Parameter-Free Connectivity for Point Clouds

Diana Marin[a], Stefan Ohrhallinger[b] and Michael Wimmer[c]
Institute of Visual Computing & Human-Centered Technology, TU Wien, Austria

Abstract: Determining connectivity in unstructured point clouds is a long-standing problem that has still not been addressed satisfactorily. In this paper, we analyze an alternative to the often-used k-nearest neighborhood (kNN) graph - the Spheres of Influence Graph (SIG). We show that the edges that are neighboring each vertex are spatially bounded, which allows for fast computation of SIG. Our approach shows a better encoding of the ground truth connectivity compared to the kNN for a wide range of k, and additionally, it is parameter-free. Our result for this fundamental task offers potential for many applications relying on kNN, e.g., parameter-free normal estimation, and consequently, surface reconstruction, motion planning, simulations, and many more.

1 INTRODUCTION

In recent times, point clouds have gained popularity as a data representation, owing to advancements in scanning technology. However, the initial state of disorganized points necessitates further processing to reconstruct the original surface's connectivity from which they were sampled. This connectivity allows for the estimation of surface properties such as neighborhoods or normals. Various methods are available for estimating the intrinsic properties of point clouds, but they often rely on selecting the appropriate parameters tailored to the specific data type. For instance, this includes determining neighborhood connectivity using the user-specified 'k' in k-nearest neighbor (kNN) calculations. Additionally, point cloud sampling can be non-uniform due to the acquisition method or tainted by artifacts like noise and outliers. These conditions make the process of parameter selection for scanned data a challenging and time-consuming endeavor.

Extracting connectivity from point clouds is a significant research challenge, not only due to its essential role as a first step in surface reconstruction but also because it forms the foundational input for graph-based learning tasks involving point clouds. Regarding surface reconstruction, connectivity graphs serve as the fundamental structure on which triangles are constructed, and as a computation base for normals, which are sometimes required as part of the reconstruction process. For learning-based tasks, establishing a method for connecting the input points that closely aligns with the original surface is imperative, since the creation of an actual surface is not necessary.

We propose a fast computation of the *spheres-of-influence* graph (SIG) and we analyze its properties as a proximity graph. This graph recovers the original connectivity of the surface better than the widely used kNN graph, as it can be seen in Figure 1, while dropping the need for users to search for a suitable parameter that typically varies depending on the input and its local properties. Our method achieves the best results on various models with different features. Hence, our method offers a good scaffolding for further processing of point clouds, such as normal estimation, surface reconstruction, or graph convolutional neural networks. We show that our method not only encodes the original connectivity better than kNN but, as an application, provides a good base for normal estimation, while remaining parameter-free.
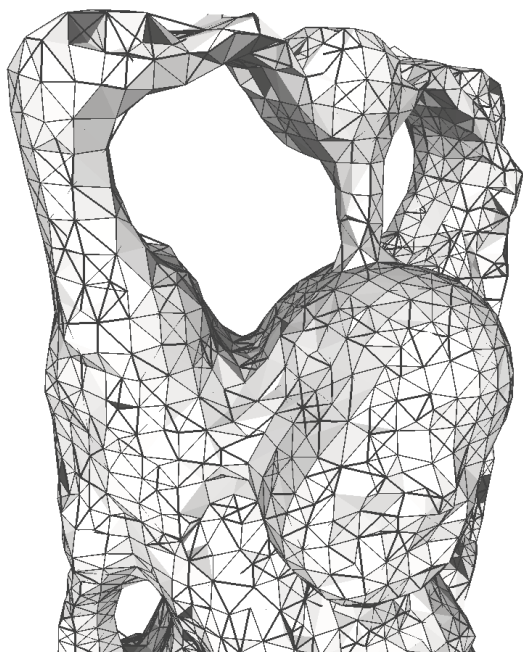
We present the following contributions:

- We introduce an effective method for constructing the spheres-of-influence graph, proving novel spatial constraints for this parameter-free graph.

- Our method is evaluated against ground truth meshes, demonstrating superior connectivity representation with a reduced space requirement when compared to traditional kNN graphs.

- As an application, we offer an analysis of normal computation on point clouds, highlighting our method's ability to deliver competitive results.
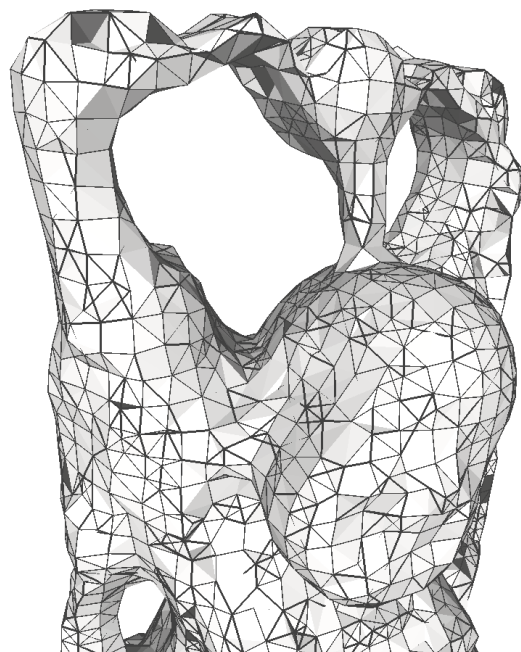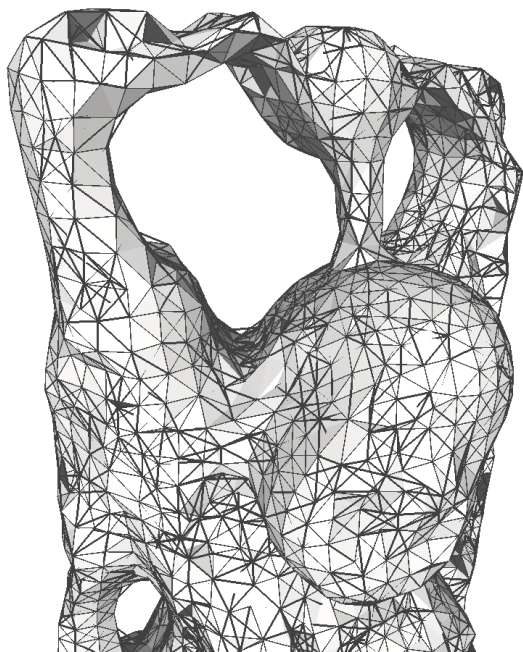
[a] https://orcid.org/0000-0002-8812-9719
[b] https://orcid.org/0000-0002-2526-7700
[c] https://orcid.org/0000-0002-9370-2663
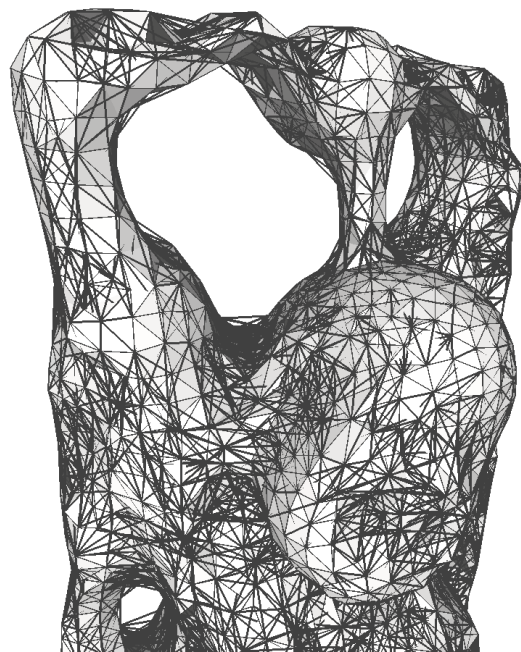
Figure 1: Results of our SIG and the kNN graphs for $k = \{6, 10, 20\}$ for the connectivity of the Buddha statue from the Stanford repository. Our graph captures the connectivity well, without many redundant edges, and without the need for a parameter compared to the kNN graphs. The original surface is shown in light gray, with the various graphs overlaid in black.

