Membrane and microtubule rapid instance segmentation with dimensionless instance segmentation by learning graph representations of point clouds

Robert K.
Simons Machine Learning Center
New York Structural Biology Center
New York, NY 10027

Biocomputing Unit
Centro Nacional de Biotecnologia
Madrid, 28049
`rkieiwsz@nysbc.org`

Tristan B.
Simons Machine Learning Center
New York Structural Biology Center
New York, NY 10027
`tbepler@nysbc.org`

Abstract

Point clouds are an increasingly common spatial data modality, being produced by sensors used in robotics and self-driving cars, and as natural intermediate representations of objects in microscopy and other bioimaging domains (e.g., cell locations over time, or filaments, membranes, or organelle boundaries in cryo-electron micrographs or tomograms). However, semantic and instance segmentation of this data remains challenging due to the complex nature of objects in point clouds. Especially in bioimaging domains where objects are often large and can be intersecting or overlapping. Furthermore, methods for operating on point clouds should not be sensitive to the specific orientation or translation of the point cloud, which is often arbitrary. Here, we frame the instance segmentation problem as a graph learning problem in which we seek to learn a function that accepts the point cloud as input and outputs a probability distribution over neighbor graphs in which connected components of the graph correspond to individual instances. We introduce the Dimensionless Instance Segmentation Transformer (DIST), a deep neural network for spatially invariant instance segmentation of point clouds to solve the point cloud-to-graph problem. DIST uses an SO(n) invariant transformer layer architecture to operate on point clouds of arbitrary dimension and outputs, for each pair of points, the probability that an edge exists between them in the instance graph. We then decode the most likely set of instances using a graph cut. We demonstrate that the DIST has the power to segment biomolecules in cryo-electron micrographs and tomograms, far surpassing existing methods for membrane and filament segmentation in our evaluation. We anticipate that DIST will underpin a new generation of methods for point cloud segmentation and that our general model and approach will provide useful insights for point cloud segmentation methods in other domains.

1 Introduction

Point clouds are a common way to represent objects or scenes on a computer, and are widely used in computer vision, augmented and virtual reality, and imaging. Point clouds of scenes are often

subsequently processed to semantically classify the points - semantic segmentation - or to segment individual objects and instances - instance segmentation (**Figure 1**). Unlike 2D or 3D images, point clouds are disordered, unstructured, and may have noisy point locations, making it difficult to design algorithms or machine learning models to process them. Deep learning methods for processing point clouds have become of increasing interest with the increase in point cloud data being generated by sensors in robotics and as a representation of objects in physics engines, natural images, and bioimaging. Many recent methods have been developed to segment point clouds using deep learning [8, 11, 15, 17, 2, 3, 10, 16], which address the instance, scene, or part segmentation problems using various architectures or training schemes. However, instance segmentation methods require prior information about the number of instances that are present or assume some fixed number of instances. Furthermore, many methods convert point clouds into pixel- or voxel-grids to process them with convolutional layers, or otherwise incorporate point coordinates directly into the network, causing their outputs not to be invariant to rotation and translation of the point cloud.
In cryo-electron microscopy, an increasingly common task is to segment individual filaments, membranes, organelles, or other biological structures in 2D micrographs or 3D tomograms. These objects are often large and can intersect or overlap causing instance segmentation to be complicated even if a semantic segmentation mask is known. Experienced scientists or technicians often spend weeks to months painstakingly manually labeling these datasets for downstream analysis, limiting throughput. Current state-of-the-art methods barely help. For filament instance segmentation, for example, these methods utilize algorithms custom-tailored to curve tracing [1] but have such high error rates that scientists still spend days manually correcting annotation errors, if the algorithms work at all [12, 13]. Faster and more accurate instance segmentation methods are urgently needed to facilitate large-scale analysis of these datasets as imaging technology improves.
To address these problems, we propose the Dimensionless Instance Segmentation Transformer (DIST). DIST is able to perform SO(n) invariant instance segmentation of point clouds using only geometric features. We accomplish this by framing instance segmentation as a graph prediction problem. Given a point cloud as input, DIST outputs a probability distribution over graphs parameterized by the probability. Instances, therefore, are defined by connected components of the full point cloud graph. This allows us to perform instance segmentation without any restrictions on the maximum number of instances. We find that DIST performs incredibly well, improving the state of the art in membrane instance segmentation in 2D micrographs and 3D microtubule (MT) segmentation in 3D tomograms by a large margin (from 0.539 mCov for Amira to 0.955 mCov with DIST).

2 Method

Instance segmentation with DIST is divided into three parts (**Figure 2**): **1)** the raw point cloud is converted into a pairwise representation based on distances between the points, **2)** the edge representations are fed through the DIST model to produce an output matrix containing the probability. In the way that for each pair of points, output predicts an edge between them in the instance graph. **3)** the maximum likelihood graph is found using a graph cut algorithm to predict instance segmentation. The DIST model is trained to predict the ground truth edges in labeled instance graphs using standard neural network training techniques, back-propagation and stochastic gradient descent.

