GEOMetrics: Supplemental Material

A. Point to Surface Loss

In this section, we describe the the Distance Between Point and Triangle in 3D algorithm (Eberly, 1999). For a given point P and triangle T, the algorithm computes the minimum distance between the point and any point contained within the triangle. Assuming the triangle is defined by corner point B and directions E_0 and E_1, then any point $T(s,t)$ contained in the triangle can be defined by a pair of scalars (s,t) such that $T(s,t) = B + sE_0 + tE_1$, where $(s,t) \in D = \{(s,t) : s \geq 0, t \geq 0, s+t \leq 1\}$. We can now define the squared distance Q between the point P and any point in the triangle $T(s,t)$ by the following quadratic function:

$$Q(s,t) = as^2 + 2bst + ct^2 + 2ds + 2et + f, \quad (1)$$

where for clarity we denote $a = E_0 \cdot E_0$, $b = E_0 \cdot E_1$, $c = E_1 \cdot E_1$, $d = E_0 \cdot (B - P)$, $e = E_1 \cdot (B - P)$, and $f = (B - P) \cdot (B - P)$. Selecting (s,t) which minimizes $Q(s,t)$ provides the minimum distance between the point P and triangle T. As Q is continuously differentiable, (s,t) can be found at an interior point where $\nabla Q = 0$ or at the boundary of the set D.

In the first case, note $\nabla Q(s',t') = 0$ if and only if s' and t' satisfy the following:

$$s' = \frac{be - cd}{ac - b^2}, \qquad t' = \frac{bd - ae}{ac - b^2} \quad (2)$$

Then if $(s',t') \in D$, we have the minimum distance $Q(s',t') = \min_{(s,t)\in D} Q(s,t)$. Otherwise, the distance minimizing (s,t) must lie on the boundary of D, where either $\{s = 0, t \in [0,1]\}$, $\{s \in [0,1], t = 0\}$, or $\{s \in [0,1], t = 1 - s\}$. In each case $Q(s,t)$, can be reduced to quadratic of one unknown variable, which can be minimized by setting the gradient to 0.

B. Mesh-to-Voxel Mapping Ablation

In this section, we perform an ablation study over the use of 0N-GCN as building block for our Mesh-to-Voxel Mapping network to highlight its impact with respect to the standard GCN layers. To that end, we compare our model on 3 different object classes to an analogous network composed of standard GCN layers with the same number of parameters. Additionally, we assess the influence of pooling across a set of vertices by comparing it to other forms of aggregation such as the one introduced by the Neural Graph Fingerprint (NGF) model (Duvenaud et al., 2015). The results of this ablation study can be found in Table 1 in terms of mean squared error (MSE). As shown in the table, results demonstrate the benefits of the 0N-GCN layers, as well as the max-pooling vertex set aggregation, for this mesh understanding task.

Table 1. Mesh-to-Voxel Mapping Reconstruction MSE scores.

Category	Ours	GCN	NGF
Plane	**0.0089**	0.0104	0.0108
table	**0.0310**	0.0393	0.0360
Chair	**0.0412**	0.0526	0.0486
Mean	**0.0270**	0.0341	0.0318

C. Differentiable Surface Loss Algorithms

This section provides the algorithmic details of both the point-to-point loss (Algorithm 1) as well as the point-to-surface loss (Algorithm 2).

Algorithm 1 Point-to-Point Loss

1: **Input:** Two mesh surfaces M and $\hat{M}$, and number of points n
2: **for** face f in mesh M **do**
3: $\quad A_f = Area(f)$
4: $\quad A_T \mathrel{+}= Area(f)$
5: **end for**
6: Define F s.t. $P(F = f) = \frac{A_f * 100}{A_T}$
7: Define $U = Uniform(0,1)$
8: Define S = []
9: **for** $i = 0$ **to** n **do**
10: $\quad f \sim F$
11: $\quad v_1,\ v_2,\ v_3 = vertices(f)$
12: $\quad u \sim U,\ w \sim U$
13: $\quad r = (1 - \sqrt{u})v_1 + \sqrt{u}(1 - w)v_2 + \sqrt{u}wv_3$
14: $\quad S.append(r)$
15: **end for**
16: Apply lines 1 to 15 to mesh $\hat{M}$ to produce $\hat{S}$
17: $\mathcal{L}_{\text{PtP}} = \sum_{p\in S} \min_{q\in\hat{S}} \|p - q\|_2^2 + \sum_{q\in\hat{S}} \min_{p\in S} \|p - q\|_2^2$

D. Loss Analysis

In this section, we present further analysis of GEOMetrics losses to emphasize the benefits of the introduced point-to-

