Supplementary Material: Neural Mesh Simplification

Rolandos Alexandros Potamias[1] Stylianos Ploumpis[1,2] Stefanos Zafeiriou[1,2]
[1]Imperial College London [2]Huawei Technologies Co. Ltd
{r.potamias, s.ploumpis, s.zafeiriou}@imperial.ac.uk

1. Implementation Details

1.1. Model

Code will be made public after reviewing process.
Point Sampler: We construct the Point Sampler using three stacked DevConv layers, followed with a ReLU activation. We select 64 as a hidden dimension, since we aim for a lightweight and fast model. We experimented with deeper architectures and larger latent spaces but we did not observe a significant increase in the performance compared to the run-time trade off.

Edge Predictor: Before passing the nodes to the edge predictor we extend the original adjacency matrix by adding links between the nodes sampled by the Point Sampler. This is done by constructing a k-nn graph for the sampled node set. We use $k = 15$. The extended graph is passed through a DevConv layer with 64 hidden dimensions. We utilized DevConv to give the model the opportunity to assign sparse attention weights. With the use of *Linear* or *vanilla*-gnn layers, points with similar features will always receive high attention scores, which in practice we observed that is not always useful. In particular, since the k-nearest neighbors of point will most often share similar features, their $softmax$ normalization will result in almost uniform attention weights. In contrary, by utilizing DevConv we achieve larger variations between the point features which enables more sparse attention weights.

Face Classifier: We utilize three stacked TriConv layers followed with ReLU activation to encode triangle to the latent space, formed by its k-nearest neighbors. We set k=20, with the triangle neighbors selected by the respective distances to their barycenters. The triangle features are encoded to an 128-dimensional embedding space with additional 9 features from the triangles relative coordinates. The first TriConv layer takes as input the initial triangle probability predicted by the edge predictor. The final TriConv layer is topped with a softmax layer to constrain the triangle probabilities to the $(0, 1)$ interval.

1.2. Loss Functions

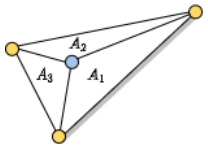
All *Triangle Collusion, Edge Crossings* and *Overlaping Triangles* losses are efficiently calculated using the k=50 nearest triangles of the query triangle based on their barycenters.

Edge Crossings: To calculate the edge crossings between two triangles we initially split each triangle to the three lines that define their edges. Then, we calculate the points of intersection between all nine possible edge-line combinations of two neighboring triangles. Finally, we validate if the points of intersection belong inside the line segments, i.e. edges of the query triangle. For each triangle, we measure the frequency it penetrates the edges of its neighboring triangle and penalize it using the formula bellow:

$$\mathcal{L}_e = \frac{1}{|\mathcal{F}_s|} \sum_{t \in \mathcal{F}_s} p_t m_e(t) \tag{1}$$

where p_t denotes the probability of triangle t, $m_e(t)$ the number of edges that triangle t crosses and $\mathcal{F}_s$ the set of generated triangles.

Overlaping Triangles: To penalize the triangles that overlap in space, we sample 100 points from each triangle and calculate their distances from the 50-nearest triangles. The aim is that each sampled point to belong to only one triangle. To identify if a point belongs to a given triangle, we measure the three areas A_1, A_2, A_3 produced by substituting each triangle's vertex with the query point and crosscheck if the sum of the three produced areas equals the area of the triangle. We also provision for triangles that share parallel planes by slightly increasing the distance tolerance to the axis that is vertical to the triangle's plane. Finally, we apply a penalty to all triangles that share a sampled point. Similar to the Edge Crossing loss, the penalty applied to the triangle is propor-



tional to the number of triangles that overlaps:

$$\mathcal{L}_o = \frac{1}{|\mathcal{F}_s|} \sum_{t \in \mathcal{F}_s} p_t m_o(t) \tag{2}$$

where p_t denotes the probability of triangle t, $m_o(t)$ the number of triangles that triangle t overlaps.

1.3. Evaluation Measures

Curvature Preservation: A significant property of high resolution meshes is the sharp details they are able to represent. A simplification algorithm should also preserve, as much as possible, these details. Mesh details are most frequently quantified in terms of point curvatures. We measure the curvature error introduced in the simplified model as:

$$\mathcal{E}_c = \left(\frac{1}{|\mathcal{P}|} \sum_{x \in \mathcal{P}} \|\bar{\mathcal{K}}(x) - \bar{\mathcal{K}}_s(y)\|^2 \right)^{1/2} \tag{3}$$

where y denotes the nearest neighbour of x in simplified set $\mathcal{P}_s$, and $\bar{\mathcal{K}}(\cdot)$ denotes the mean curvature. For the point cloud experiments presented in Section 4.3 we estimate the point curvature as suggested in [1].