Pairwise-feature input embedding. The DIST model operates on pairwise features. That is, for each pair of points in the point cloud, DIST learns a vector representation. We refer to these as pairwise embeddings or edge embeddings. In order to ensure that our representations are invariant to the translation and rotation of the point cloud, we initialize the edge features using the corresponding distance between each pair of nodes. Because we think that nearby points are more important than distant points within the point cloud, we define these features using a scaled exponential of the negative squared distance. More formally, given a point cloud represented as a set of points p_i where *i = 1, . . . , n*, we initialize a graph *G* with a featureless node for each p_i, and edges $e_{i,j}$ connecting all nodes, with weights $e_{i,j} = exp(\frac{-d_{i,j}^2}{s^2 * 2})$, where $d_{i,j}$ is the euclidean distance between p_i and p_j, and s is sigma denoted as fixed scaling factor for the normalized point cloud. This weighting approach allows us to embed geometrical information about all points directly from the Euclidean distance. Weights are then multiplied with a learned d-dimensional embedding vector to define the initial edge representations.

Geometric transformer layer. Starting from the initial edge representations, we apply several steps of axial multi-head attention (MHA) and triangular multiplicative updates (**Appendix Figure A-B**).

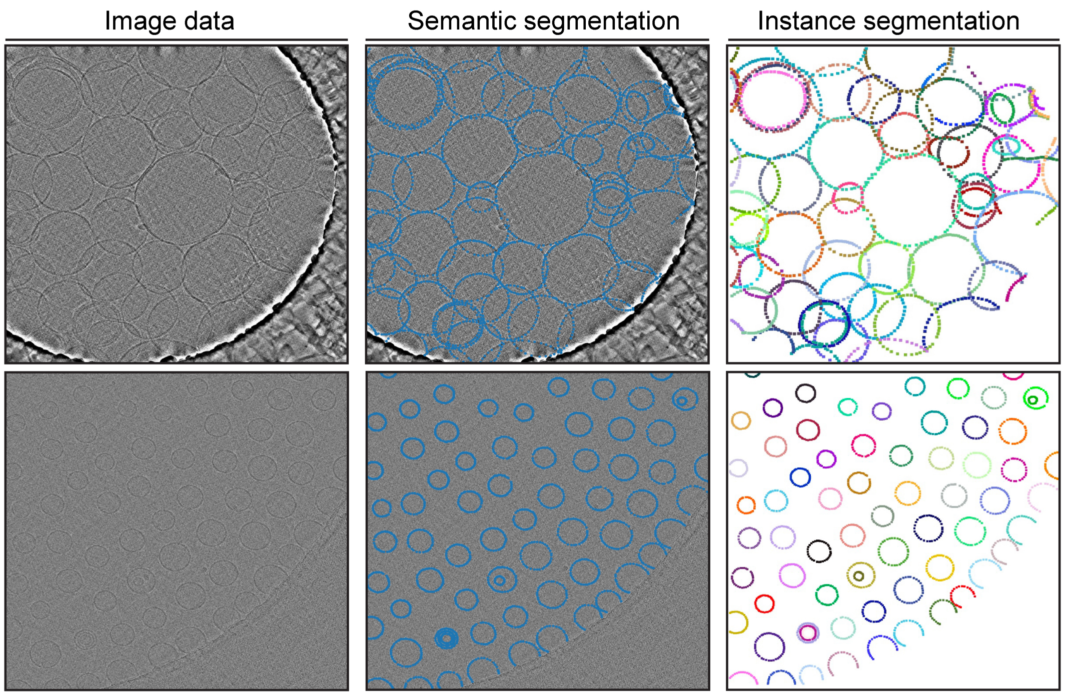


Figure 1: Examples of semantic and instance segmentation in 2D on a cryo-EM micrograph. Semantic segmentation classifies points into semantic classes. Instance segmentation classifies points into individual instances, which may be of the same semantic class.

Each overall step is referred to as a "DIST layer" (**Appendix Figure A-C**), and the edge update that occurs within each DIST layer (composed of simultaneous MHA and triangular multiplicative updates) is referred to as the "Edge update" module (**Appendix Figure A-D**). In the appendix description **Appendix B**, we describe actions on rows/columns of the graph representation of G, which corresponds to the set of incoming or outgoing edges of G.

At each DIST layer, the output from both the MHA and triangular multiplicative updates are added to the incoming edge features, and the resulting updated edge features are fed through an activation function before being used as the input to the next DIST layer. Finally, after the last DIST layer, the resulting edges in the graph are fed through a single linear layer to obtain a probability for that edge.

Graph cut instance segmentation. The final DIST graph representation contains the predicted probability for each edge in the instance graph of the point cloud, which can be used to obtain an instance segmentation. A simple segmentation can be achieved by thresholding the probabilities and examining the resulting graph. We explain the detail of the graph instance segmentation in the **Appendix D**.