Algorithm 2 Point-to-Surface Loss

Input: Two mesh surfaces M and $\hat{M}$, and number of points n
for face f in mesh M **do**
  $A_f = Area(f)$
  $A_T \mathrel{+}= Area(f)$
end for
Define F s.t. $P(F = f) = \frac{A_f * 100}{A_T}$
Define $U = Uniform(0, 1)$
Define S = []
for $i = 0$ **to** n **do**
  $f \sim F$
  $v_1,\ v_2,\ v_3 = vertices(f)$
  $u \sim U,\ w \sim U$
  $r = (1 - \sqrt{u})v_1 + \sqrt{u}(1 - w)v_2 + \sqrt{u}wv_3$
  $S.append(r)$
end for
Apply lines 1 to 15 to mesh $\hat{M}$ to produce $\hat{S}$
$L_{\text{PtS}} = \sum_{p \in S} \min_{\hat{f} \in \hat{M}} dist(p, \hat{f}) + \sum_{q \in \hat{S}} \min_{f \in M} dist(q, f)$

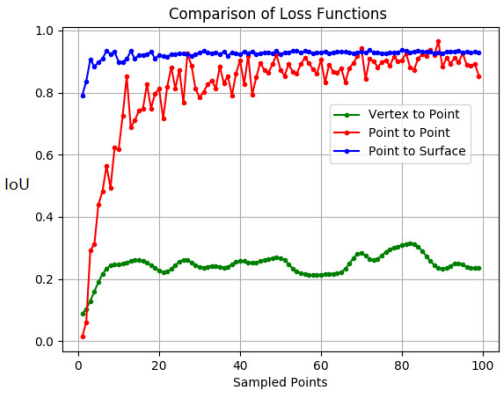


Figure 1. Comparison of vertex-to-point, point-to-point and surface-to-point losses, in terms of IoU, on a toy problem: optimizing the placement of vertices of a square to match that of a target triangle. Results are compared by increasing the number of sampled points on the surfaces they optimize. Vertex-to-point is the loss employed by Wang et al. (2018). Point-to-point and point-to-surface are the losses introduced in our paper.

point and surface-to-point losses over the vertex-to-point loss. To that end, we design a toy problem, which consists in optimizing the placement of the vertices of an initial square surface to match the surface area of a target triangle in 2D. Figure 2 (top) depicts the above-mentioned initial and target surfaces. We optimize the placement of the vertices of the initial square by performing gradient descent on each of the losses independently, and calculate the intersection over union (IoU) of the predicted object and the target triangle. Moreover, in order to assess the impact of the number of points sampled, we repeat this experiment 100 times, increasing the number of sampled points from 1 to 100. Figure 1 shows the results of this experiment. Firstly, we observe that the vertex-to-point loss fails to match the target surface entirely, no matter the number of sampled points. Secondly, we observe that the point-to-point loss performance is notably affected by the number of sampled points. While it exhibits poor performance for lower number of sampled points (e.g. below 20), it rapidly improves as the number of sampled points increases, and ultimately, converges to an average performance, which is only slightly lower than that of the point-to-surface loss. Finally, the point-to-surface loss begins with a far higher IoU and remains the stronger option for nearly all numbers of sampled points.

Figure 2 illustrates qualitative results for the three compared losses when optimizing with 50 points sampled. As can be seen, the point-to-surface deformation of the square better matches the target triangle shape, followed by point-to-point, which somewhat emulates the triangle, and vertex-to-point, which exhibits the poorest results.

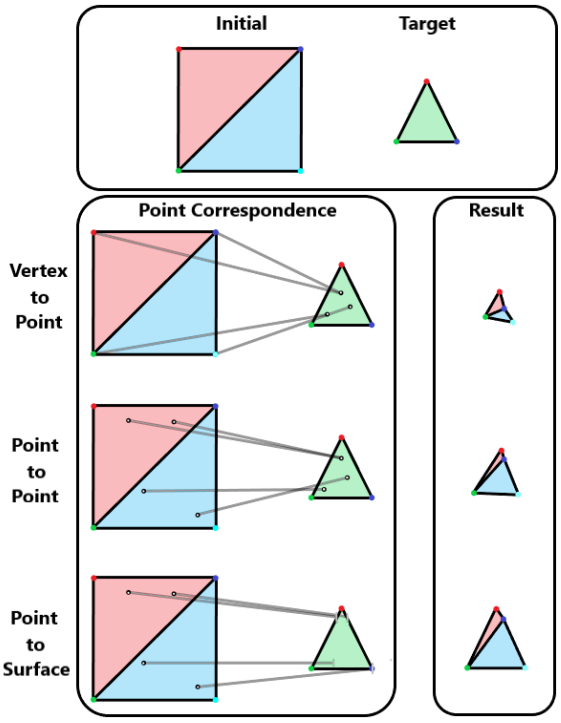


Figure 2. Qualitative comparison of vertex-to-point (Wang et al., 2018), point-to-point and surface-to-point losses on the square-to-triangle problem when using 50 sampled points. We highlight the correspondence between the points in the initial surface and the target surface, which are chosen to be compared, and show the result of optimizing the placement of the vertices when using each of the losses.