2 RELATED WORK

Connectivity. Determining the neighborhood of unstructured points in 3D requires a local description of the points' relations depending on their distance. A k-neighbourhood is strongly influenced by the choice of the parameter k, which could result in over-smoothing if the covered area is too large, or sensitivity to noise for small areas. Hence, picking a value depends heavily on the particular type of data and how it is sampled. Being able to choose k usually requires more time to find a suitable value, but it allows the freedom of dealing with varied data types.

Various subsets of the Delaunay triangulation can be used as connectivity graphs, such as the relative neighborhood graph, the Gabriel Graph, the α-complex (Edelsbrunner et al., 1983), and the β-skeleton (Kirkpatrick and Radke, 1985). For an edge pq to exist in the relative neighborhood graph, there cannot exist another point r that is closer to p or q than they are to each other. In the Gabriel graph, two points - p, q, are connected if the closed disc with diameter pq, passing through p and q, contains no other samples. The α-complex contains all simplices of the Delaunay triangulation that can be enclosed with a circle of radius $1/\alpha$, empty of other samples. The β-skeleton has been introduced as a scale-invariant version of α-shapes, where the edge pq is part of the graph if angles prq are bound by a threshold - β. However, we do not include these graphs in our evaluation, since, as subsets of the Delaunay triangulation (Jaromczyk and Toussaint, 1992b), they are computed by pruning the triangulation, which is slower than both kNN and SIG computations. Furthermore, since the triangulation computation is a prerequisite for these methods, one could directly reconstruct the surface, bypassing the need for a proximity graph.

Normal Estimation. An important application of connectivity retrieval is represented by normal estimation. Normal estimation for point clouds has been a heavily researched area of computer graphics, as it usually represents a first step or requirement in surface reconstruction, e.g., for Poisson reconstruction (Kazhdan et al., 2006) or data-driven approaches such as Point2Surf (Erler et al., 2020). One of the basic methods for computing normals for unstructured point clouds is using Principal Component Analysis (PCA). This method considers a local patch of vertices and finds the axis of variance with the least amount of variance since the points should vary the least in the normal direction. The relation between neighborhood choice and normal estimation has been extensively studied (Mitra and Nguyen, 2003). Other methods improve on the tangent plane estimation by using a weighted approach when considering the local neighborhood (Pauly et al., 2003) or by fitting algebraic spheres (Guennebaud and Gross, 2007). However, these methods all require a good parameter for choosing an appropriate neighborhood size, which we do not need with our simple parameter-free method.

Another avenue for estimating normals for point clouds has been developed with the concept of *poles* (Amenta and Bern, 1999). This method is based on computing the Voronoi diagram on the input points and extracting the normals as the line connecting each sample point and the farthest Voronoi vertex to their Voronoi cell (the pole). This method is sensitive to noise, but it has been improved to handle noisy samples (Dey and Goswami, 2006) where it requires additional parameters.

Data-Driven Approaches. Recently, multiple data-driven approaches have been developed, which usually take advantage of large data repositories to learn the geometric relations between the point clouds' structure and their expected normals. Here (Boulch and Marlet, 2016), they are using a Hough transform to estimate normals. PCPNet (Guerrero et al., 2018) builds on the PointNet architecture (Charles et al., 2017) and uses local patches to estimate the properties of point clouds with various noise levels and sampling densities. However, these types of methods require a long processing time and possible re-training depending on the type of data.

An alternative to dealing with noise in the existing point clouds is to first denoise them and then use a connectivity/normal computation approach that works well on clean data. In this direction, data-driven approaches have been dealing with noise in (Rakotosaona et al., 2020), where they build on PCPNet to classify outliers and then reproject noisy samples on the original surface. (Luo and Hu, 2021) use a noisy point distribution model to estimate scores and the direction of the surface. Classical, analytical approaches for denoising include filtering in various forms, such as the bilateral filter that takes normals into account (Digne and de Franchis, 2017) or voting schemes for feature detection that later help in normal repositioning (Liu et al., 2020). This kind of preprocessing could enable our method to also connect noisy point clouds since it is designed as an interpolating method to recover the connectivity.

Moreover, connectivity retrieval has applications in other data-driven approaches. Methods that use graph convolutional neural networks on point clouds for various purposes, such as reconstruction or segmentation, create a graph on the input points (and

sometimes on the deeper layers of the network as well) and use it to learn about the data and infer the surface or various labels on the input points (Shi and Rajkumar, 2020; Wang et al., 2019). Usually, they use *k*NN or a fixed radius neighbor search as their connectivity encoding, which could be replaced by our method to improve results.

