UC Santa Barbara

Title

A robust approach to 3D neuron shape representation for quantification and classification.

Permalink

https://escholarship.org/uc/item/8dz2x3kk

Journal

BMC Bioinformatics, 24(1)

Authors

Jiang, Jiaxiang
Goebel, Michael
Borba, Cezar
et al.

Publication Date

2023-09-28

DOI

10.1186/s12859-023-05482-y

Peer reviewed

RESEARCH

A robust approach to 3D neuron shape representation for quantification and classification

Jiaxiang Jiang[1*], Michael Goebel[1], Cezar Borba[3], William Smith[2] and B. S. Manjunath[1*]

*Correspondence:
jjiang00@ucsb.edu; manj@ucsb.edu

[1] Department of Electrical and Computer Engineering, University of California, Santa Barbara, USA
[2] Department of Molecular, Cell and Developmental Biology, University of California, Santa Barbara, USA
[3] The Whitney Laboratory for Marine Bioscience, University of Florida, St. Augustine, USA

Abstract

We consider the problem of finding an accurate representation of neuron shapes, extracting sub-cellular features, and classifying neurons based on neuron shapes. In neuroscience research, the skeleton representation is often used as a compact and abstract representation of neuron shapes. However, existing methods are limited to getting and analyzing "curve" skeletons which can only be applied for tubular shapes. This paper presents a 3D neuron morphology analysis method for more general and complex neuron shapes. First, we introduce the concept of skeleton mesh to represent general neuron shapes and propose a novel method for computing mesh representations from 3D surface point clouds. A skeleton graph is then obtained from skeleton mesh and is used to extract sub-cellular features. Finally, an unsupervised learning method is used to embed the skeleton graph for neuron classification. Extensive experiment results are provided and demonstrate the robustness of our method to analyze neuron morphology.

Keywords: 3D neuron morphology, Skeleton mesh, Graph, Sub-cellular features, Embedding, Classification

Introduction

The importance of neuronal morphology has been recognized from the early days of neuroscience [1]. There are three obstacles in automatic neuron morphology analysis. First, we need to have a good shape representation of each neuron. Skeleton representations are widely used in neuroscience [2–5] as they provide a compact and abstract shape representation. Mathematically, skeletonization or medial axis transform (MAT) has a rigorous definition for arbitrary shapes. The skeleton of a shape is defined as a collection of interior points that have at least two closest points on the surface of the shape. We refer to those interior points as skeleton points and each skeleton point is associated with a radius. Figure 1 shows an example of MAT. However, in reality, it is not an easy task to get skeleton representation directly from images. Most automatic or manual segmentation methods output a cloud of surface points. Thus, we need to compute the 3D neuron skeleton from 3D surface point clouds. The skeleton representation further
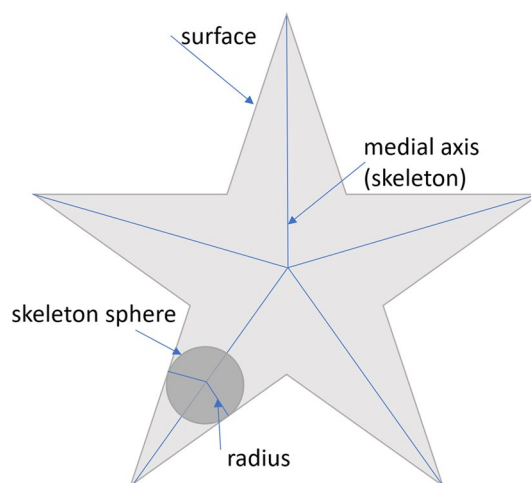
Fig. 1 Illustration of MAT by using a 2D shape example

enables computing sub-cellular features such as length and number of branches of neurons as well as classification of neurons.

The main contribution of this paper is a robust and efficient method for computing a skeleton representation from a set of 3D surface points. This 3D skeleton representation can be used for a quantitative analysis of neuronal cell structures, including sub-cellular feature calculations and for neuron type classification based on 3D shapes.

