Face Extrusion Quad Meshes: Supplementary Material

KARRAN PANDEY, University of Toronto, Canada
J. ANDREAS BÆRENTZEN, Technical University of Denmark, Denmark
KARAN SINGH, University of Toronto, Canada

In this document we include the supplementary material for our paper "Face Extrusion Quad Meshes". Specifically, we describe the extrusion-based remeshing procedure for generating FEQs in greater detail. We have aimed to make this discussion self-contained. This document also contains additional tests of the method and comparisons with other remeshing methods.

1 EXTRUSION-BASED REMESHING

Our extrusion-based remeshing takes as input a triangle mesh and a curve skeleton represented as a graph. The skeletal graph is converted to a mesh with FEQ connectivity by creating a polyhedron for each branch node in the skeleton and, subsequently, using bridging and extrusion operations to form a pure quad mesh. The resulting mesh is finally fitted to the input quad mesh.

The code for extrusion-based remeshing is made available in the GEL open source library available at https://github.com/janba/GEL.

1.1 Branch Node Meshes

The branch node mesh for a junction in the skeletal graph is constructed using three steps.

First, vertices corresponding to each outgoing path, called *path vertices*, are identified by intersecting outgoing edges with a sphere centered on the junction node. A spherical Delaunay triangulation of the identified path vertices is then computed, resulting in a branch node polyhedron (BNP) [Bærentzen et al. 2012]. Thus, by construction, each vertex in a BNP corresponds to an outgoing path of the junction.

Next, the BNP connectivity is improved using two heuristics which are discussed further in subsection 1.3.

Finally, the BNP is subdivided using Catmull-Clark subdivision to build the branch node quad mesh. The faces incident on a path vertex are denoted its *branch face set*. The number of quads in the branch face set are, of course, the same as the valence of the corresponding path vertex. Please see Figure 2 for examples of branch face sets around path vertices.

1.2 Graph Path Reconstruction

In order to reconstruct the graph paths, we extrude the branch face sets if the associated path leads to a leaf node. For instance, in Figure 1 the legs, tail, and neck of the quadruped are meshed by extruding branch face sets along their respective paths.

If the skeletal path connects two branch nodes, we instead bridge the corresponding branch face sets. If the two branch face sets have k_1 and k_2 faces where $k_1 = k_2$, we simply bridge the complete face sets since they are identical. When the branch face sets form a complete ring around the path vertex, we will denote this configuration a *closed ring*.

Authors' addresses: Karran Pandey, karran@cs.toronto.edu, University of Toronto, Toronto, Canada; J. Andreas Bærentzen, janba@dtu.dk, Technical University of Denmark, Copenhagen, Denmark; Karan Singh, karan@dgp.toronto.edu, University of Toronto, , Toronto, Canada.

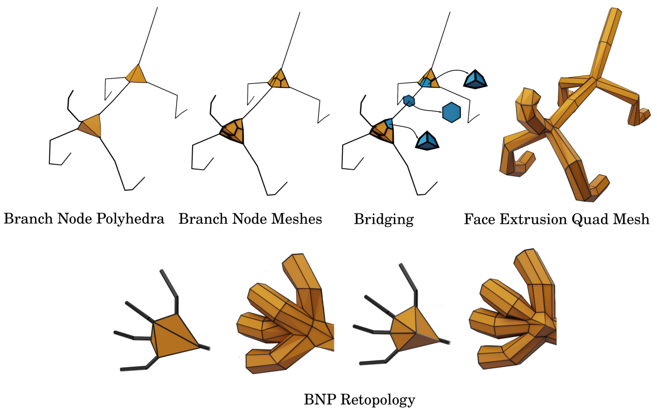

Fig. 1. Top: the extrusion based remeshing pipeline consists of the steps shown from left to right. First the BNP is created by spherical Delaunay triangulation followed by a retopology step. Next, the BNP are subdivided forming the branch node meshes, and finally bridging and extrusion lead to the final FEQ. Below: an example of how the vertex insertion alters the connectivity of a hand mesh.