Laplacian Error: To evaluate the preservation of the mesh spectral properties, we calculated the Laplacian error between the original and the simplified mesh, defined as the Mean Squared Error over the first 150 eigenvectors of the Laplacian operator of the two meshes:

$$\mathcal{E}_L = \left(\sum_i \|\hat{\phi}_i - \phi_i\|^2 \right) \tag{4}$$

where $\phi_i, \hat{\phi}_i$ denotes the i-th eigenvector of the Laplacian operator for the original and the simplified mesh, respectively.

Normal Error: To assess the visual appearance of the simplified models we utilize a normal error that measures the cosine similarity between the normals of the two models. Regarding the forward term, for each face of the simplified model we find its nearest face in the original mesh and measure their normal differences. The reverse term estimates the closest face of the simplified mesh for each face of the original mesh and calculates their normal difference. The mathematical formulation of the total normal error is:

$$\begin{aligned} \mathcal{E}_n = & \frac{1}{|\mathcal{P}_1|} \sum_{\substack{x \in \mathcal{P} \\ y \in NN(x, \mathcal{P}_s)}} 1 - \frac{\mathbf{n_x} \cdot \mathbf{n_y}}{\|\mathbf{n_x}\| \|\mathbf{n_y}\|} + \\ & \frac{1}{|\mathcal{P}_s|} \sum_{\substack{y \in \mathcal{P}_s \\ x \in NN(y, \mathcal{P})}} 1 - \frac{\mathbf{n_x} \cdot \mathbf{n_y}}{\|\mathbf{n_x}\| \|\mathbf{n_y}\|} \end{aligned} \tag{5}$$

where $\mathbf{n_x}$ denotes the normal of face x and $NN(x, \mathcal{P}_s)$ the nearest neighbours of face x in simplified mesh $\mathcal{P}_s$.

2. Experiments

2.1. Ablation Study over the loss functions

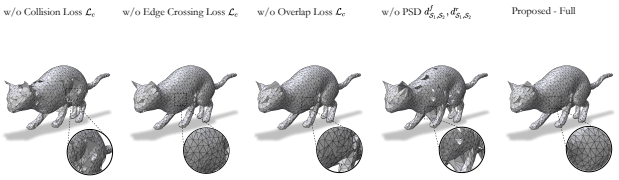


Figure 1. Ablation study: Qualitative comparison indicating the contributions of each loss function. Zoomed areas illustrate areas with irregular triangles.

In this section we assess the importance that each loss function holds by means of an ablation study (see Table 1). In particular, we remove one component of the loss function at a time and measure the Chamfer distance (CD), watertightness percetage (WA), the Laplacian error (LE) and the normals error (NE) at several simplification ratios. It can be easily observed that the models trained without overlap (OV), edge crossing (EC) or triangle collusion (TC) losses, although they achieve similar CD and NE, they fail to preserve the Laplacian of the initial mesh since the irregular triangles perturb the geodesics of the mesh. In addition, as illustrated in Figure 1, model trained without Probabilistic Surface Distance (PSD) loss, generates irregular triangles and introduces holes to the triangulation that increase CD and WA errors. We do not report ablation result for the Probabilistic Chamfer Distance since it is the only loss applied to the Point Sampler module. All methods were trained on TOSCA dataset using the same train-test split.

2.2. Simplification of Textured Meshes.

Although in this study we mainly focus on mesh shape simplification, it is reasonable to assess the appearance of simplified textured meshes. In this experiment we qualitatively examined the texture preservation of simplified meshes. To do so, we applied a checker pattern on TOSCA shapes and evaluated the similarity of the simplified models with the originals. From Figure 2 one may observe that the texture of QEM method is unsettled and that the sharp corners of the checker have become smoother. In contrast, the proposed method carries significant less noise and better preserves the details of the original texture. As an additional experiment, we assessed the proposed method in a real-world mesh scenario, using a textured mesh from a human face (bottom row Figure 2). As it may be observed the proposed method achieves to maintain the texture characteristics of the face.