E. Network Architectures

In this section, we provide details on the architectures of the networks used in the paper. Table 2 describes the feature extractor network of the mesh reconstruction module. Similarly, Table 3 specifies the mesh deformation network of the reconstruction module. Finally, Table 4 and 5 detail the mesh-to-voxel encoder and decoder architectures, respectively.

F. Single Image Reconstruction Visualizations

Figures 3 and 4 depict additional reconstruction results from each ShapeNet object class, with three objects shown per class.

Layers	1-2	3-8	9	10-11	12	13-14	15	16-18
Output Resolution	224×224	224×224	112×112	112×112	56×56	56×56	28×28	28×28
# Channels	16	32	128	128	256	256	512	512
Kernel Size	3×3	3×3	3×3	3×3	3×3	3×3	3×3	3×3
Stride	1	1	2	1	2	1	2	1
Extracted Layer	-	8	-	11	-	14	-	18

Table 2. **Feature extraction network:** Details of the convolutional neural network architecture used to extract image features. Each layer performs a 2D convolutional, followed by batch normalization (Ioffe & Szegedy, 2015) and a ReLU activation function (Nair & Hinton, 2010). The last row indicates which layer's features are extracted for use in the mesh reconstruction module.

Layers	1	2-14	15
Input Feature Vector Length	3	192	192
Output Feature Vector Length	192	192	3

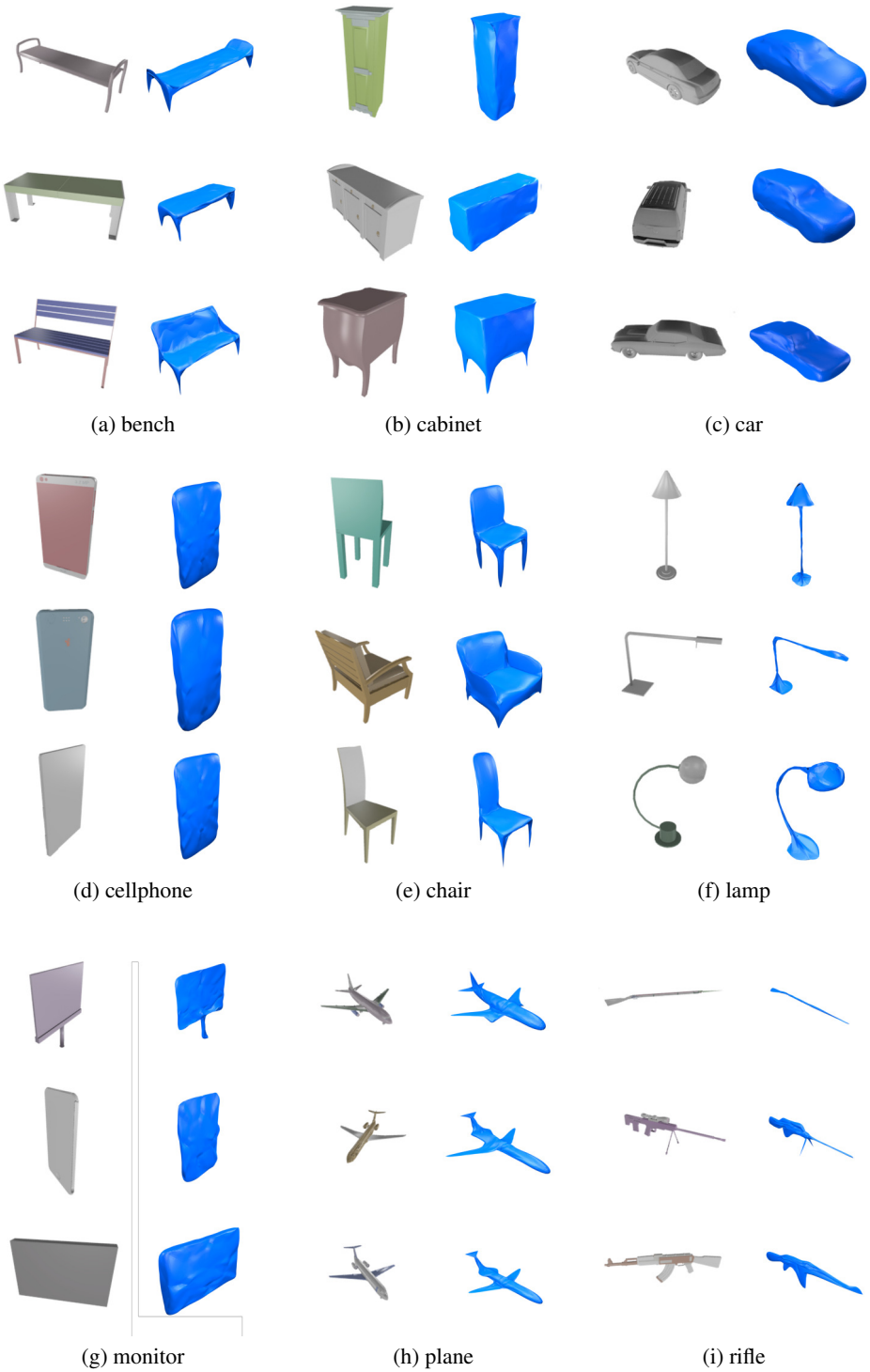
Table 3. **Mesh deformation network:** Details of the graph convolutional network architecture used to compute the mesh deformation in each reconstruction module. Each layer is composed of a 0N-GCN, followed by an ELU activation function (Clevert et al., 2015).