However, if $k_1 \neq k_2$, we select $\min(k_1, k_2) - 1$ faces from each branch face set (i.e. an *open ring*) as illustrated in Figure 2. Clearly, we could have chosen to use the closed ring of the smaller face set, but a closed ring of quads has a different mesh structure from an open ring as illustrated in Figure 2. On the other hand, if we choose open rings with equal numbers of quads from either face set, we ensure that the two face sets have identical structure.

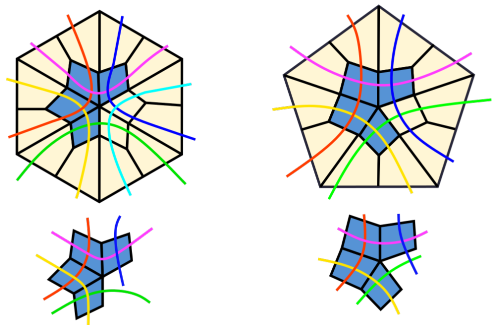

Fig. 2. The figure illustrates face loops in 5 and 6-sized branch face sets. We can see how selecting 4 (i.e $k_1 - 1$ faces) results in the same number of corresponding face loops intersecting the selected face set. In a closed ring, all face loops are shared by two faces whereas in an open ring, two face loops only traverse a single quad.

Having identified the connectivity of the input face sets for the bridge operation, we now construct the bridge geometry along the graph path connecting the two branch nodes.

If there exist valence 2 nodes in the path, we create an n-sided polygonal face pair centered at these nodes, where n is the size of the outer boundary of the branch face set. As each mesh element now has the same outer boundary size, bridging the mesh elements corresponding to adjacent nodes in the graph path returns a quad bridge with face loops going around the path geometry.

To construct the bridge between adjacent nodes, we identify an optimal mapping between vertices of the corresponding face set boundaries. The mapping aims to minimize torsion and ensure that the bridge edges are aligned with the underlying graph path. A bridge constructed in this manner will have the underlying graph path as their medial axis and face loops which go around graph edges. Bridging is illustrated in Figure 1 (top row).

1.3 BNP Connectivity Improvement

Planar Junctions. Planar regions in a BNP can be identified as a collection of faces with a small dihedral angle between them: we use the specific threshold of 15°. Such regions correspond to a set of outgoing graph paths which approximately lie within the same plane, like, say, the fingers of a hand. In terms of this example, we would expect that a given finger is only connected to the adjacent fingers, but since the BNP is a triangulation, we also get edges connecting non-adjacent fingers as illustrated in Figure 3.

To prevent this, we *retopologize* planar regions in a BNP by introducing an auxiliary vertex in the center of the planar region and connecting this new vertex to all of the boundary vertices. This can be seen as a relatively simple refinement step, similar to normal Delaunay refinement [Shewchuk 2002]. Returning to the hand example, this means that the fingers are now only connected to adjacent fingers and the auxiliary vertex which roughly corresponds to the palm.

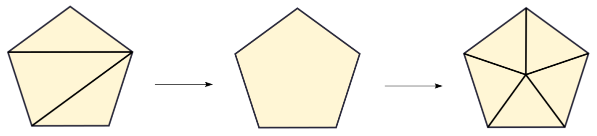
Fig. 3. The figure illustrates the retopology step for planar regions in a BNP. We see how undesirable long diagonal edges can form in triangulating the region. We then remove all the interior edges and triangulate the region by introducing a central vertex and connecting it to all boundary vertices. The triangles are visibly more uniform following this process.

Surplus Quad Patches. When open rings are matched, left over faces appear as surplus quad patches in junctions of the final FEQ. While such patches are sometimes a necessity for the quad layout (for instance the pink wedge shaped patch in the hand shown in the top row of Figure 7) they can often be removed, leading to a simpler mesh.