Method	$N_s/N_{org} = 0.05$				$N_s/N_{org} = 0.1$				$N_s/N_{org} = 0.2$				$N_s/N_{org} = 0.5$			
	CD	WA	LE	NE	CD	WA	LE	NE	CD	WA	LE	NE	CD	WA	LE	NE
Proposed w/o OV	**1.01**	2.21	0.98	0.20	0.42	2.24	0.52	0.15	**0.19**	2.52	0.27	0.12	**0.06**	3.59	0.12	0.08
Proposed w/o EC	1.02	2.20	0.94	0.20	0.43	2.22	0.53	0.16	0.20	2.50	0.27	0.11	0.06	3.58	0.13	0.07
Proposed w/o TC	1.03	2.22	1.54	0.42	0.42	2.24	0.97	0.36	0.19	2.51	0.78	0.32	0.06	3.58	0.58	0.28
Proposed w/o PSD	2.12	10.24	1.35	0.23	1.33	9.14	0.96	0.19	0.96	6.75	0.77	0.18	0.52	5.97	0.54	0.13
Proposed-Full	1.02	**2.17**	**0.90**	**0.19**	**0.42**	**2.21**	**0.47**	**0.15**	0.19	**2.49**	**0.24**	**0.11**	0.06	3.57	**0.10**	**0.06**

Table 1. Quantitative results of the ablation study over the loss functions. OV, EC, TC, PSD denote overlap loss, edge crossing loss, triangle collusion loss and probabilistic surface distance loss, respectively.