Spheres-of-Influence. The *spheres-of-influence* graph (SIG) has been introduced as a clustering method (Toussaint, 1988). Two vertices are connected in the SIG if their distance is less or equal to the sum of the distances to their respective nearest neighbors. This graph was previously used for estimating local densities for surface reconstruction (Klein and Zachmann, 2004), but our method aims to recover connectivity, not a surface. Moreover, the rarely used SIG has recently gotten attention in reconstruction methods (Marin et al., 2022; de Figueiredo and Paiva, 2022) where combined with the Delaunay graph it showed promising results for curve and region reconstruction.

3 METHOD

We define a mesh as a collection of vertices - $V = \{p \in \mathbb{R}^3\}$ and triangulated faces - $T = \{(a,b,c)|a,b,c \in V, a \neq b \neq c \neq a\}$. Using only the set of unstructured points $V = \{p \in \mathbb{R}^3\}$, we aim to recover a set of edges that connect the given samples as similar as possible to the original connectivity of the mesh which is encoded in T. We define similarity here as edge connectivity, instead of the exact triangulation of the original mesh. Our method uses the spheres-of-influence graph, and we provide an improved algorithm to efficiently compute this proximity graph. We are using the vertices of triangulated meshes as our input in order to have access to a ground truth connectivity to which we can compare our results.

Spheres-of-Influence Graph. The SIG contains all the edges (v_0, v_1) such that for their respective endpoints, v_0 and v_1, the following holds:

$$\|v_0, v_1\| \leq nn(v_0) + nn(v_1), \tag{1}$$

where $\|a,b\|$ represents the Euclidean distance between points a and b, and $nn(a)$ represents the distance between a and its nearest neighbor. This condition can be interpreted visually as centering a ball at each vertex, with radius equal to the distance to the nearest neighbor of that vertex, and connecting all points whose balls intersect, as it can be seen in

Figure 2. This graph encodes the spatial proximity of vertices well, without the need for a parameter. It contains the Nearest Neighbor graph (Toussaint, 1988), as the edge between points and their respective nearest neighbor will always satisfy the condition.
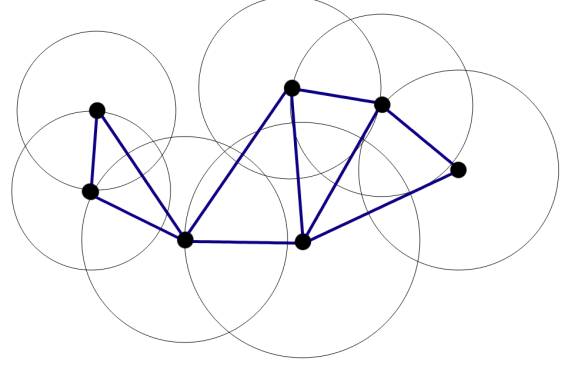

Figure 2: Visual representation of the SIG connectivity - the intersections of circles generate the set of SIG edges.

Various properties of the sphere-of-influence graph have been investigated, such as the bound on the size of the graph (Dwyer, 1995) or its behavior under different metric spaces (Michael and Quint, 1999). However, even if the number of edges has been proven to be linear, algorithms to efficiently compute the graph have not yet been researched, especially considering that SIG is not included in the Delaunay triangulation and hence, the latter cannot be used as a starting point for pruning (Jaromczyk and Toussaint, 1992a). We show that the neighbors of each vertex in SIG are bounded inside a fixed radius - Figure 3, and we use this spatial constraint to improve the speed of the computation.

Theorem 1. *For all vertices $a \in V$, all SIG edges $(a,b), b \in V, a \neq b$, are contained in a radius of at most $2nn(a)$ from a or in a radius of at most $2nn(b)$ from b, where $nn(v)$ is the distance between v and its nearest neighbor, for $v \in V$.*

Proof. We will prove the statement by contradiction. Let us compute the distance to the nearest neighbor for each vertex and store the result as $nn(v)$ for all vertices $v \in V$. For each vertex v, retrieve all the vertices within $2nn(v)$ distance of v and only store the edges that satisfy the SIG criterion. Let us call the obtained graph FastSIG (FSIG).

Since we are trying to prove Theorem 1 by contradiction, we assume that there exists at least an edge that is outside the range boundaries we have defined (within $2nn(a)$ for each vertex). This means that there has to exist an edge (a,b) which is in SIG but not in
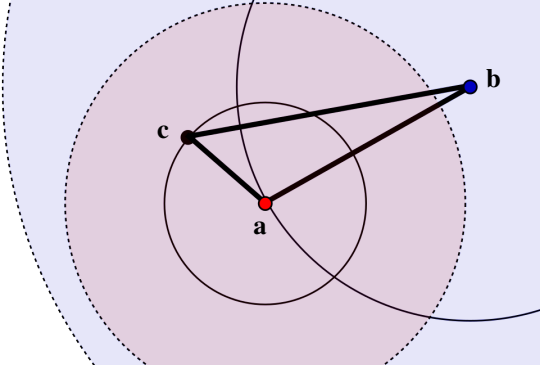
Figure 3: We illustrate the spheres of influence of vertices **a** and **b** in 2D as outlined circles, with a radius equal to the distance to their respective nearest neighbor. The bounding radius of the incident SIG edges are colored disks with dashed borders of radius equal to twice the SIG radius. We can observe that even if **b** is not in the incidence radius of **a**, **a** is contained in **b**'s radius, showing that SIG edges are contained in at least one of the endpoints' incidence radii.

FSIG. Hence,

$$\|a,b\| \leq nn(a) + nn(b), \tag{2}$$

since $(a,b) \in$ SIG and

$$\|a,b\| > 2nn(a), \tag{3}$$

from $(a,b) \notin$ FSIG, while checking a's neighbours which do not include b and

$$\|b,a\| > 2nn(b), \tag{4}$$

from $(a,b) \notin$ FSIG, while checking b's neighbours which do not include a. Summing the last two equations, we get:

$$2\|a,b\| > 2nn(a) + 2nn(b), \tag{5}$$

since $\|a,b\| = \|b,a\|$. Dividing by 2, we get:

$$\|a,b\| > nn(a) + nn(b), \tag{6}$$

which contradicts our assumption of $(a,b) \in$ SIG. Since we have arrived at a contradiction, we have proved that all SIG edges are in FSIG as well. $\square$

Having proved that SIG edges live in a fixed boundary from every node, we use this information to retrieve all the neighboring nodes within the given radius using a *kdtree*, as presented in Algorithm 1.