There is an extensive literature on neuron skeleton extraction [3, 6, 7]. In [3], the skeleton representation is computed from a 3D mesh by using a traditional morphological thinning algorithm [8]. This method has two main drawbacks. First, the thinning algorithm is sensitive to noise of 3D mesh. Second, in reality, we usually get discrete 3D surface points of neurons from the segmentation step, and constructing 3D mesh from those discrete 3D surface points will introduce additional noise. To make the skeleton extraction model more robust, Ref. [7] proposes to use deep learning network to learn skeleton representation. The main idea of the paper is to use the deep learning network to predict skeletons from features of multiple spatial scale layers. This model still takes a continuous surface as input, as opposed to discrete surface points. Further, this is a supervised method and it requires training samples. In [6], they propose extracting skeleton representations directly from discrete surface points by using a 3D discrete distance transform. However, this only works well for curve skeletons and only tubular structures have curve skeletons. General 3D shapes will result in surface skeletons as shown in Fig. 2. In practice, skeleton mesh is used to represent the surface skeleton. There is no existing method to extract skeleton mesh from surface point clouds for neuron morphology analysis.

There are also methods to analyze neuron shapes using the skeleton representation. For skeleton classification task, [3, 9–13] use predefined handcrafted features to represent neuron morphology and classify neurons based on those representations. [3] proposes to compare similarity of skeletons by using local skeleton features. It breaks a neuron skeleton into short segments and characterize segments by location and direction of segments. Similarly, compared to [3, 11] provides a more robust
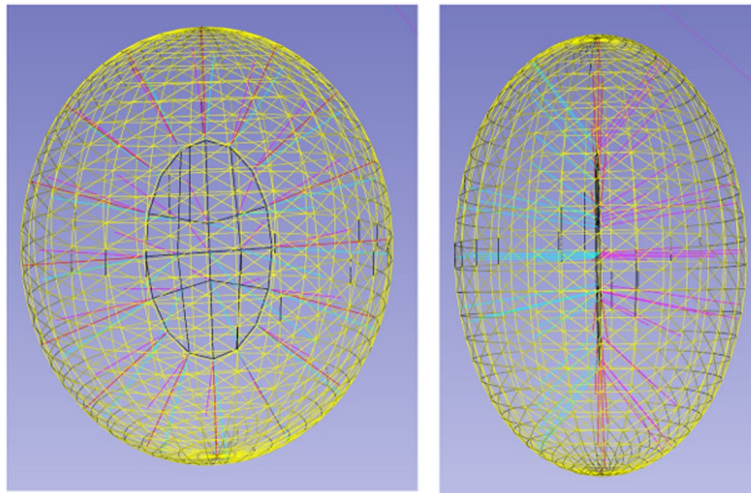
Fig. 2 Visualization of a 3D ellipsoid shape and its surface skeleton from two points of view. Yellow triangle mesh represents object surface. Black contour represents the outline of the skeleton surface. Magenta and Cyan line segments represent two closest surface points from the skeleton point. Two colors are used to differentiate different directions

method to find branches of neuron which do not have many local fluctuations. Reference [9, 10] extract other cellular and sub-cellular features, such as length of neuron, the surface area of soma, dendritic length from "curve" skeletons to represent neuron morphology. Reference [12] models the skeleton as a graph and uses paths in the graph as the feature to represent neuron morphology. Reference [13] defines a specific descriptor function to capture global and local information from the skeleton. The main drawback of the handcrafted feature is its limited representation capability. To solve the problem of handcrafted feature, several learning based methods [14–16] have been proposed recently to classify neuron skeletons. References [14, 15] project 3D skeletons back to 2D images and use convolution neural networks to get skeleton representation from 2D images. However, 3D shape information is lost when projecting onto 2D images even with multi-view projections like [14]. To avoid projecting 3D skeletons into 2D images, Ref. [16] models the neuron skeleton as a graph directly and proposes a contrastive graph neural network (GNN) learning framework to represent the neuron. Similar to all above mentioned methods, Ref. [16] only works for "curve" skeletons but not for surface skeletons. There are fundamental differences between curve skeletons and skeleton meshes. For example, there are no cycles in curve skeletons but that is not the case in skeleton meshes. Ref. [17] can handle skeleton meshes. However, they only consider features from skeleton points for the classification and it does not fully utilize the skeleton mesh information.

  To analyze general neuron shapes, this paper presents a robust 3D neuron morphology analysis framework based on the surface skeleton representation of neurons. In [18], the authors propose an unsupervised deep learning skeleton mesh extraction method. However, this method does not work well when neurons have concave shapes. Our skeleton mesh extraction method is built upon [18], and by using estimated surface norm of point clouds as part of the optimization function, we address

this drawback. Next the skeleton mesh is converted to an undirected graph called skeleton graph. Inspired by [19], we embed the skeleton graph by maximizing mutual information, and then classify neurons based on the embedding of each skeleton. To compute cellular/sub-cellular features of neurons from the skeleton representation, we also utilize the skeleton graph. A simple but effective recursive algorithm is proposed to get number of branches and length of neurons.