3 Experiments

Datasets. Because our primary motivation is to perform instance segmentation on biological structures in electron microscopy (EM) data, we compare the performance of DIST to the current state-of-the-art algorithms for instance segmentation of filament-like structures on real-world EM data. We evaluate DIST on a membrane segmentation task in 2D cryo-EM micrographs and a microtubule segmentation task in 3D plastic section tomograms. For both datasets, an experienced microscopist derives ground truth labels from manual annotation. The datasets contained of 76 micrographs of membranes (2D scenes; with 500-2’500 points each) and 48 MT tomograms (3D scenes; with 10’000-50’000 points each). We split these into a train, a validation, and a test sets with an 80/10/10 split ratio. The detail explaining point cloud pre-processing are presented in the **Appendix C** as well as how we generated ground truth from graph representation in the **Appendix E**.
In all experiments, we use a DIST model with 6 layers, 8 attention heads, and a hidden dimension of

128. We lightly tuned the value of sigma, finding that both biological datasets a value of 2 gave the best results on the training set. All DIST models were trained with binary cross entropy loss using the ADAM optimizer with learning rate 10^{-5} and no weight decay on a single NVIDIA A100 GPU. DIST model training was stopped when the validation loss did not improve after 50 epochs.

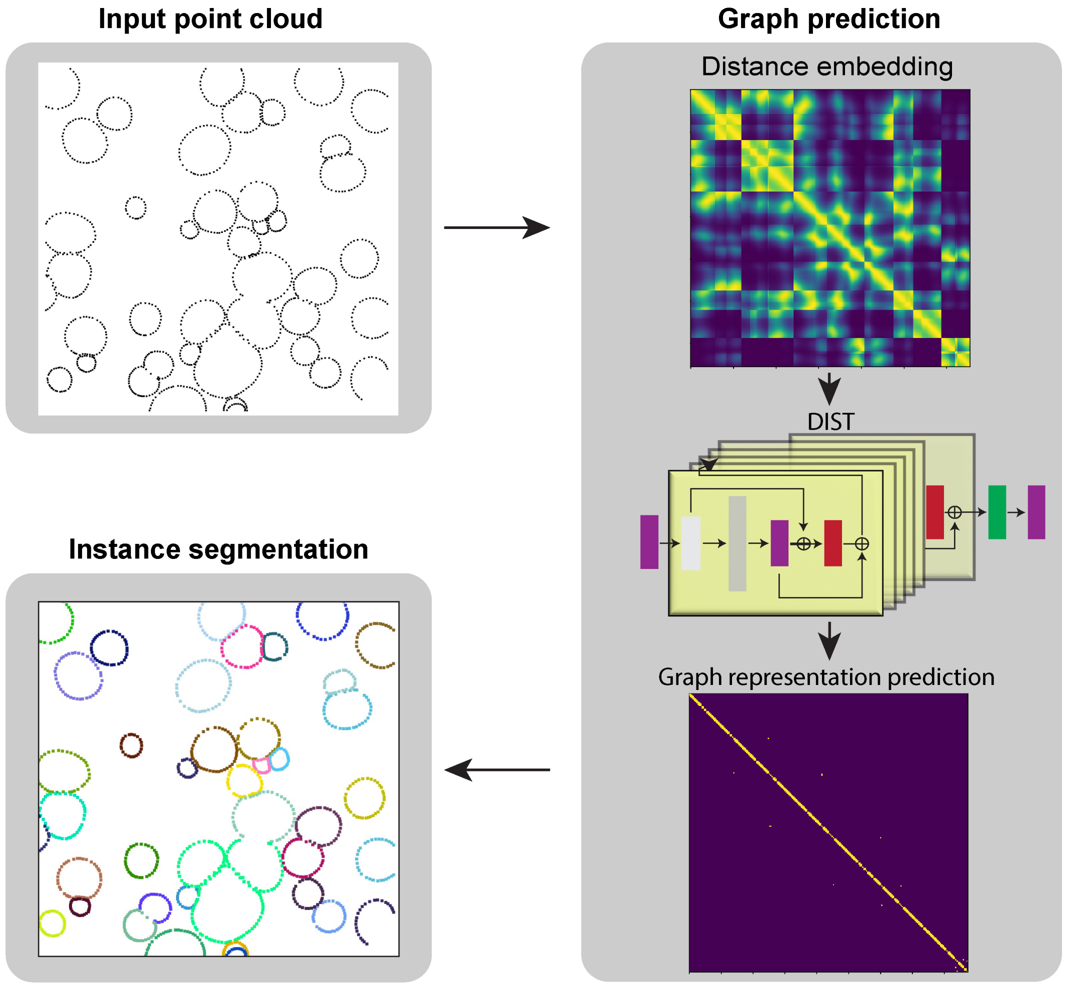


Figure 2: Instance segmentation with DIST. The input pairwise feature representations are defined from the Euclidean distance between points in the input point cloud. The DIST model learns to refine these representations and then outputs edge probabilities. These edge probabilities are used to find the connected components which define individual instances.