Layers	1	2-4	5	6-7	8	9	10	11	12	13-16	17
Input Feature Dimension	3	60	60	120	120	150	200	210	250	300	300
Output Feature Dimension	60	60	120	120	150	200	210	250	300	300	50

Table 4. **Mesh-to-voxel encoder:** Details of the graph convolutional network architecture used to encode mesh graphs into latent vectors. Each layer is composed of a 0N-GCN followed by an ELU activation function (Clevert et al., 2015), except for the final layer which is a max pooling aggregation over the set of vertices.

Layers	1	2	3	4	5
Output Resolution	$4 \times 4 \times 4$	$8 \times 8 \times 8$	$16 \times 16 \times 16$	$32 \times 32 \times 32$	$32 \times 32 \times 32$
# Channels	64	64	32	8	1
Stride	2	2	2	2	1
Type	DeConv	DeConv	DeConv	DeConv	Conv

Table 5. **Mesh-to-voxel decoder:** Details of the 3D convolutional neural network archictecture used to decode latent vectors into voxel grids. Each layer performs a 3D deconvolution (Shelhamer et al., 2017) with batch normalization (Ioffe & Szegedy, 2015) and an ELU activation function (Clevert et al., 2015), except for the final layer which is a standard 3D convolutional layer.

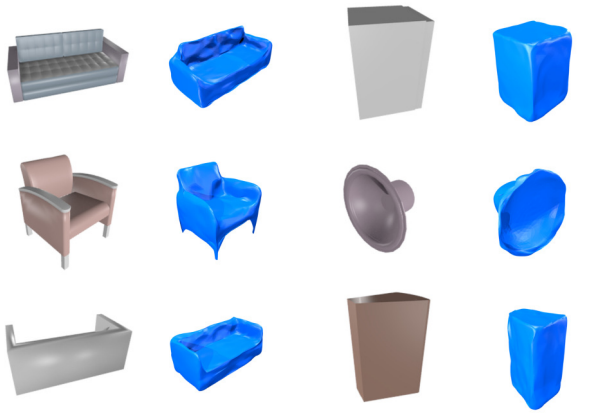


(a) bench (b) cabinet (c) car

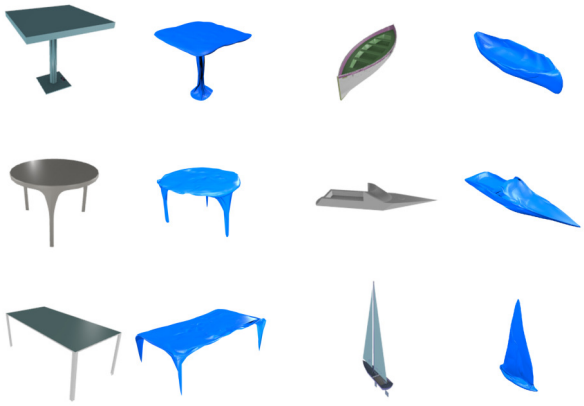
(d) cellphone (e) chair (f) lamp

(g) monitor (h) plane (i) rifle

Figure 3. Single image reconstruction results on bench, cabinet, car, cellphone, chair, lamp, monitor, plane and rifle classes.



(a) sofa (b) speaker



(c) table (d) watercraft

Figure 4. Single image reconstruction results on sofa, speaker, table and watercraft classes.

References

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pp. 2224–2232, Cambridge, MA, USA, 2015. MIT Press.

Eberly, D. Distance between point and triangle in 3d. *Geometric Tools*, 1999.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Shelhamer, E., Long, J., and Darrell, T. Fully convolutional networks for semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):640–651, 2017.

Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., and Jiang, Y.-G. Pixel2mesh: Generating 3d mesh models from single rgb images. *arXiv preprint arXiv:1804.01654*, 2018.