We apply our neuron morphology analysis method to classify *Ciona* neurons. The *Ciona* sea squirt is one of the widely studied tunicates in neuroscience [20]. Its brain is closely related to vertebrates with a much simpler neuronal structure. In a single *Ciona* larva, it has only about 187 neurons with about 6600 synapses [20]. Studying the *Ciona* brain in depth can reveal the general principles behind the mechanism of how vertebrate brains work [21]. We also present our results on the NeuroMorpho[22] dataset. In summary, the main contribution of our paper include

- A robust and efficient skeleton mesh extraction method with novel cost function by using properties of MAT. To the best of our knowledge, this is the first one to use skeleton mesh instead of the curve skeleton to analyze neuronal shapes
- A 3D *Ciona* neuron dataset that can be used for neuron morphology analysis.

Method

Figure 3 illustrates our overall neuron morphology analysis method. Given a set of surface point clouds as input, we introduce an unsupervised deep learning method to get the skeleton mesh representation of each neuron. This is achieved by using the properties of the traditional medial axis transform (MAT). The skeleton mesh representation includes skeleton points with radii as well as the connection of those skeleton points as shown in Fig. 3. Second, the skeleton representation of each neuron is transformed into a skeleton graph. Each node in the skeleton graph represents a skeleton point. If there is an edge between two nodes, it means those two skeleton points are connected. The weight of the edge represents distance between the two skeleton points. Radii as well as the location of each skeleton point are attributes of each node. Next, length and number of branches of neurons are computed from the skeleton graph. To compare different shapes of neurons, a graph level representation learning method is used to embed
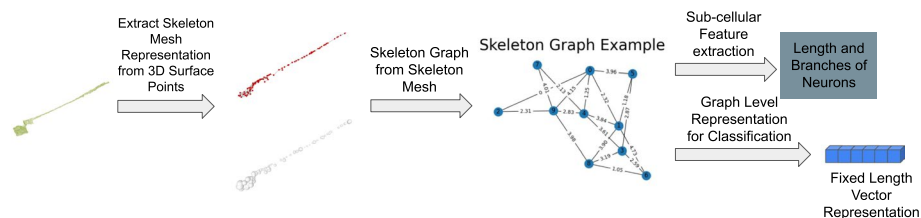

Fig. 3 Overview of our proposed neuron morphology analysis pipeline. Given a surface point cloud as input, we extract the skeleton mesh. The skeleton mesh includes skeleton points with their radii as well as the connection of skeleton points. Then we construct the skeleton graph. Each node in a skeleton graph represents a skeleton point, and edge in the graph represents the connection between skeleton points. Next, we propose a graph analysis method to get length and number of branches of neurons based on the skeleton graph. We also use the skeleton graph for classification task by embedding it into a fixed length vector

the skeleton graph. The representation learning method is an unsupervised method that maximizes mutual information of the skeleton graph.

Skeleton mesh from 3D surface point cloud

Our unsupervised 3D neuron skeleton extraction method is built upon the method in [18] as illustrated in Fig. 4. Given a 3D surface point cloud as input, PointNet++ [23] is used as the encoder to obtain the sampled surface points with features. Next, a multi-layer-perceptron (MLP) is used to learn the geometric transformation to predict the skeleton points with their radii using linear combination of the MLP input points. Compared to [18], we propose to use properties of skeleton points as the prior knowledge to make the geometric transformation learning process more robust to general shapes. After getting skeleton points with radii, a graph auto-encoder is used to predict links between skeleton points.

Skeleton points prediction

Mathematically, given a set of 3D surface points $\mathbf{P} \in \mathbb{R}^{M \times 3}$ where M is a number of points, we want to predict N skeleton spheres $\mathbf{s}_i = <\mathbf{c}_i, r_i>$ where $\mathbf{c}_i$ is the center of each sphere and r_i is the radius of each sphere.