Graph representation evaluation. We performed experiments on point clouds manually segmented by experienced users. The evaluation metric used for the graph prediction is intersection over union (mIoU) averaged over all cropped graphs. Due to the lack of a comparable real or synthetic benchmark dataset to which we could compare our DIST graph prediction, we defined here our baseline independently. The baseline was defined as a point cloud graph representation generated from point distances. This was achieved by computing the distance between each pair of nodes in a point cloud.

Instance segmentation evaluation. To benchmark instance segmentation performance we measured mean class coverage (mCov; [5]).This metric measures IoU between the ground truth label and its matching predictions. Additionally, for these datasets we also generate our baseline using state-of-the-art segmentation software ZiB Amira [13] and multi-curve fitting (MCF; [1]). Both of this software are nowadays wildly used for MT segmentation tasks. For both of the methods, we used the 'standard' setting. In the case of ZiB Amira, the setting was heavily tuned for the MT dataset and is suggested to use by the authors. In the case of MCF, the author optimized the 'standard' setting for filament-like structures. MT and Mem datasets are filament-like structures, and therefore using a setting tuned by the author and adjusting only the pixel size value in our opinion should yield the best results.

Benchmark results. The metrics of the point clouds was shown in **Table 1**. For the inference of the graph representation, the output of the DIST was thresholds before calculating mIoU. The threshold was selected by manually picking the best value based on a sub-sample of 10 datasets that were not used for the evaluation. The proposed DIST achieved a great leap in performance comparing mIoU of 30% and 113% compared to our baseline **Appendix Figure F**.

	mIoU		AUPR	
Model	Mem	MTs	Mem	MTs
DIST (our)	**0.934**	**0.916**	**0.967**	**0.994**
Baseline (DC - point distances)	0.718	0.430	0.861	0.923

Table 1: Graph prediction comparison biological and synthetic datasets.

The evaluation of the graph instance inference model is shown in **Table 2**. Due to the uniqueness of the EM data, we also compared our DIST framework with the current state-of-the-art method used for membrane and MT automatic segmentation. On the 2D dataset, we compared DIST to two methods: MCF [1] and distance clustering (DC). MCF was recently demonstrated as a workflow for instance segmentation based on iterative fitting of points to spline based on multiple hyperparameters that must be tuned for each dataset. The DC method on the other hand relies on building instances by clustering points based on *k*NN distances. We compared our 3D MT dataset to MCF and currently used Amira software which produces state-of-the-art performance on MT segmentation tasks [13]. DIST achieved the best results for both membrane and MT segmentation. **Appendix Figure I** shows further segmentation examples provided by DIST, DC, MCF and Amira. Moreover, MT data in comparison to current state-of-the-art software, Amira, shows outstanding results (**Table 2** and **Appendix Figure I**).

Method	mCov	
	Mem	MTs
DIST (our)	**0.913**	**0.955**
DC	0.253	0.214
MCF [1]	0.135	0.000
Amira [13]	-	0.539

Table 2: Instance segmentation comparison for the membrane and MT dataset.

4 Conclusion

We propose DIST, a neural network architecture for instance segmentation of point clouds. By using geometric features to learn SO(n) invariant graph representations, we are able to achieve state-of-the-art instance segmentation of biological structures in electron micrographs and tomograms. By framing the instance segmentation problem as a graph prediction problem where the instances are defined by connected sub-graphs, we are able to identify any number of instances without any built-in constraints in our network. Furthermore, we show that using geometrically inspired updates, the triangular update module, was critical for model performance and that this was enhanced by the inclusion of axial attention modules.

We expect that this approach to instance segmentation will transform our ability to understand biological structures in the increasing amounts of structural data being generated by electron microscopy, where biomedical researchers are in need of fast and accurate instance segmentation methods. In the future, we expect that DIST will underpin filament, membrane, and organelle segmentation software. Furthermore, DIST can be applied to other point cloud instance segmentation problems and extends easily to arbitrary dimension point clouds. This would enable it to be applied to object tracking over time in 3D imaging, for example, which can be represented as a 4D space. DIST can also be extended to include node features, allowing additional semantic information for each point to be passed to the network, further increasing performance.

References

P. Chai, Q. Rao, and K. Zhang. Multi-curve fitting and tubulin-lattice signal removal for structure determination of large microtubule-based motors. *BioRxiv*, 2022. doi: https://doi.org/10.1101/2022.01.22.477366. URL https://doi.org/10.1101/2022.01.22.477366.

M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu. PCT: Point cloud transformer. *Computational Visual Media*, 12 2020. doi: 10.1007/s41095-021-0229-5. URL http://arxiv.org/abs/2012.09688http://dx.doi.org/10.1007/s41095-021-0229-5.

J. Hong and T. P. Pavlic. KCNet: An Insect-Inspired Single-Hidden-Layer Neural Network with Randomized Binary Weights for Prediction and Classification Tasks. *NeruIPS*, 8 2021. URL http://arxiv.org/abs/2108.07554.