To reduce the number of unmatched faces in the bridging step, the BNP is refined before Catmull-Clark subdivision. We know the number of faces in the branch face set will be equal to the valence of the BNP path vertex. Thus, we can increase the number of faces in the smaller branch face set by splitting one of the BNP edges in the link of the path vertex and introducing two new edges as illustrated in Figure 4. It is clear that we thereby increase the valence of both the path vertex whose valence we intend to increase and the path vertex that happens to share the link edge.

Splitting is only performed if it leads to a match between closed rings (i.e. $k_1 == k_2$), and it is applied when one of two further conditions are met.

- both path vertices opposite the edge considered for splitting come closer to a closed ring match, or
- one of the path vertices comes closer to a closed ring match whereas the other vertex is associated with a leaf path.

The procedure is applied greedily by performing the splitting to the configurations that benefit the most until no more improvements can be made.

Importantly, we do not make changes that would cause a path vertex to move away from a closed ring match, since that would propagate the need for edge splitting. Thus, unlike [Bærentzen et al. 2012; Suárez and Hubert 2018; Usai et al. 2015], we avoid searching for a global solution, e.g. by solving an integer linear programming problem.

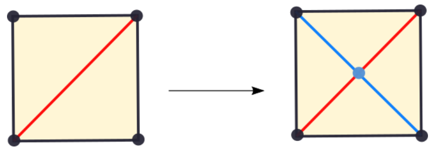
Fig. 4. The figure illustrates the edge splitting step. The edge in red is first split, to introduce the blue vertex, and then blue edges are introduced connecting it to its opposite vertices

1.4 Fitting

Having constructed the inverse skeletonized quad mesh M with the desired face loop structure, we inflate and fit M to the reference shape S. Our fitting procedure uses the sparse-linear least-squares system outlined in Least-Squares Meshes [2004]. In particular, the system solves for the fitted vertex locations x by minimizing an energy, $||A\boldsymbol{x} - \boldsymbol{b}||$, consisting of a weighted euclidean distance term for a selection of control vertices C and a uniform laplacian regularizer L:

$$||A\boldsymbol{x} - b||^2 = ||L\boldsymbol{x}||^2 + \sum_{s \in C} w_s |x_s - v_s|^2 \tag{1}$$

where v_s is the control point location corresponding to an input vertex x_s. This formulation allows for an extensive flexibility in the selection of control points, admitting both sparse or over-determined correspondences between the vertices of M and S. Given a correspondence, the least-squares system reconstructs a smooth mesh with the connectivity of M and geometry constrained to adhere as far as possible to the specified control point correspondence.

In order to fit M to S, we iteratively vary correspondence schemes to perform three types of geometric updates to M. M is first inflated by specifying correspondences along vertex normals of M. We then perform a direct distance-based fit by finding the closest-point on S for each vertex in M. Finally, we perform an over-determined inverse distance-based fit, by identifying the closest-point on M for each vertex in S. Each correspondence is accompanied by a weight w_s which measures the alignment of vertex normals between corresponding vertices. The weights help us ensure that each geometric update retains the quality of an inflation throughout the process. The weight, $w_s(x_s, v_s) = ||N(x_s) \cdot N(v_s)||$, is defined as the dot product of the normals of the corresponding vertices.

2 COMPARISONS

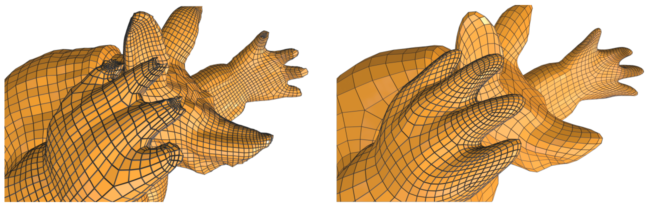
Fig. 5. The figure illustrates differences in the fitting procedure of Usai et al. (left) and our method (right).