As illustrated in Fig. 4, we first use PointNet++ [23] as the encoder to obtain a set of sampled input points $\mathbf{P}' \in \mathbb{R}^{M' \times 3}$ and their associated contextual features $\mathbf{F} \in \mathbb{R}^{M' \times D}$. $M'(M' < M)$ is the number of feature points after PointNet++ and D is the feature dimension of each feature point. PointNet++ groups points and extract point features hierarchically. It contains a number of set abstraction levels. For each set abstraction level, there are three layers: Sampling layer, Grouping layer, and PointNet layer. For the first set abstraction level, the input is $\mathbf{P}$, a set of M number of 3D surface points. Next, Sampling layer is applied. The iterative farthest point sampling (FPS) [23] is used to get M' sampled points. In the grouping layer, those M' sampled points are used as the centroid points. Then for each centroid, all M points within a radius are viewed as neighbor points and are grouped into that local region. Therefore, each centroid has K neighbors and K can vary for different groups. After Sampling and Grouping layers, PointNet [24] layer is used to extract features for each local region. The sampling layer, grouping layer, and PointNet layer consists one set abstraction level, and we stack such abstraction levels to form a hierarchical architecture to get
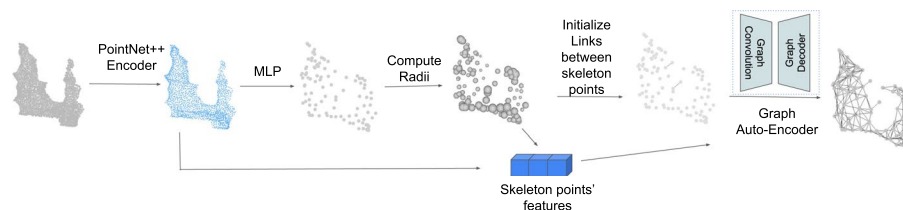

Fig. 4 Overview of neuron skeleton representation method. Given 3D surface point cloud as input, PointNet++ [23] is used to extract features of the input point cloud. Then a geometric transformation learned by MLP will predict the skeleton points location with their radii. After skeleton points prediction, two simple priors are used to initially connect some skeleton points, and a graph auto-encoder is used to predict all links that connect skeleton points

Jiang *et al. BMC Bioinformatics* (2023) 24:366

Page 6 of 19

features at different spatial scales. Next, multi-scale grouping is applied to concatenate the features from different spatial scales.

A Multi-Layer Perceptron (MLP) is used to estimate the geometrical transformation to get a set of skeleton spheres' center points $\mathbf{C}$, $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_N\}$. The geometric transformation we use is a convex combination of input points $\mathbf{P}'$. MLP with softmax function is used to estimate the weight $\mathbf{W} \in \mathbb{R}^{M' \times N}$ of the convex combination in Eq. 1.

$$\mathbf{C} = \mathbf{W}^T \mathbf{P}' \qquad \text{subject to} \qquad \forall j \in \{1, ..., N\} \quad \sum_{i=1}^{M'} W_{i,j} = 1 \tag{1}$$

As shown in [18], the same weight matrix $\mathbf{W}$ can be used to compute $r(\mathbf{c}) \in \mathbf{R}$ using Eq. 2

$$\mathbf{R} = \mathbf{W}^T \mathbf{D} \tag{2}$$

where $D \in \mathbb{R}^{M' \times 1}$ is a vector of $d(\mathbf{p}', \mathbf{C})$. $d(\mathbf{p}', \mathbf{C})$ is the closest distance of one surface point $\mathbf{p}'$ to all skeleton points $\mathbf{C}$ and is defined as $d(\mathbf{p}', \mathbf{C}) = \min_{\mathbf{c} \in \mathbf{C}} ||\mathbf{p}' - \mathbf{c}||_2$

A set of loss functions are defined in [18] to train the MLP. The loss function includes sampling loss L_s, point-to-sphere loss L_p, and radius regularizer loss L_r. The first two losses are based on the recoverability of skeleton representation. The last loss term is to encourage larger radii to avoid instability induced by surface noise.

For the sampling loss L_s, we sample points on the surface of each skeleton sphere and measure the Chamfer Distance (CD) between the set of sampled sphere points $\mathbf{T}$ and the set of sampled surface points from PointNet++ $\mathbf{P}'$ as in Eq. 3:

$$L_s = \sum_{\mathbf{p}' \in \mathbf{P}'} \min_{\mathbf{t} \in \mathbf{T}} ||\mathbf{p}' - \mathbf{t}||_2 + \sum_{\mathbf{t} \in \mathbf{T}} \min_{\mathbf{p}' \in \mathbf{P}'} ||\mathbf{t} - \mathbf{p}'||_2 \tag{3}$$