L. Hui, M. Cheng, J. Xie, and J. Yang. Efficient 3D Point Cloud Feature Learning for Large-Scale Place Recognition. *thecvf*, 1 2021. doi: 10.1109/TIP.2021.3136714. URL http://arxiv.org/abs/2101.02374http://dx.doi.org/10.1109/TIP.2021.3136714.

H. Jiang, F. Yan, J. Cai, J. Zheng, and J. Xiao. End-to-End 3D Point Cloud Instance Segmentation Without Detection. In *CVPR*, pages 12793–12802. IEEE, 6 2020. ISBN 978-1-7281-7168-5. doi: 10.1109/CVPR42600.2020.01281.

J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen, D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 8 2021. ISSN 14764687. doi: 10.1038/s41586-021-03819-2.

K. Kyzirakos, F. Alvanaki, and M. Kersten. In memory processing of massive point clouds for multi-core systems. In *DaMoN '16*, pages 1–10, New York, New York, USA, 2016. ACM Press. ISBN 9781450343190. doi: 10.1145/2933349.2933356.

X. Lai, J. Liu, L. Jiang, L. Wang, H. Zhao, S. Liu, X. Qi, and J. Jia. Stratified Transformer for 3D Point Cloud Segmentation. *CVPR*, 3 2022. URL http://arxiv.org/abs/2203.14508.

Z. Liu, H. Tang, Y. Lin, and S. Han. Point-Voxel CNN for Efficient 3D Deep Learning. *NeurIPS*, 7 2019. URL http://arxiv.org/abs/1907.03739.

G. Pan, J. Wang, R. Ying, and P. Liu. 3DTI-Net: Learn Inner Transform Invariant 3D Geometry Features using Dynamic GCN. *arxiv*, 12 2018. URL http://arxiv.org/abs/1812.06254.

C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *NeurIPS*, 6 2017. URL http://arxiv.org/abs/1706.02413.

S. Redemann, B. Weber, M. Möller, J.-M. Verbavatz, A. A. Hyman, D. Baum, S. Prohaska, and T. Müller-Reichert. The Segmentation of Microtubules in Electron Tomograms Using Amira. In D. J. Sharp, editor, *Mitosis: Methods and Protocols, Methods in Molecular Biology*, volume 1136, chapter 12, pages 261–278. Springer Science+ Business Media, New York, 1136 edition, 2014. ISBN 9780080961569. doi: 10.1007/978-1-4939-0329-0{_}12. URL http://link.springer.com/10.1007/978-1-4939-0329-0_12.

D. Stalling, M. Westerhoff, and H.-C. Hege. Amira: a Highly Interactive System for Visual Data Analysis. In C. D. Hansen and C. R. Johnson, editors, *The Visualization Handbook*, chapter PART IX: V, pages 749–767. Elsevier, i edition, 2005. ISBN 978-0-12-387582-2.

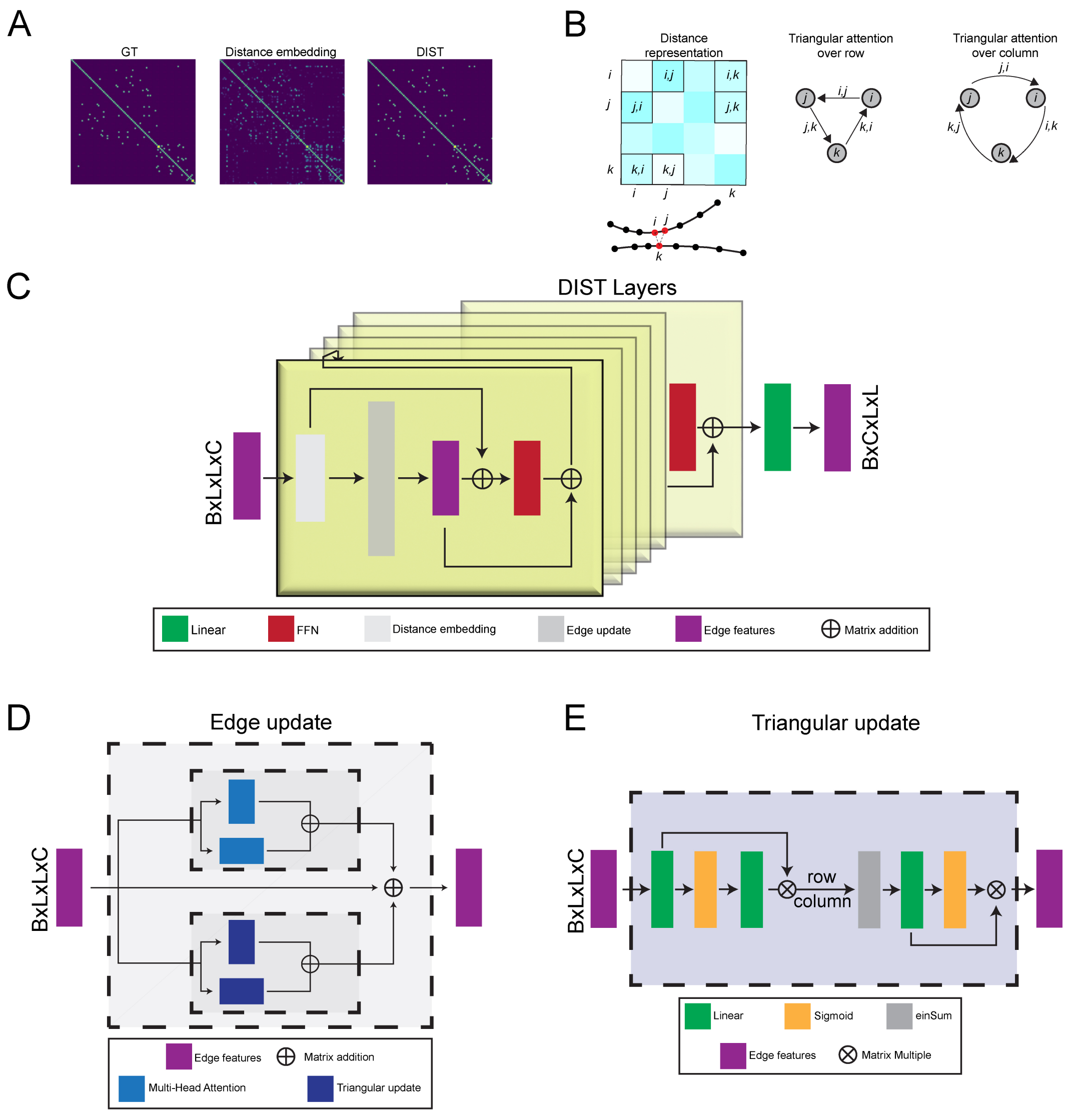
A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. *NeurIPS*, 6 2017. URL http://arxiv.org/abs/1706.03762.