We have compared our extrusion based remeshing approach to the related method of Usai et al. [2015] and the recent, feature line-driven method of Pietroni et al. [2021].

The method produced by Usai et al. produces results similar to ours as shown in Figure 7. However, there are notable differences. In the fertility model, Usai et al. produce an additional junction at the foot of the model, whereas the face loops of the FEQ simply curve around the bend. The waist of the warrior model highlights the potential pitfalls of assuming a cube-like symmetry at junctions. Usai et al. struggle to align with the symmetry of the Y-shaped junction with a cube, whereas our method is free of such assumptions and therefore cleanly reconstructs the Y-shaped junction of the warriors legs. Usai et al. do allow for manual adjustments which were used to fix the Y-junction in a post processing step [Usai et al. 2015]. The fitting procedure of Usai et al. seems to lead to artefacts at extremeties whereas our Laplacian deformation scheme better preserves the mesh structure in these regions as shown in Figure 5.

Comparing to Pietroni et al. in Figure 6, we mainly observe that it is often possible for a face loop to self-intersect multiple times and cover large regions of the model in case of Pietroni et al., thus losing most, if not all, of the structural information. In contrast, FEQs have a coherent set of non-self intersecting face loops which encode the skeletal structure of the shape.

Finally, we tested our fitting using different amounts of subdivision and with features present in the skeleton or removed from the skeleton. This is seen in Figure 8.

REFERENCES

J. Andreas Bærentzen, Marek Krzysztof Misztal, and K Wełnicka. 2012. Converting skeletal structures to quad dominant meshes. *Computers & Graphics* 36, 5 (2012), 555–561.

Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, and Marco Tarini. 2021. Reliable Feature-Line Driven Quad-Remeshing. *ACM Transactions on Graphics* 40, 4, Article 155 (jul 2021), 17 pages. https://doi.org/10.1145/3450626.3459941

Jonathan Richard Shewchuk. 2002. Delaunay refinement algorithms for triangular mesh generation. *Computational geometry* 22, 1-3 (2002), 21–74.

Olga Sorkine and Daniel Cohen-Or. 2004. Least-squares meshes. In *Proceedings Shape Modeling Applications 2004*. IEEE, 191–199.

AJ Fuentes Suárez and Evelyne Hubert. 2018. Scaffolding skeletons using spherical Voronoi diagrams: Feasibility, regularity and symmetry. *Computer-Aided Design* 102 (2018), 83–93.

Francesco Usai, Marco Livesu, Enrico Puppo, Marco Tarini, and Riccardo Scateni. 2015. Extraction of the Quad Layout of a Triangle Mesh Guided by Its Curve Skeleton. *ACM Transactions on Graphics* 35, 1, Article 6 (dec 2015), 13 pages. https://doi.org/10.1145/2809785

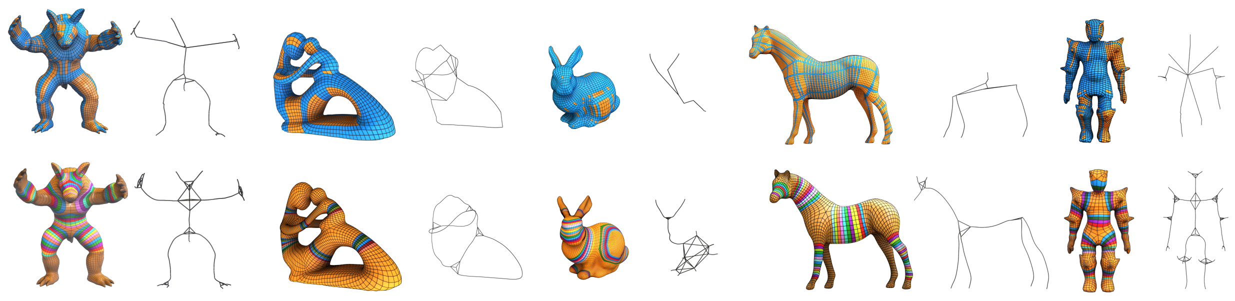
Fig. 6. Top row: in blue we see a single, particularly long face loop highlighted on meshes from Pietroni et al. Below: the FEQ face loops in regions covered by the blue face loop above are shown color coded.