Point-to-sphere loss L_p measures the reconstruction error by explicitly optimizing the coordinate of skeleton points and their radii:

$$L_p = \sum_{\mathbf{p}' \in \mathbf{P}'} \left(\min_{\mathbf{c} \in \mathbf{C}} ||\mathbf{p}' - \mathbf{c}||_2 - r(\mathbf{c}_{p'}^{min}) \right) + \sum_{\mathbf{c} \in \mathbf{C}} \left(\min_{\mathbf{p}' \in \mathbf{P}'} ||\mathbf{c} - \mathbf{p}'||_2 - r(\mathbf{c}) \right) \tag{4}$$

where $\mathbf{C}$ is a set of predicted skeleton points, $r(c)$ is a radius of skeleton point $\mathbf{c}$, and $\mathbf{c}_{p'}^{min}$ is the cloest skeleton points to a point $\mathbf{p}'$.

Radius regularizer loss L_r is defined in Eq. 5 where $r(\mathbf{c})$ is a radius of the skeleton point $\mathbf{c}$. By minimizing this loss, it encourages large radii of skeleton points to make the skeleton points prediction more stable.

$$L_r = - \sum_{\mathbf{c} \in \mathbf{C}} r(\mathbf{c}) \tag{5}$$

However, based on above three losses, predicted skeleton points can be outside of a shape if the shape is concave. Therefore, we introduce the skeleton-to-surface norm loss L_n to encourage the skeleton points to be inside the shape. L_n is a term to measure the reconstruction error by utilizing the property of spoke direction in MAT. Figure 5 illustrates a spoke of a skeleton point in MAT. The length of a spoke of the skeleton point $\mathbf{c}$
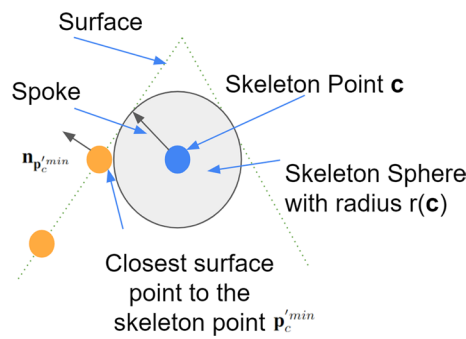
Fig. 5 Spoke is a vector connecting a skeleton point and that skeleton point's one of two closest surface points. The vector points from the skeleton point to the surface point. Green dashed lines represent the implicit surface of an object, blue dot is one skeleton point, orange dots represent surface points, and the arrow represents a spoke. Spoke is perpendicular to the object surface at the surface point

is $r(\mathbf{c})$. This is also one of our main contribution compared to [18] for skeleton points prediction. In theory, the direction of the spoke should be perpendicular to the object surface at the surface point [17]. Also the spoke direction should be pointing outside of a shape. To capture this property, we define L_n:

$$L_n = \sum_{\mathbf{c} \in \mathbf{C}} \left(1 - \mathbf{n}_{\mathbf{p}_c^{'min}} \cdot \frac{\mathbf{p}_c^{'min} - \mathbf{c}}{||\mathbf{p}_c^{'min} - \mathbf{c}||_2}\right) + \sum_{\mathbf{p}' \in \mathbf{P}'} \left(1 - \mathbf{n}_{p'} \cdot \frac{\mathbf{p}' - \mathbf{c}_{p'}^{min}}{||\mathbf{p}' - \mathbf{c}_{p'}^{min}||_2}\right) \tag{6}$$

$\mathbf{p}_c^{'min}$ is the closest surface point to the skeleton point c and $\mathbf{n}_{\mathbf{p}_c^{'min}}$ is the estimated surface norm of the surface point. The "·" denotes the dot product between two vectors. To estimate the norm of each point in the 3D surface points, the adjacent points are found first and then principal axis of the adjacent points using covariance analysis are calculated. More details of the norm estimation of each surface point are described in [25]. L_n encourages the skeleton points to be located within a shape even the shape is concave.