D. Wang. Unsupervised semantic and instance segmentation of forest point clouds. *ISPRS*, 165: 86–97, 7 2020. ISSN 09242716. doi: 10.1016/j.isprsjprs.2020.04.020.

W. Yuan. Point Cloud Semantic Segmentation using Graph Convolutional Network. *thecvf*, 2021. URL `https://github.com/TonythePlaneswalker/pcd_seg`.

F. G. Zanjani, A. Pourtaherian, S. Zinger, D. A. Moin, F. Claessen, T. Cherici, S. Parinussa, and P. H. de With. Mask-MCNet: Tooth instance segmentation in 3D point clouds of intra-oral scans. *Neurocomputing*, 453:286–298, 9 2021. ISSN 18728286. doi: 10.1016/j.neucom.2020.06.145.

A Detail DIST design



Appendix Figure 1: Example of instance segmentation using DIST.
(**A**) Illustration of 2D cryo-EM micrograph with segmented membranes. Instance segmentation was compared with MCF and DC methods. (**B**) Electron tomography reconstruction of mammalian cells with segmented MTs in 3D. Instance segmentation was compared with MCF and Amira methods.

B Detail implementation of DIST layer

MHA module was adapted from the original paper (Vaswani et al. [2017]). It performs axial attention updates over rows or columns of the graph. In axial updates, we switch between interpreting the rows or columns as the long axis of the tensor and applying typical multi-head attention. This corresponds to attending over edges outgoing from a node (row attention) or incoming to a node (column attention). This is followed by a fully connected layer with residual connections from the row and column axial attention operations to generate the output edge features.

Triangular multiplicative update was adapted from (Jumper et al. [2021]), with a detailed design depicted in **Appendix figure A-E** and pseudo-code shown in **Appendix G**. The triangular update takes an input edge feature embedding and performs Einstein summation over row or column features, followed by a linear layer with gating to obtain an update for the edge feature embedding.

C Datasets Implementation Details

Point Cloud Pre-Processing. There are two major challenges in point cloud instance segmentation: **1)** uneven point spacing within and between datasets, which makes learning generalizable local features difficult, and **2)** the size of the point cloud, which is often too large to process due to memory constraints, because the RAM usage of DIST scales cubically (axial attention requires quadratic attention for each point) with the number of points, making large point clouds require far more RAM than is available on current GPUs.

Point cloud re-scaling. We tackle the uneven sampling resolution problem by re-scaling the point cloud. We scale the point clouds for biological data - where point clouds are derived from image data - by using the physical pixel size and normalizing the data such that distances between points are scaled in Angstroms.

Cropping and stitching. Next, we tackle the memory cost of computing. In the case of the DIST model, the size of the point cloud poses a challenge due to cubic complexity. This makes it impossible to process very large scenes due to GPU memory constraints. Although many methods for improving memory efficiency were demonstrated (Kyzirakos et al. [2016], Hui et al. [2021], Liu et al. [2019]), all of them achieve it at the cost of computation speed, down-sampling, or losing semantic or geometrical information. We define the instance segmentation problem as the segmentation of geometrically similar objects. Therefore, we expect most of the information to be present within local regions. With this in mind, we reduced the size of the point clouds by cropping. During training, we select crops at random. For inference, we split the point cloud into tiled regions and then use overlap between the regions to stitch the graph across region boundaries. For each dataset, we selected the maximum crop size that would fit into GPU RAM.