4 RESULTS

We are aiming to provide an alternative for kNN that is parameter-free, fast to compute, efficient to store,

Data: $V = \{v \in \mathbb{R}^3\}$
Result: SIG=$\{(a,b) : a,b \in V, a \neq b, \|a,b\| \leq nn(a) + nn(b)\}$
SIG=$\{\}$;
create kd-tree KT of V;
for $v \in V$ **do**
     find $nn(v)$ using KT;
end
for $v \in V$ **do**
     $N = KT$.findNbrInRange($2nn(v)$);
     for $u \in N$ **do**
         if $\|u,v\| \leq nn(u) + nn(v)$ **then**
   SIG += (u,v);
         end
   **end**
end

Algorithm 1: Fast SIG computation.

and achieves better or similar surface properties to kNN for a wide variety of data. In this section, we will show how SIG satisfies all these requirements, representing a good, parameter-free alternative to kNN for estimating unstructured point clouds' connectivity.

We have tested our method on a varied dataset of points clouds exhibiting various features, such as various types of non-uniform sampling and sharp edges. We compared our results to the kNN neighborhood for connectivity recovery, for usual values of $k = \{6, 10, 20\}$. We did not use k-values lower than 6 since this is the average degree of vertices in triangulated meshes, as can be shown using Euler's formula. As an application, we have also computed normals using PCA from our graph and from the kNN graphs and compared these with the face-based normals of the original meshes.

Dataset. We have used a subset of the Thingy10K (Zhou and Jacobson, 2016) dataset of meshes that are manifold and have less than 1k vertices for our quantitative results, consisting of around 3k different models. Most of these meshes exhibit CAD-like features in the form of having samples mainly along edges, with thin and long faces. This type of sparse sampling affects the results of our connectivity measures, as, on average, none of the investigated connectivity graphs can perfectly capture the original connectivity, but we have chosen to include this type of data as well to test the resilience of our method in the presence of sparse sampling. For qualitative results, we have included some of the meshes from the Stanford repository (https://graphics.stanford.edu/data/3Dscanrep/) - the resulting graphs using the Stanford bunny are presented in Figure 4.
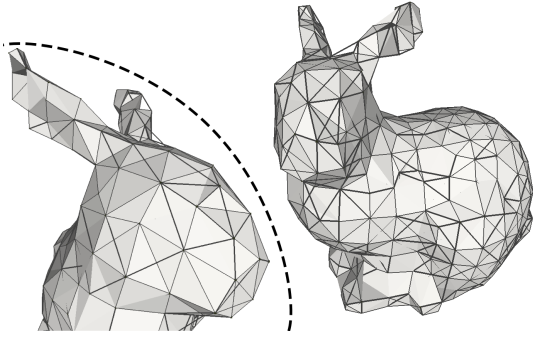
(a) SIG edges computed for the Stanford bunny. The edges mostly follow the original triangulation, with additional diagonals, but without any edges that incorrectly connect the input samples.
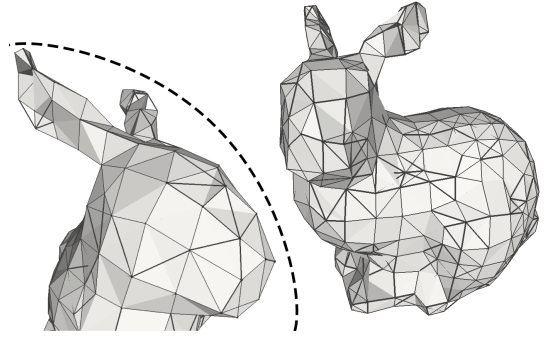
(b) 6NN graph of the Standford bunny. The close-up shows that not all edges of the face are captured, but additional edges appear close to the ears.
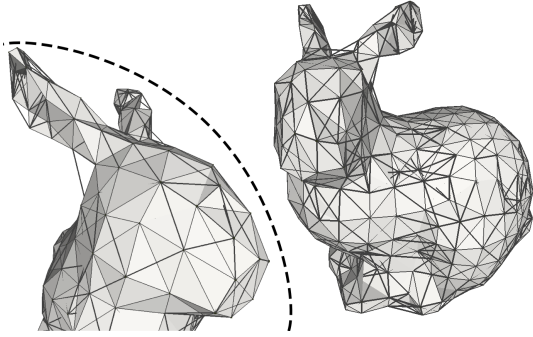
(c) 10NN graph of the input samples. Redundant edges are visible around the ears.
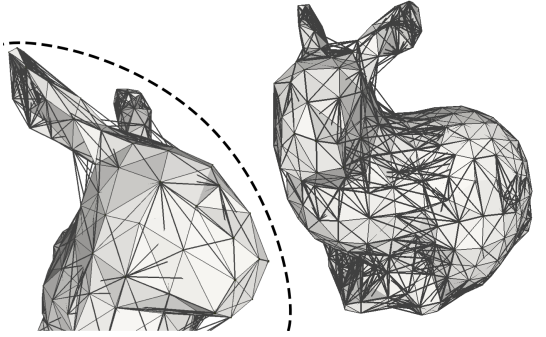
(d) 20NN graph of the input points. Many redundant edges are visible on the ears and the body.

Figure 4: Results of our graph and kNN for $k = \{6, 10, 20\}$ on the Stanford bunny, where the original surface is presented in gray, with various connectivity graph edges overlaid in black.

Connectivity. The connectivity of the ground truth is what our graph aims to reconstruct. However, the original edge set is not a unique representation of the intrinsic connectivity of points, as it is highly dependent on the chosen triangulation, e.g., edges may flip, as can be visible in Figure 5. Hence, we do not aim to reproduce the exact edges of the ground truth meshes (i.e., comparing the 1-ring of each vertex), but to create a good approximation of the connectivity of the entire surface.