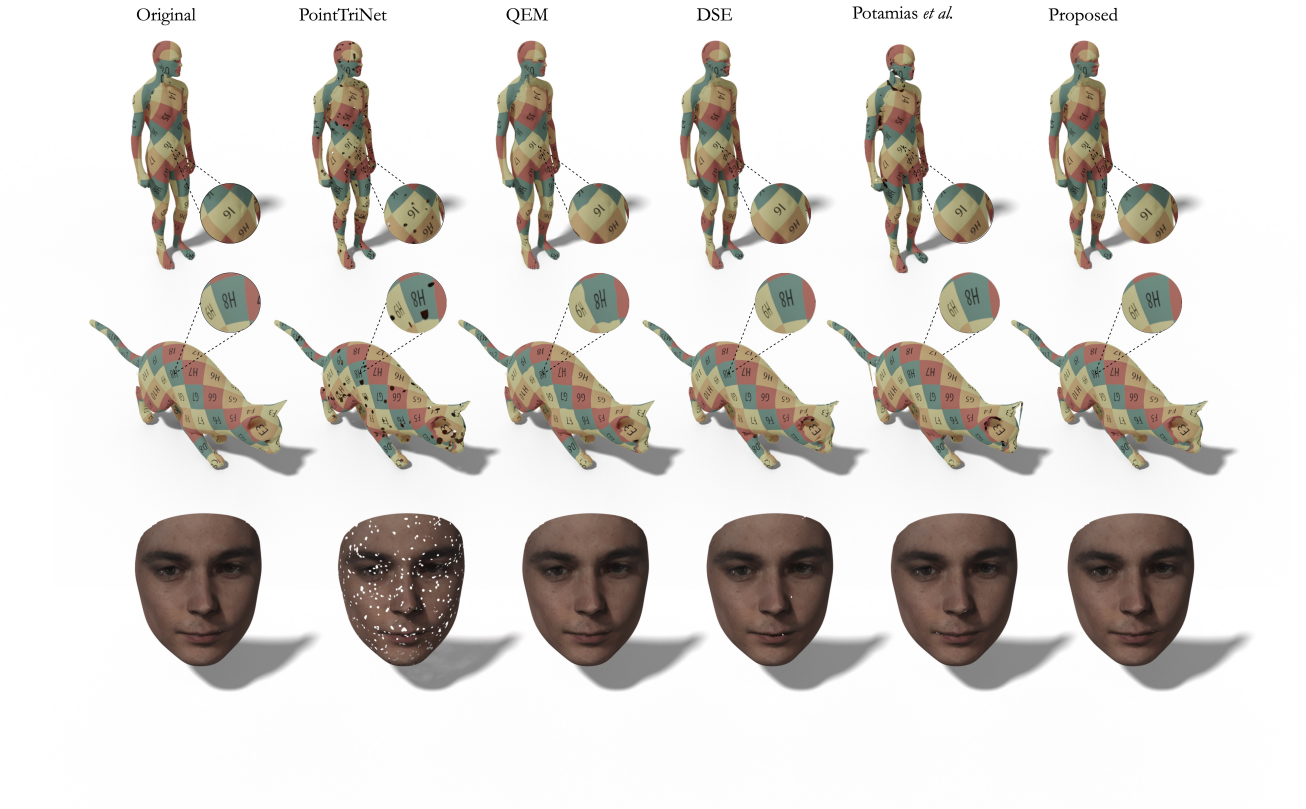


Figure 2. Simplification of textured meshes. For the human shape model (top row) we visualize meshes simplified by 85%, the cat model (medium row) is simplified by 90% and the face model (bottom row) is simplified by 80%

2.3. Intrinsic distances

Additionally to the QEM method utilized as a baseline in the main paper, we evaluated the intrinsic distance preservation for all the baseline methods. In Figure 3 we visualize the Geodesic and the Laplacian distances for the proposed and the baseline methods. All distances are measured form the nose tip of each shape. It can be easily observed that the proposed method preserves both geodesic and spectral distances of the original model compared to the distorted distances produced by the baseline models. In particular, QEM method produces a thoroughly different isolines compared to the original mesh for the cow model. Additionally, PointTriNet [3], DSE [2] and Potamias *et al.* [1] not only they introduce geodesic error, but also their biharmonic distances (spectral) are less smooth compared to the proposed method.

2.4. Simplification of noisy meshes.

Although in real-world applications a noise filtering prepossessing step is always present, we also examined the simplification performance of the proposed method under noise conditions. As mentioned in the main paper, the devised Point Sampler module is less affected by noise compared to its counterparts due to the DevConv structure. Qualitative comparison between the proposed and the baseline models is illustrated in Figure 4. The perfor-

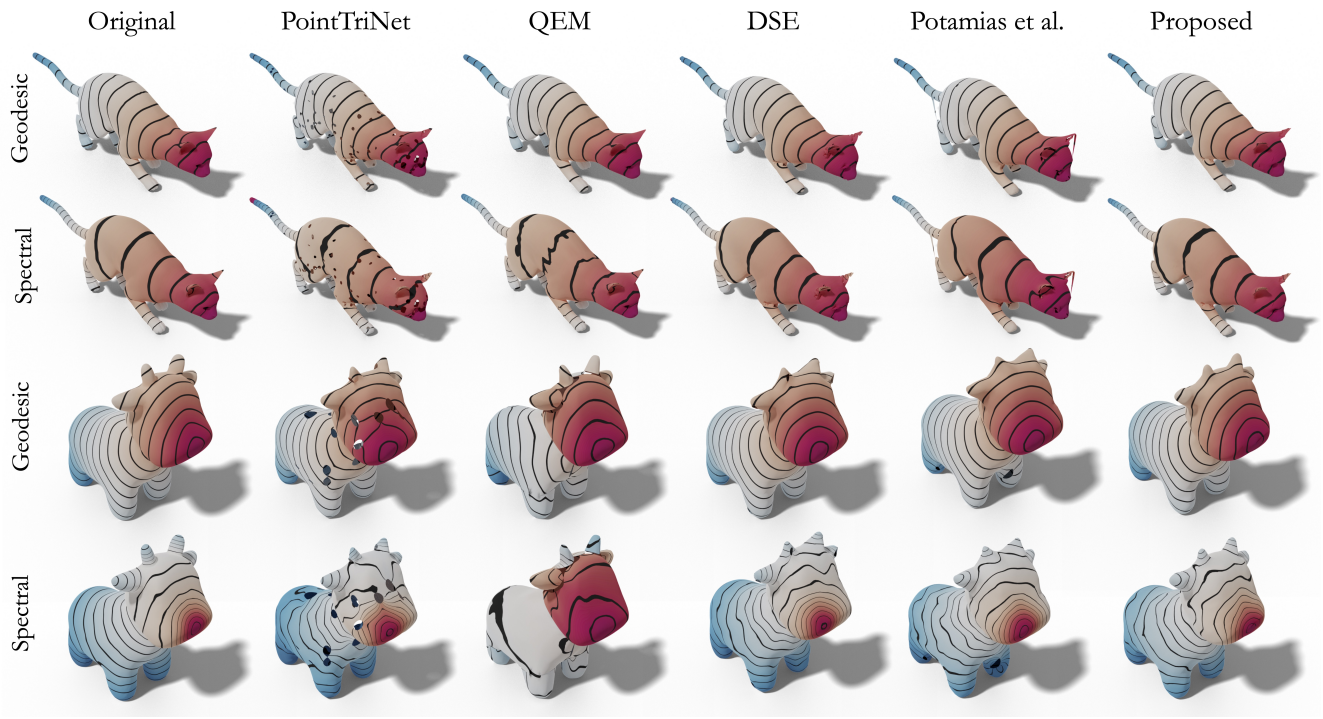


Figure 3. Qualitative comparison of intrinsic distances preservation.

