Visualizing Point Cloud Classifiers by Curvature Smoothing Supplementary Materials

Chen Ziwen[1]
chenziwe@grinnell.edu
Wenxuan Wu[2]
wuwen@oregonstate.edu
Zhongang Qi[3]
zhongangqi@gmail.com
Li Fuxin[2]
lif@oregonstate.edu

[1] Grinnell College
IA, USA
[2] Oregon State University
OR, USA
[3] Applied Research Center
Tencent PCG
Shenzhen, China

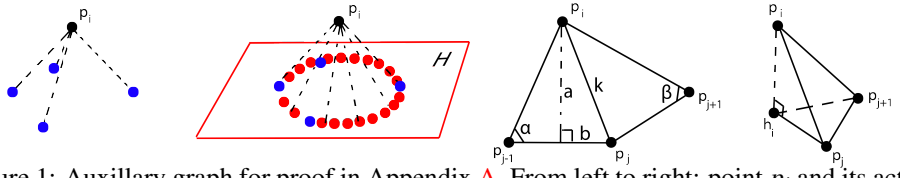


Figure 1: Auxillary graph for proof in Appendix A. From left to right: point p_i and its actual neighbors (in blue), p_i and its virtual neighbors (in red) and the fitted local plane H, enlarged graph of p_i and three of its neighbors, p_i and its projection h_i on the fitted plane H. Note that h_i is also the center of the ring formed by the virtual neighbors.

A Curvature approximation proof

Theorem 1. *Let $p_i \in \mathbb{R}^3$ be a point in point cloud. Let $H = \{x : \langle x, \boldsymbol{n} \rangle + D = 0, x \in \mathbb{R}^3\}, \boldsymbol{n} \in \mathbb{R}^3, ||\boldsymbol{n}|| = 1$ be the plane fitted to the neighbors of p_i. Let h_i be the projection of p_i on H. Assuming p_i's neighbors distribute evenly and densely on a ring surrounding h_i, then the curvature normal at p_i can be approximated by the expression $\frac{1}{2k^2}(h_i - p_i)$, where k is the distance from p_i to any of its neighbor.*

Proof. Our proof will refer to Fig. 1.

[1] has already showed that on a 3-D mesh, given a point p_i and its neighbors, the local "carvature normal" can be calculated using

$$\frac{1}{4A} \sum_{j \in \mathcal{N}(p_i)} (\cot \alpha_j + \cot \beta_j)(p_j - p_i) \tag{1}$$

where A is the sum of the areas of the triangles having p_i as common vertex and α_j, β_j are the two angles opposite to the edge e_{ij} (i.e. $p_j - p_i$). This arrangement is demonstrated Fig. 1.

Since point cloud data are usually sparse and noisy, we want to utilize some mechanism to mitigate this sparsity and irregularity. Here, we first fit a local plane to p_i's neighborhood, and then we define the notion of "virtual neighbors" as a means to fill in the gaps left by the "actual neighbors". We assume the "virtual neighbors" distribute evenly and densely on a ring surrounding h_i on the fitted plane H, each having the same distance k to p_i (k is calculated using the average distance of the actual neighbors). Let a be the distance from p_i to each edge $e_{j,j+1}$. Let b be half of the length of $e_{j,j+1}$. Thus we can calculate A in Eq. 1 as $n \cdot ab$. Since we assumed the points are distributed evenly, we have $\cot\alpha = \cot\beta = \frac{b}{a}$. Thus we have the curvature normal to be $\frac{1}{4A}\sum_j(\cot\alpha_j + \cot\beta_j)(p_j - p_i) = \frac{1}{4nab} \cdot \frac{2b}{a}\sum_j(p_j - p_i) = \frac{1}{2na^2}\sum_j(p_j - p_i)$.

Note that the vector $p_j - p_i$ is equal to $(p_i - h_i) + (h_i - p_i)$, and it can be easily shown that $\sum_j(p_i - h_i) = \vec{0}$. Thus we can continue derive the curvature normal to be $\frac{1}{2na^2}\sum_j(p_j - p_i) = \frac{1}{2na^2}\sum_j(h_i - p_i) = \frac{n}{2na^2}(h_i - p_i) = \frac{1}{2a^2}(h_i - p_i)$. Since we assume the points are distributed densely, thus we have as $n \to \infty$, $a \to k$. Hence, the curvature normal at p_i can be approximated by the expression

$$\frac{1}{2k^2}(h_i - p_i) \tag{2}$$

where $h_i - p_i$ is just the vector pointing from p_i to the local plane H as in Eq. **??** and **??**. This equation makes sense in that when the distance from p_i to H is fixed, the further away the neighbors are, the "flatter" the surface at p_i is. □

In our actual experimentation however, we found that due to the extremely irregular distribution of the point cloud data, the neighborhood distance is misleading sometimes rather than helpful. Thus, in our final algorithm, we abandon the distance information $\frac{1}{2k^2}$ and directly use the vector pointing from p_i to plane H as our approximation for the local curvature.