In order to quantify how close our graph is to the original connectivity, we used the DeltaCon metric (Koutra et al., 2016), where a value of 0 means the graphs are completely different and a score of 1 implies identical graphs. DeltaCon computes a matrix of node-to-node influence for each graph and the final score is a difference between these two n^2 matrices. This is equivalent to using m-ring neighborhoods ($m \in [1, n]$) for each vertex with decreasing weights as we move farther from the current node. Moreover, this metric benefits from edge awareness - disconnecting changes are penalized more than removing an edge from a complete graph, and this is a property that highly influences the type of connectivity we want to measure for our method. For a detailed definition of the metric and its implementation, we direct to the original works (Koutra et al., 2016).

We have computed DeltaCon for the initial 3431 meshes from Thingy10K, with various numbers of vertices and different sampling densities. The results are presented in Figure 6, where SIG consistently obtains higher scores than the kNN graphs. The results are clustered in equally sized buckets depending on the number of vertices. For each bucket, we present the averaged DeltaCon measurement over all inputs with the total number of vertices in the specified range. Even if the metric does not achieve 1, as the graphs are not identical (as we do not aim for this), our graph manages to encode the original connectivity better than the kNN graphs. The maximum DeltaCon value achieved by all methods is also lower than 1 due to the sparse sampling of the dataset, as mentioned previously.
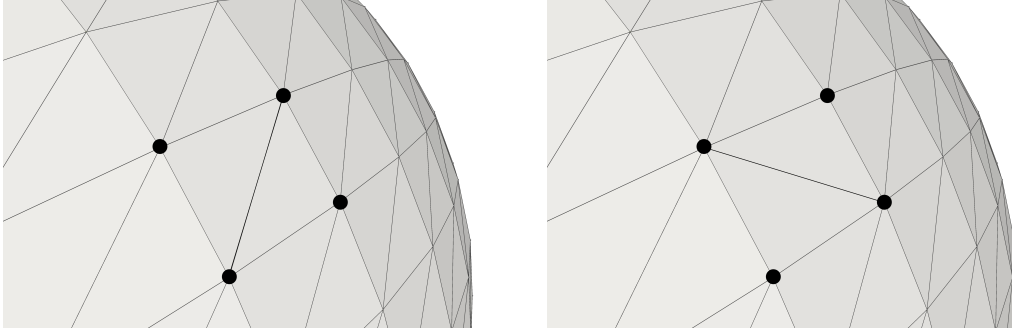
Figure 5: Local changes in the triangulation choice - such as edge flips, do not affect the overall connectivity of the mesh. The four highlighted vertices create the same connected surface in both figures.
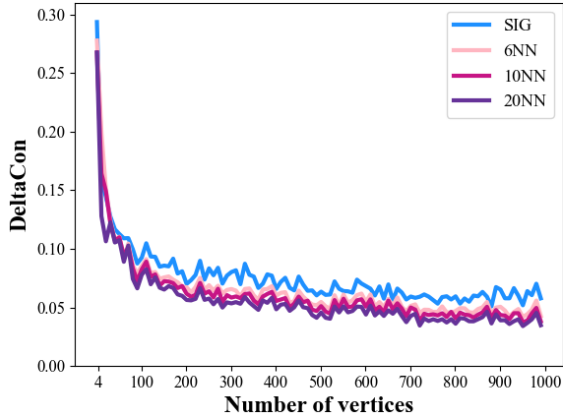

Figure 6: DeltaCon graph similarity metric - a higher value corresponds to a closer similarity to the original graph.
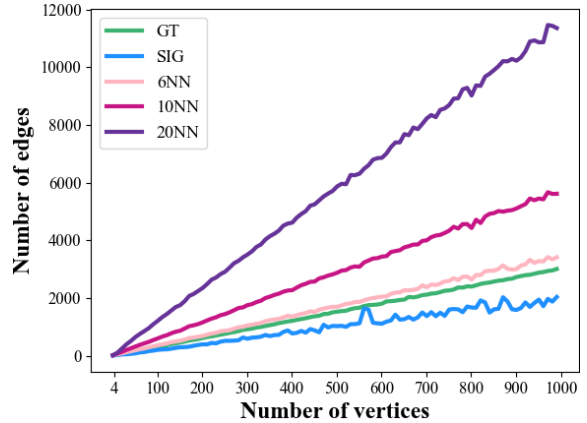

Figure 7: Number of edges in graphs plotted against the number of vertices. We consistently obtain the smallest number of edges by a large margin, while still correctly preserving the original connectivity.

Storage. Low storage represents another requirement for our method, since we try to use the minimum amount of edges that preserve the connectivity by making use of the spatial proximity properties of SIG. The total number of edges for each graph can be seen in Figure 7, where the results are presented with respect to the number of vertices. Our method achieves the lowest number of edges and hence, has the lowest storage requirement. Our number of edges is lower than the ground truth since the ground truth number of edges is extracted from triangulated meshes, which contain some redundant edges with respect to connectivity.

Surface Approximation: Geodesic. We are also comparing the distance between pairs of nodes in our graph to the geodesic distance over the ground truth surface, computed using the Heat Method (Crane et al., 2017). This way, we measure how close the graph edges follow the surface. For all pairs of nodes in the original mesh, we compute the ratio between the geodesic distance over the original mesh and the shortest distance between the same nodes in SIG and between the same nodes in the kNN graphs. The re-

sults are shown in Figure 8, where we present the ratio of distances in relation to the number of vertices in the input. The differences between buckets are due to the varying number of input datasets in each bucket, as well as the type of meshes - a single mesh with close sheets and sparse sampling, such as the one presented in Figure 9 highly increases the error for the connectivity graphs. We aim to obtain a resulting ratio of 1, as values lower than 1 indicate longer paths in the proximity graphs, while values higher than 1 would imply the existence of shortcuts (too many edges) in the connectivity graphs. Our method is consistently close to the desired ratio of 1 across the tested meshes, without the need to tune any parameters. Even if for some of the tested input ranges some of the kNN graphs have better results, these are not universal and the user would need to adjust k depending on the usage, which is an issue solved by our method.