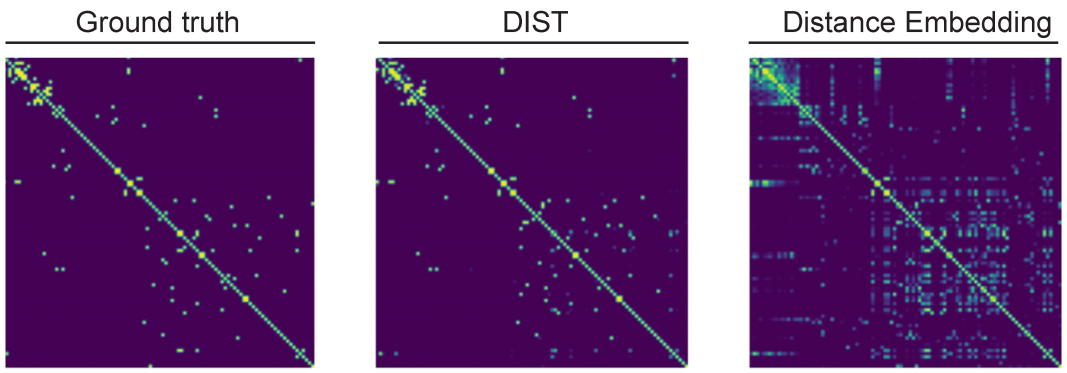
D Graph cut

For some problems, we may have additional information about the graph structure. For example, for filament and membrane segmentation, we know the graph is defined as a linear chain in which a node can have at most 2 neighbors. In this case, we adopt a greedy algorithm for graph inference (**Appendix H**). First, we build a hash map for each node in the point cloud containing node ID P_i, edge probability $p_{i,j}$. Next, for each new instance, we searched the hash map for the initial node. This allows us in the final step to iteratively find up to two edges with the highest probability for all connected nodes. This process is continued until no new edges can be recognized for a given instance. We use this approach when segmenting membranes and microtubules which follow this chain structure.

E Ground truth generation

In order to train and evaluate the DIST model we need to build an instance graph representation of the point cloud. The graph representation is a 2D matrix with nodes (P_i, j; matrix diagonal) representing each individual coordinate point, and the edge represent spatial connectivity between two nodes i and j, where i is the matrix row, and j is matrix column. The graph representation for filament-like structures (membranes and microtubules) is constructed from ground truth annotations in which each individual instance is an ordered list of nodes. Knowing the order, the edge matrix is defined as 1 for each $P_{i,j}$ if i is in the same instance as j and j is a neighbor of i in the ordered list. This approach imposes restrictions where each i can have only up to two edges.

F Graph prediction example



Appendix figure 2: Example of graph prediction.

