: a patch-based deep representation of 3D shapes". *ACM Transactions on Graphics* 37.6 (2018), 217:1–217:11 2.

[WSS18] WANG, CHU, SAMARI, BABAK, and SIDDIQI, KALEEM. "Local Spectral Graph Convolution for Point Set Feature Learning". *European Conference on Computer Vision*. Ed. by FERRARI, VITTORIO, HEBERT, MARTIAL, SMINCHISESCU, CRISTIAN, and WEISS, YAIR. Vol. 11208. Lecture Notes in Computer Science. Springer, 2018, 56–71 2.

[XLZ*20] XIAO, YUN-PENG, LAI, YU-KUN, ZHANG, FANG-LUE, et al. "A survey on deep geometry learning: From a representation perspective". *Comput. Vis. Media* 6.2 (2020), 113–133 2.

[ZKR*17] ZAHEER, MANZIL, KOTTUR, SATWIK, RAVANBAKHSH, SIAMAK, et al. "Deep Sets". *Advances in Neural Information Processing Systems*. Ed. by GUYON, ISABELLE, von LUXBURG, ULRIKE, BENGIO, SAMY, et al. 2017, 3391–3401 2.