Application: Normal Estimation. We propose SIG as a connectivity graph that encodes surface properties well, and can act as an alternative for the commonly used kNN graphs. One usual application is
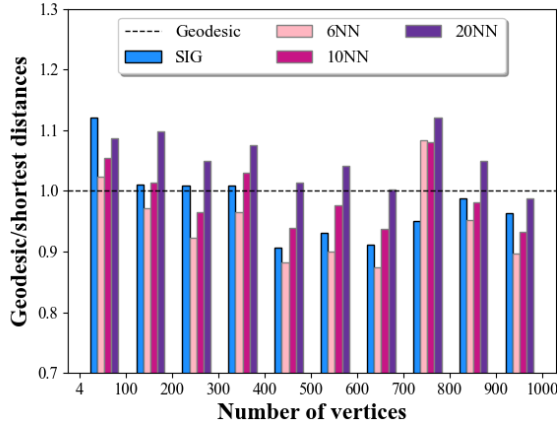
Figure 8: Ratio of the geodesic distance traced on the surface to the shortest path in the computed graph, ideally 1.
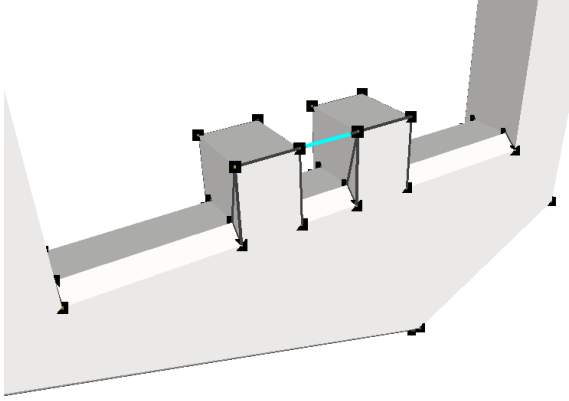

Figure 9: Sparse sampling of close sheets (the parallel tower-like structures in the center) generates edges across the surface. These skew our measurement of the geodesic ratio to the shortest distance, as the surface originally follows the U-structure, while our graph shortcuts it through the edge connecting the two towers, highlighted in blue. However, similar behavior is exhibited by the kNN graphs, since all of them are distance-based.

normal computation for unstructured point clouds using PCA. Even though more advanced normal computation methods have been developed, constructing them using PCA on a connectivity graph still represents a widely used method and good results in this direction indicate an overall good representation of the underlying surface. Moreover, we are not aiming to improve the normal computation in general, but to show that our graph can be used in similar applications as kNN.

For each vertex, we computed the covariance matrix using its neighbors and extracted the normal as the normalized eigenvector corresponding to the smallest eigenvalue. We do not consistently orient the normals, as this can be done in a post-processing step and we are only interested in the angle difference
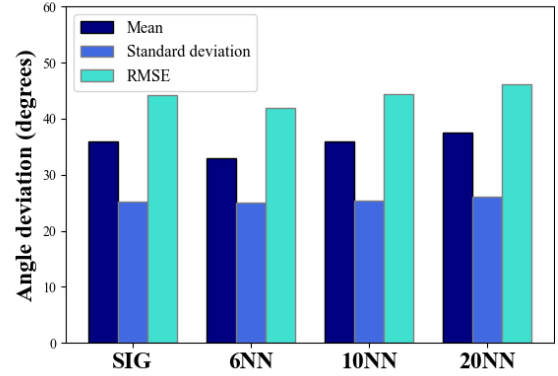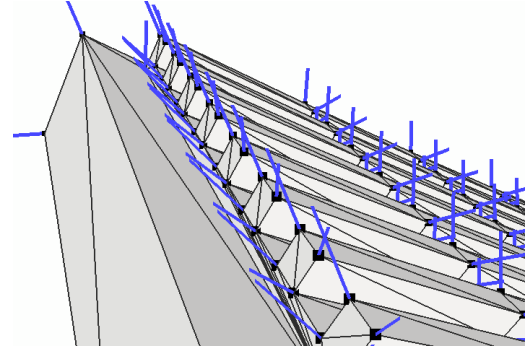

Figure 10: Angle variation between normals computed using PCA over connectivity graphs and original, face-based normals. We compute the mean and the standard deviation of the angle difference, and the root mean square error. All of the methods achieve similar deviations. The overall error is high due to the sharp angles in the input dataset.
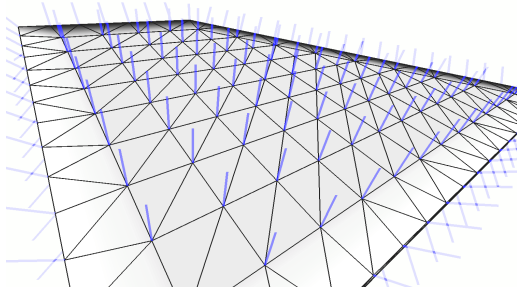

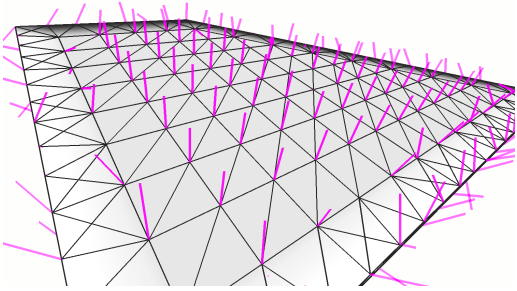(a) Ground-truth normals computed using the incident faces of each vertex.
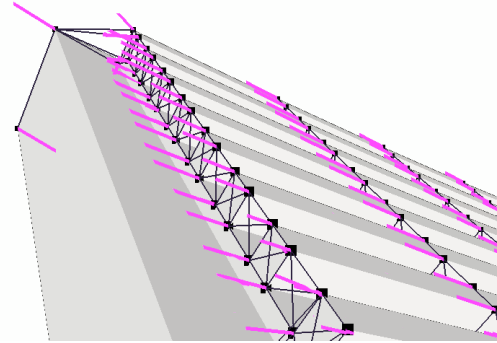
(b) Normals computed with PCA using the SIG connectivity. The sparse sampling creates parallel layers and normals are oriented accordingly since the top vertices do not get connected across layers. However, this is an issue encountered by distance-based methods in general.