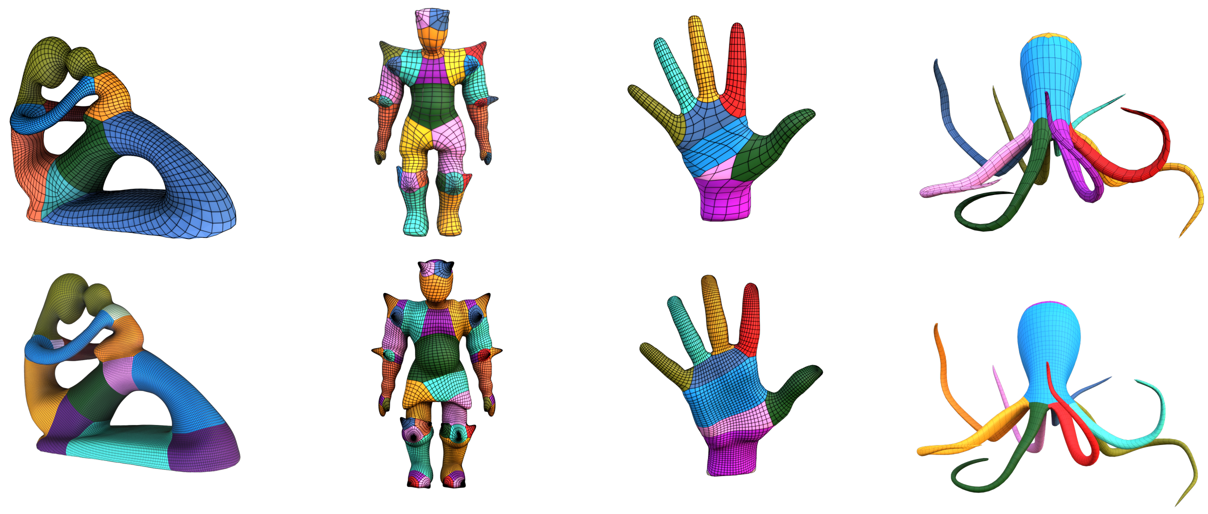
Fig. 7. FEQs produced using our method (top), and the quad meshes of Usai et al (below).

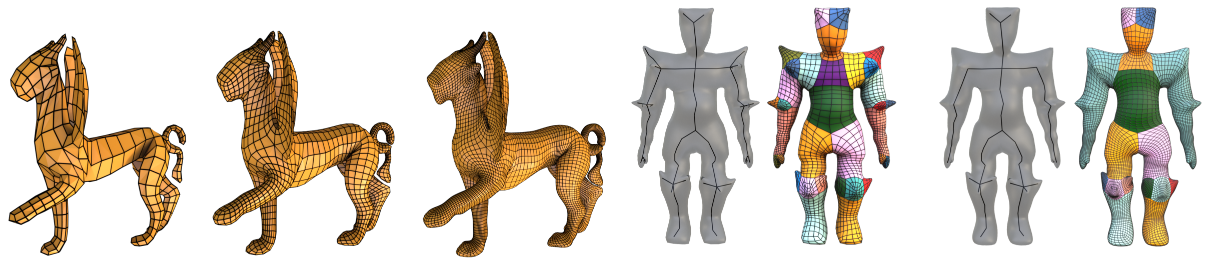
Fig. 8. Left: effect of subdivision iterations during the fitting procedure. Right: the effect of using different skeletons on the fitting. In particular, we see that the spikes of the warrior are captured by the fitting even if not articulated in the skeleton