B Implementation Details for Point cloud smoothing

(a) The "isolated neighborhood" ($K = 2$). Red dots are closed under $\mathcal{N}(\cdot)$ operation, losing contact with other points.

(b) The "false neighbor" ($K = 3$). The rightmost red dot is a false neighbor for the red dot in the center, as indicated by the line drawing on the right.

Figure 2: Two issues innate to point clouds (due to the missing edge information between vertices).

An important implementation detail for the point cloud smoothing algorithm is that the size of the neighborhood we use increases as the smoothing goes further. In practice, after every 4 rounds of erosion and dilation, we expand the neighborhood size by 20 points. The reason for this is twofold. On one hand, there might exist isolated neighborhoods in a point cloud (i.e. a set of points that is closed under the $\mathcal{N}(\cdot)$ operation) as shown in Fig. 2a. If the curvature information cannot be propagated to the entire point cloud, the algorithm will fail. On the other hand, a larger neighborhood speeds up the smoothing process. As mentioned in [2], the time step restriction $(0 < \lambda < 1)$ results in the need of hundreds of updates to cause a noticeable smoothing using the original implementation in [3]. Note that however, we also cannot make the neighborhood size too large, especially at the beginning, due to the "false neighbor" issue innate to the point cloud data structure (explained in Fig. 2b).

C Smoothing Algorithm Evaluation Metrics

Figure 3: Left: A 2-D ellipse shape with 202 unevenly distributed points. Middle: Taubin smoothing. Right: Our smoothing. In the case of Taubin smoothing, highly concentrated areas are pushing points outward, resulting in an undesired shape (i.e., more frequent change in curvature), while our algorithm is not influenced by point density (and thus, the constant curvature is achieved as desired).

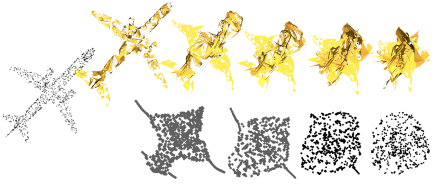


Figure 4: Upper row: meshing a point cloud and then applying Laplacian smoothing. Lower row: our smoothing algorithm.

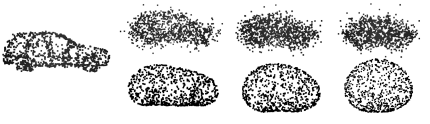


Figure 5: Upper row: mean-curvature-flow algorithm utilizing curvature from fitted quadratic surfaces to local neighborhoods. Lower row: our smoothing algorithm.

In the experiment section of our paper, we use the following three metrics to compare our smoothing method against the baselines (Fig. 3, 4 and 5 show some example comparisons in pictures):

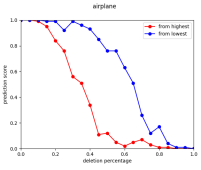
CSD. Curvature standard deviation. We regard large curvature on a shape as "features", and we want to eliminate those "features" through the smoothing algorithms. As we remove the most distinct curvatures on the surface like edges and corners, the standard deviation of the curvatures will decrease, due to the elimination of the large outliers. In our experiment, we measure the distance from each point to its locally fitted plane ($K = 60$) as an approximation of the local mean curvature.

MR. Min-max ratio. Assuming that the underlying manifold is closed, our smoothing should eventually morph the point cloud into a sphere. Hence, we propose to evaluate the ratio between the length on the short side and the long side of the point cloud. This is computed by first applying principal component analysis (PCA) to the point cloud and finding

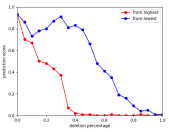
the top two principal components, say $\vec{u}$ and $\vec{v}$. Then we compute the ratio between ranges of the values on these two principal directions. The closer this ratio is to 1, the better.

DDS. Density distribution similarity. We want the morphing process to be smooth in that the density distribution of each point cloud to *remain the same* throughout the morphing process. We conduct the kernel density estimation at each point (using a Gaussian kernel with $\sigma = 0.1$) to obtain the density distribution of the entire point cloud, and then compare the similarity between the distributions of two consecutive blurred levels using the Kolmogorov-Smirnov test (p-values are recorded as results).

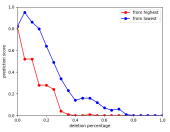
D Curve figures



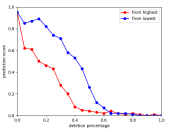
(a) Airplane.



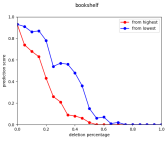
(b) Bathtub.



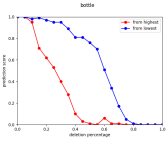
(c) Bed.



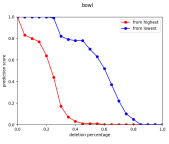
(d) Bench.



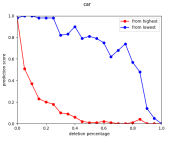
(e) Bookshelf.



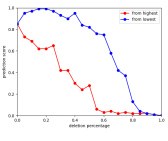
(f) Bottle.