Figure 11: Example of how CAD-like models with sparse sampling affect the computed connectivity, and hence, the normal computation.
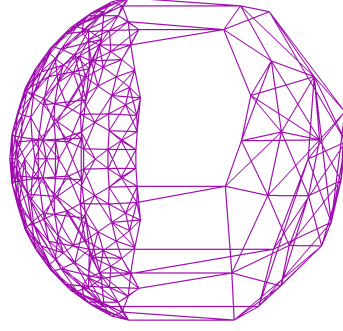
(a) Ground-truth normals computed using the incident faces of each vertex.

(b) PCA normals computed using SIG are very close to the ground truth. Normals are computed for every vertex, but since we do not orient them consistently, some of them are facing the other way and are not visible. Note that for our evaluation, orientation does not matter, and consistency is usually achieved with a post-processing step.

Figure 12: Improved normal computation of our method for more uniformly sampled meshes.

(a) 6NN creates bridges between the two hemispheres, since the graph does not consider local densities. Increasing the k value only aggravates the issue, as more bridge edges will be constructed.
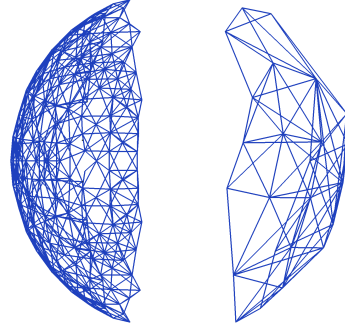
(b) SIG only creates edges on each hemisphere, without crossing the gap between the two surface sheets.

Figure 13: Different sampling densities on two surface sheets that are close together - the two hemispheres are not connected as ground truth.

when compared to the ground truth, which can be computed without the consistent orientation. For the original meshes, we used the triangulated faces to compute the normals. We do not use the ground truth edge graph since that would bias the normal computation in the direction of a specific triangulation. Instead, face-based normal computation takes into account more information about the surface, and not only a specific 1-ring. Then, we computed the average angle deviation for SIG and the kNN graphs when compared to the ground truth normals. Results are presented in Figure 10, where the difference among the various tested graphs is less than 1 degree for all the metrics. Since the chosen dataset contains surfaces that are sparsely sampled, the normal computation achieved high errors for all graphs. Thus, for this metric, we resampled the chosen dataset (adding new vertices along edges longer than a specified threshold and retriangulating), obtaining models with up to 7k vertices. An example of how sparse sampling, which is also an issue in LIDAR scans, can cause problems, is presented in

Figure 11, where there is not enough information for the vertices placed on edges to have their normal computed correctly. However, for well-sampled models, the normals are close to the ground truth - Figure 12. The overall angle deviation is still high for all methods, since some of the meshes exhibit sharp angles, for which the normal computation is also erroneous for all graphs.

Timings. Due to the proved bounded radius in which SIG neighbors are found, our method's timings are comparable to kNN, as can be observed in Figure 14. We observe a linear increase in the computational time with the number of vertices, which is expected due to the linear nature of our algorithm. Our method is only slightly slower than 6NN, but achieves better results overall and manages to do so with many fewer edges.

All experiments have been performed using an AMD Ryzen 7 5800 processor. Both kNN and SIG graphs have been implemented using the
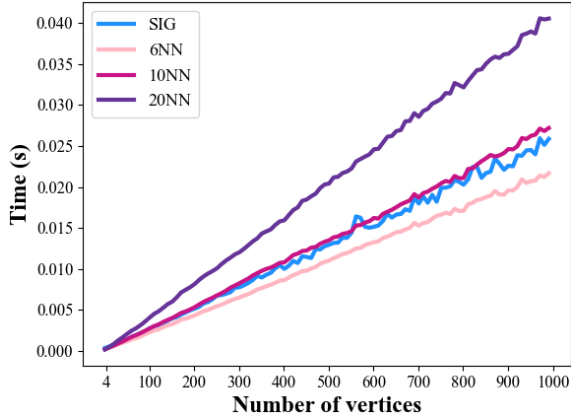
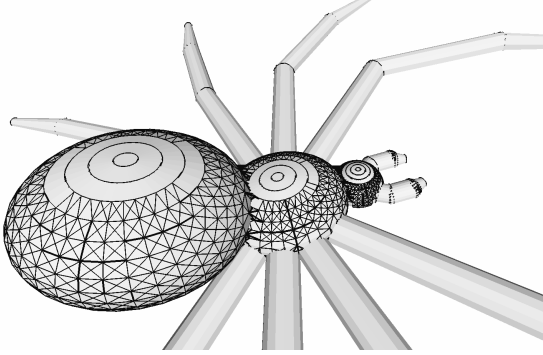Figure 14: Timings of our method compared to *k*NN graphs for inputs with various numbers of vertices.

Figure 15: Parallel layers of sampling result in disconnected ring-like structures as can be observed on the spider's body and legs. However, *k*NN graphs exhibit similar issues if the sampling is too sparse and nodes are clustered in layers.

`scipy.spatial.KDTree` in `python`.

Limitations. Since our method depends on distances, non-uniform sampling may negatively affect the result, as is commonly the case with connectivity reconstruction. An example can be seen in Figure 15, where the spider's legs and top of the body are uniformly sampled along parallel layers. For real-world point clouds, LIDAR scans can produce such artifacts. These configurations are difficult to handle for all methods, as they may fail to connect subsets of the point cloud. Such issues could be mitigated by employing an incremental SIG - using the next nearest neighbors until a specific vertex degree or average neighbor angle has been reached for the current graph.

However, in the case of different sampling densities on distinct parts of the same object, our method has an advantage over *k*NN. Our method is less likely to create edges between surface sheets that are geometrically close, but geodesically distant (Figure 13).

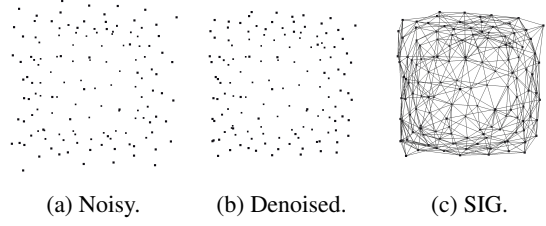(a) Noisy.   (b) Denoised.   (c) SIG.

Figure 16: Denoising pipeline that would allow our method to give good results on noisy point clouds. The noisy point cloud of a cube is denoised in 16b using the Bilateral Filter (Digne and de Franchis, 2017), and then used as input for our SIG computation, resulting in a graph that closely approximates the original cube.