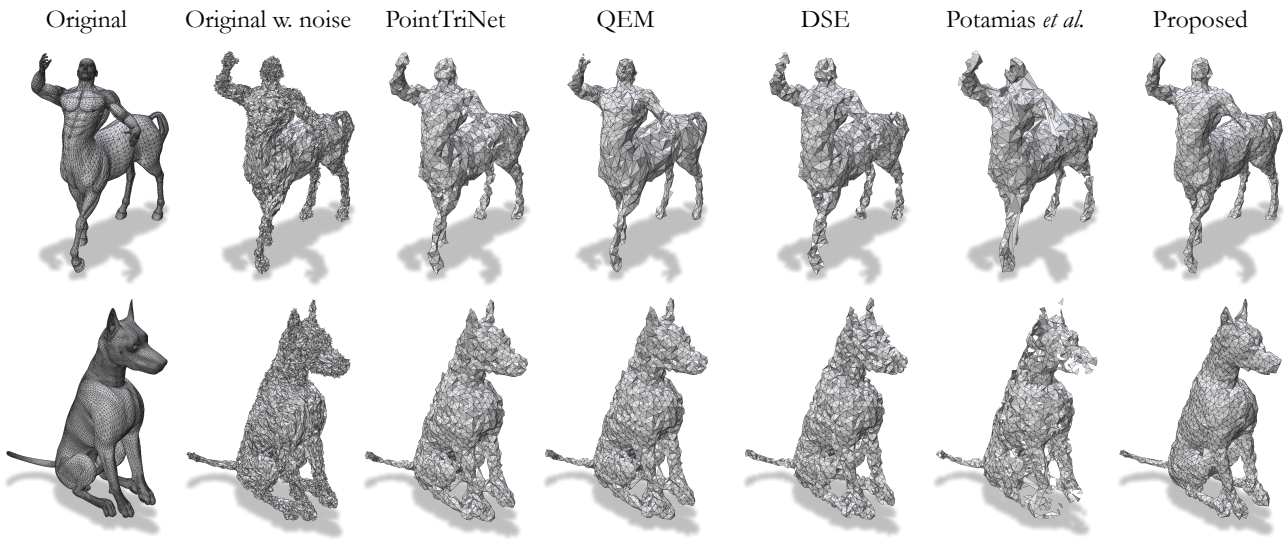


Figure 4. Qualitative comparison of the proposed and the baseline methods on noisy mesh simplification. The top row contains a centaur shape simplified by 90% and the bottom row shows a dog model simplified by 90%. Figure better viewed in zoom.

mance of Point Sampler leads to better triangulation and thus smoother simplified meshes compared to the PointTriNet [3] and DSE [2] modules. QEM method struggles to find the planes associated with each point and generates artifacts to the simplified mesh. Ball pivoting algorithm, utilized in Potamias *et al*. [1], fails to properly triangulate the simplified point cloud and requires careful hyperparameter tuning to avoid irregular triangles. On the contrary, the proposed method produces smooth results, e.g. the rack of the Centaur model, and manages to generate a simplified model that preserves the appearance of the input.

References

[1] Rolandos Alexandros Potamias, Giorgos Bouritsas, and Stefanos Zafeiriou. Revisiting point cloud simplification: A learnable feature preserving approach. *arXiv preprint arXiv:2109.14982*, 2021. 2, 3, 4

[2] Marie-Julie Rakotosaona, Noam Aigerman, Niloy Mitra, Maks Ovsjanikov, and Paul Guerrero. Differentiable surface triangulation. *arXiv preprint arXiv:2109.10695*, 2021. 3, 4

[3] Nicholas Sharp and Maks Ovsjanikov. Pointtrinet: Learned triangulation of 3d point sets. In *European Conference on Computer Vision*, pages 762–778. Springer, 2020. 3, 4