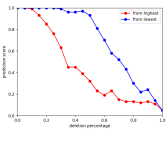
(g) Bowl.



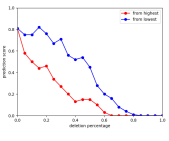
(h) Car.



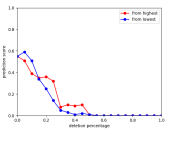
(i) Chair.



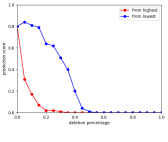
(j) Cone.



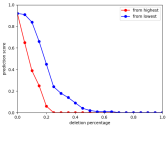
(k) Cup.



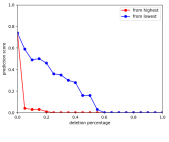
(l) Curtain.



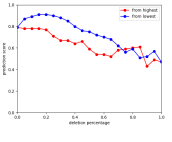
(m) Desk.



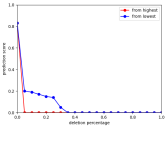
(n) Door.



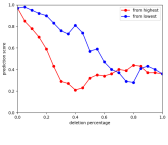
(o) Dresser.



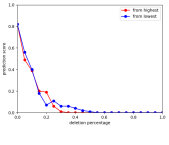
(p) Flower pot.



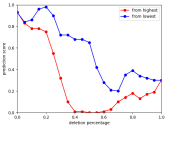
(q) Glass box.



(r) Guitar.

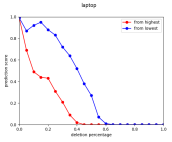


(s) Keyboard.

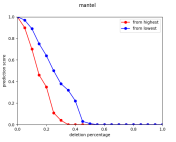


(t) Lamp.

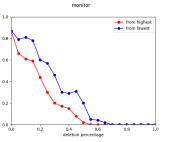
Figure 6: *Deletion* and *insertion* curves for all 40 classes in ModelNet40 for PointConv. Horizontal axis is the deletion percentage (top 5%, 10%, etc.), and vertical axis is the predicted class score. The red line is the *deletion* curve which blurs points from highest mask values, and the blue line is the *insertion* curve (if read from right to left) which blurs points from lowest mask values.



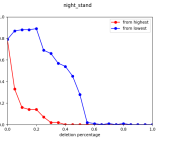
(u) Laptop.



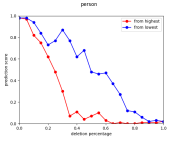
(v) Mantel.



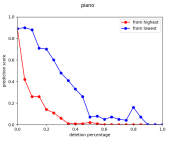
(w) Monitor.



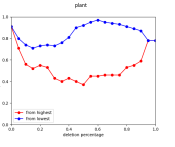
(x) Night stand.



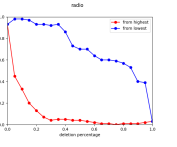
(y) Person.



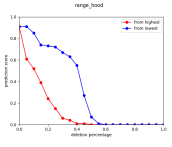
(z) Piano.



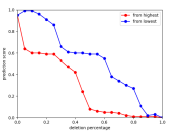
(aa) Plant.



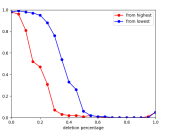
(ab) Radio.



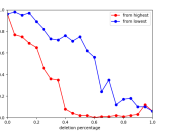
(ac) Range hood.



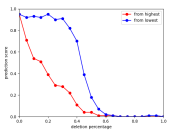
(ad) Sink.



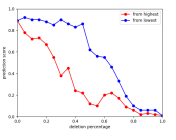
(ae) Sofa.



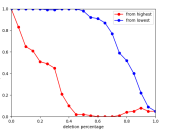
(af) Stairs.



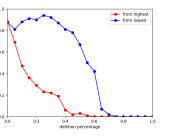
(ag) Stool.



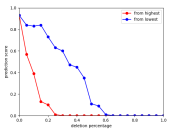
(ah) Table.



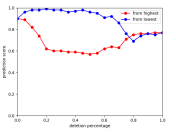
(ai) Tent.



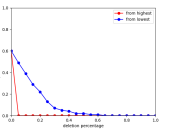
(aj) Toilet.



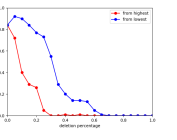
(ak) TV stand.



(al) Vases.



(am) Wardrobe.



(an) Xbox.

Figure 6: *Deletion* and *insertion* curves for all 40 classes in ModelNet40 for PointConv. Horizontal axis is the deletion percentage (top 5%, 10%, etc.), and vertical axis is the predicted class score. The red line is the *deletion* curve which blurs points from highest mask values, and the blue line is the *insertion* curve (if read from right to left) which blurs points from lowest mask values. (cont.)

References

[1] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324. Citeseer, 1999.

[2] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Siggraph*, volume 98, pages 105–114, 1998.

[3] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 351–358, New York, NY, USA, 1995. ACM. ISBN 0-89791-701-4.