Our method is not robust to noise by design, as it computes a distance-based neighborhood, exhibiting similar drawbacks as *k*NN graphs. Noisy point clouds could still be used with our method, provided they have been cleaned in a pre-processing step using, for example, PointCleanNet (Rakotosaona et al., 2020), the Bilateral Filter (Digne and de Franchis, 2017) or Score-Based Denoising (Luo and Hu, 2021), as demonstrated in Figure 16.

5 CONCLUSIONS

We present an alternative to the commonly used *k*NN graphs for establishing the connectivity of point clouds: the SIG, a parameter-free proximity graph. In our work, we introduce novel spatial constraints for the extent of SIG edges, leveraging this new property to enhance its computational efficiency. We demonstrate the SIG's improved connectivity representation that is parameter-free. Moreover, as a sparse graph, it has a lower edge count, thus minimizing storage requirements. Consequently, it offers three key advantages over *k*NN: no need for parameter tuning, sparsity, and improved connectivity encoding. As an incidental application, we have shown that computed normals are of competitive quality to *k*NN.

Future Work. We aim to utilize the advantage of parameter-free improved connectivity for surface reconstruction. Moreover, we are planning to use the SIG neighborhood (with possible extensions to n^{th} nearest neighbor) in graph-convolutional networks for learning from 3D point cloud data. We also plan to investigate how our method can create an advantage in other fields, such as motion planning and simulations.

ACKNOWLEDGMENTS

REFERENCES

Amenta, N. and Bern, M. (1999). Surface reconstruction by voronoi filtering. *Discr. & Comp. Geom.*, 22:481–504.

Boulch, A. and Marlet, R. (2016). Deep learning for robust normal estimation in unstructured point clouds. *Computer Graphics Forum*, 35(5):281–290.

Charles, R. Q., Su, H., Kaichun, M., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE CVPR*, pages 77–85.

Crane, K., Weischedel, C., and Wardetzky, M. (2017). The heat method for distance computation. *Commun. ACM*, 60(11):90–99.

de Figueiredo, L. H. and Paiva, A. (2022). Region reconstruction with the sphere-of-influence diagram. *Computers & Graphics*, 107:252–263.

Dey, T. K. and Goswami, S. (2006). Provable surface reconstruction from noisy samples. *Computational Geometry*, 35(1):124–141. Special Issue on the 20th ACM Symposium on Computational Geometry.

Digne, J. and de Franchis, C. (2017). The bilateral filter for point clouds. *Image Processing On Line*, 7:278–287.

Dwyer, R. A. (1995). The expected size of the sphere-of-influence graph. *Comp. Geometry*, 5(3):155–164.

Edelsbrunner, H., Kirkpatrick, D., and Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559.

Erler, P., Guerrero, P., Ohrhallinger, S., Mitra, N. J., and Wimmer, M. (2020). Points2surf learning implicit surfaces from point clouds. In *Computer Vision–ECCV 2020, Proceedings, Part V*, pages 108–124. Springer.

Guennebaud, G. and Gross, M. (2007). Algebraic point set surfaces. *ACM Trans. Graph.*, 26(3):23–es.

Guerrero, P., Kleiman, Y., Ovsjanikov, M., and Mitra, N. J. (2018). Pcpnet learning local shape properties from raw point clouds. *Computer Graphics Forum*, 37(2):75–85.

Jaromczyk, J. and Toussaint, G. (1992a). Relative neighborhood graphs and their relatives. *Proceedings of the IEEE*, 80(9):1502–1517.

Jaromczyk, J. W. and Toussaint, G. T. (1992b). Relative neighborhood graphs and their relatives. *Proc. IEEE*, 80:1502–1517.

Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson Surface Reconstruction. In *Symposium on Geometry Processing*. The Eurographics Association.

Kirkpatrick, D. G. and Radke, J. D. (1985). A framework for computational morphology. *Machine Intelligence and Pattern Recognition*, 2:217–248.

Klein, J. and Zachmann, G. (2004). Point cloud surfaces using geometric proximity graphs. *Computers & Graphics*, 28(6):839–850.

Koutra, D., Shah, N., Vogelstein, J. T., Gallagher, B., and Faloutsos, C. (2016). Deltacon: Principled massive-graph similarity function with attribution. *ACM Trans. Knowl. Discov. Data*, 10(3).

Liu, Z., Xiao, X., Zhong, S., Wang, W., Li, Y., Zhang, L., and Xie, Z. (2020). A feature-preserving framework for point cloud denoising. *Computer-Aided Design*, 127:102857.

Luo, S. and Hu, W. (2021). Score-based point cloud denoising. In *2021 ICCV*, pages 4563–4572.

Marin, D., Ohrhallinger, S., and Wimmer, M. (2022). Sigdt: 2d curve reconstruction. *CGF*, 41(7):25–36.

Michael, T. and Quint, T. (1999). Sphere of influence graphs in general metric spaces. *Mathematical and Computer Modelling*, 29(7):45–53.

Mitra, N. J. and Nguyen, A. (2003). Estimating surface normals in noisy point cloud data. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, page 322–328, New York, NY, USA. Association for Computing Machinery.

Pauly, M., Keiser, R., Kobbelt, L., and Gross, M. (2003). Shape modeling with point-sampled geometry. *ACM SIGGRAPH 2003 Papers, SIGGRAPH '03*.

Rakotosaona, M.-J., La Barbera, V., Guerrero, P., Mitra, N. J., and Ovsjanikov, M. (2020). Pointcleannet: Learning to denoise and remove outliers from dense point clouds. *Comp. Graph. Forum*, 39(1):185–203.

Shi, W. and Rajkumar, R. (2020). Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proc. of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719.

Toussaint, G. T. (1988). A graph-theoretical primal sketch. In *Machine Intelligence and Pattern Recognition*, volume 6, pages 229–260. Elsevier.

Wang, L., Huang, Y., Hou, Y., Zhang, S., and Shan, J. (2019). Graph attention convolution for point cloud semantic segmentation. In *2019 CVPR*, pages 10288–10297.

Zhou, Q. and Jacobson, A. (2016). Thingi10k: A dataset of 10,000 3d-printing models. *arXiv:1605.04797*.