G Triangular multiplicative edge feature update

Algorithm 1 Triangulation algorithm

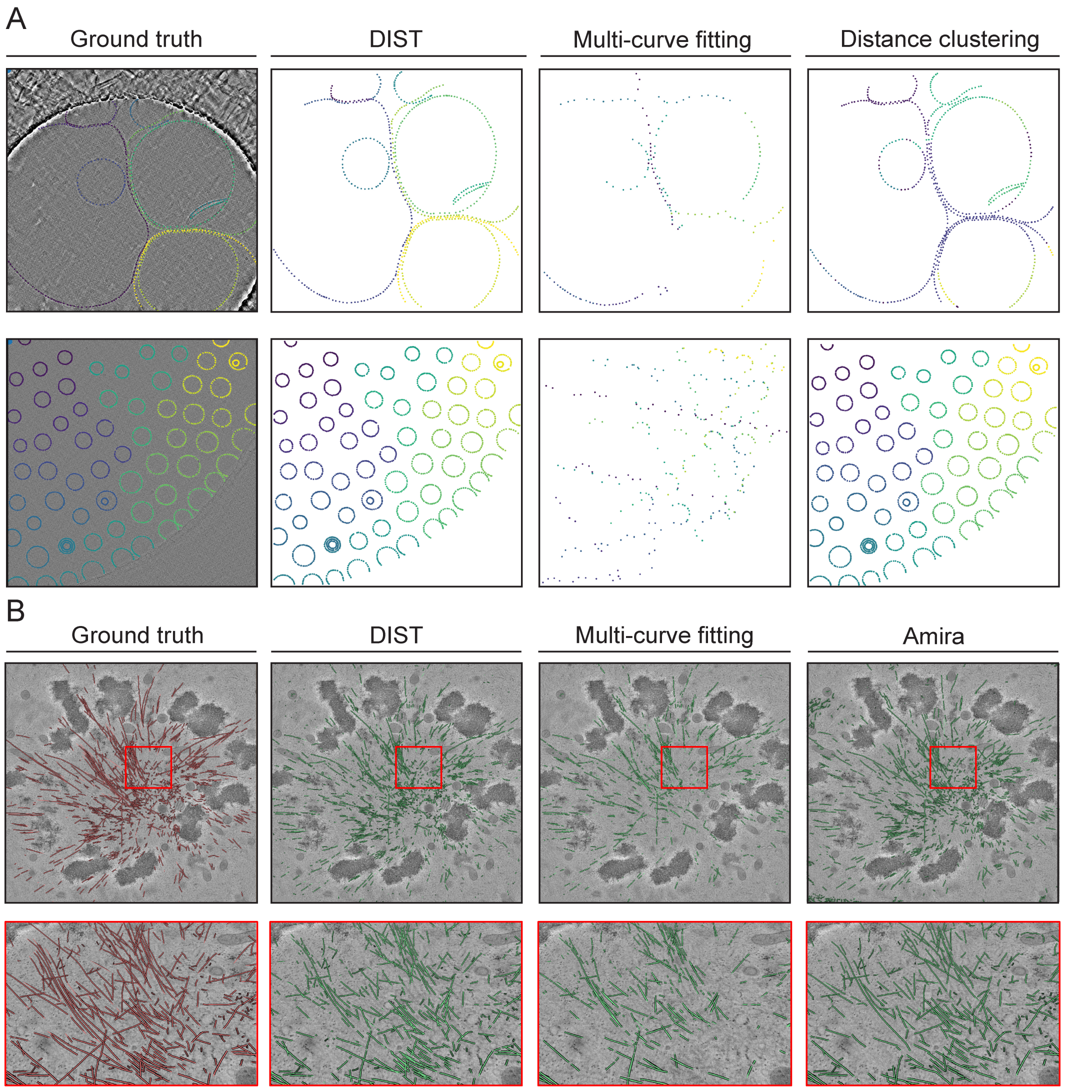
```
Require: $edge_features$
  $edgefeatures \leftarrow layerNorm(edgefeatures)$        ▷ Initial normalization of edge features

Ensure:
  $a = torch.sigmoid(Linear(edgefeatures)) * Linear(edgefeatures)$
  $b = torch.sigmoid(Linear(edgefeatures)) * Linear(edgefeatures)$

  if $axis$ == 2 then
      $k = einsum(biko, bjko- > bijo, a, b)$
  else
      $k = einsum(bkio, bkjo- > bijo, a, b)$
  end if

  $o = torch.sigmoid(Linear(edgefeatures)) * Linear(LayerNorm(k))$
```

H Greedy algorithm

Algorithm 2 Greedy algorithm for graph inference

```
Require: $C^{k}_{patch}\ k \leftarrow P^{local}_{3}$
Require: $G^{i,j}_{patch}\ i \leftarrow P^{local}_{i}\ j \leftarrow P^{local}_{j}$
Require: $IDX^{k}_{patch}\ k \leftarrow P^{global}_{ID}$

  for $coord, graph, index \leftarrow C^{k}_{patch}, G^{i,j}_{patch}, IDX^{k}_{patch}$ do          ▷ Adjacency matrix
      for $i, j \leftarrow graph$ do
          $adjacency \leftarrow index^{k}_{i}$
          $adjacency \leftarrow coord_{i}$
          $adjacency \leftarrow index^{graph_j >= threshold}_{j}$
          $adjacency \leftarrow graph^{graph_j >= threshold}_{j}$
      end for
  end for

  $segment = []$                                                    ▷ Zero-out greedy algorithm
  $stop = False$
  $segment_{id} = 0$

  while not $stop$ do                                               ▷ Greedy instance segmenter
      $ids \leftarrow len(adjacency_{index_j}) >= 0$                 ▷ Pick initial point from Adjacency
      $NewSegment \leftarrow ids$

      $growing = False$
      while not $growing$ do                              ▷ Find all nodes associated with initial node
          $size = len(NewSegment)$

          for $id \leftarrow ids$ do
              Pick all points associated with the initial point
              Check 1: Check if id is not already on the $NewSegment$
              Check 2: Check if id is reversible connected $i \leftarrow j$ and $j \leftarrow i$
              Check 3: Check if id is not associated to already segmented instance

              Add new nodes to $NewSegment$
          end for

          if $len(NewSegment) == size$ then $growing = True$
          end if
      end while

      $Sort(NewSegment)$                                            ▷ Optional: Sort point in filament
      $Smooth(NewSegment)$                                          ▷ Optional: Smooth point in filament

      $segment.append(segment_{id} + NewSegment)$
      $prune(NewSegment)$                                ▷ Remove segmented nodes from adjacency

      if $sum(adjacency) == 0$ then $stop = True$    ▷ Stop segmenter when there no more points
      else
          $segment_{id}$ += 1
      end if
  end while
```

I Comparison of DIST instance segmentation performance



Appendix Figure 3: Example of instance segmentation using DIST.
(**A**) Illustration of 2D cryo-EM micrograph with segmented membranes. Instance segmentation was compared with MCF and DC methods. (**B**) Electron tomography reconstruction of mammalian cells with segmented MTs in 3D. Instance segmentation was compared with MCF and Amira methods.