Links prediction

After predicting skeleton points, our target is to predict links to connect skeleton points to form the skeleton mesh. In theory, for any pair of skeleton points, if all points that are on the line connecting the two skeleton points are also on the skeleton surface, there should be links between those two points. We adapt the graph auto encoder (GAE) as used in [18] to predict links between skeleton points. GAE takes input the initialized adjacency matrix A_{ini} of the skeleton mesh graph G_{mesh} and the skeleton points' features. The skeleton points' features is concatenation of $\mathbf{C}$, $\mathbf{R}$, and $\mathbf{W}^T \mathbf{F}$. $\mathbf{C}$ are coordinates of skeleton points, $\mathbf{R}$ are radii of skeleton points, $\mathbf{W}$ is the learned weights from the MLP, and $\mathbf{F}$ is the contextual features of the surface points from PointNet++. GAE outputs the estimated adjacency matrix $\hat{A}$ of G_{mesh}. The loss function is a Masked Balanced Cross-Entropy (MBCE) loss as proposed in [26].

Sub-cellular feature extraction from the skeleton graph

Skeleton model can be represented as the skeleton graph $G(V, E)$ where V represents all skeleton points and E represents connection between skeleton points. The weight of the edge represents distance between skeleton points.

Neuron length computation from skeleton graph

We formulate the neuron length computation problem as finding the longest simple path in the skeleton graph *G*. A simple path in the graph is a path that does not have repeat nodes. In *G*, the length of the path is the sum of all edges' weights along the path. Note that our skeleton graph can have loops. To solve this problem, for each node, we find the longest simple path from that node and denote as *path*(*i*) for node *i*. To avoid getting stuck during the loop, we mark any node when we visits as shown in the algorithm below. Next, we find *i** that maximizes *path*. We use the following recurrent algorithm to find the longest simple path from one node in the skeleton graph.

Algorithm 1 Find the longest simple path from node i

```
D[j] represents the longest path from j to i. Initially, D[j]=0 for all j
function LONGESTPATH(i, currLength)
    if Node i is visited then
        return
    end if
    change i status to visited
    if D[i] <currLength then
        D[i]=currLength
    end if
    for all nodes j that is connected to i do
        LongestPath(j, currLength+edgeweight[i,j])
    end for
    change i status to not visited
end function
```

Algorithm 1 Find the longest simple path from node i

Neuron branch calculation

After finding the longest simple path, we are able to identify a set of nodes on that path. Those nodes are possible branching nodes. We name a set containing all possible branching nodes as **B** For each node *i* ∈ *B*, we find the longest simple path from that node *i* which does not contain any other nodes in *B*. Therefore, the branch is identified as the longest simple path.

Skeleton model comparison

We cluster neuron morphology by comparing different skeleton graphs. Specifically, we embed the skeleton graphs and then cluster neurons based on their embeddings. We embed the skeleton graph based on InfoGraph [19] as illustrated in Fig. 6. The embedding process is in an unsupervised manner.

First, graph convolutional layers are used to generate node features which we refer to as patch representation $\mathbf{h}_i^j$ (*i* is the skeleton graph index and *j* is the node index of the skeleton graph *i*). Then graph-level pooling is used on all patch representations to get the graph level representation (global representation) $\mathbf{H}_i$. The mutual information (MI) estimator on global-patch pairs over the given graph dataset **G** is defined as:

$$MI = \sum_{i \in K} \frac{1}{K} \sum_{j \in G_i} I(\mathbf{h}_i^j, \mathbf{H}_i) \tag{7}$$

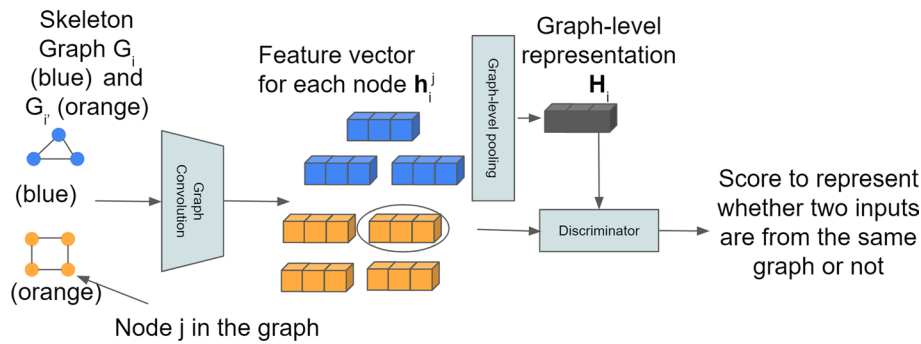where *K* is the total number of graphs in the dataset and $G_i \in \mathbf{G}$.

Fig. 6 We use two example skeleton graphs (blue and orange) to demonstrate how we embed the skeleton graph. Each node of a skeleton graph is encoded into a feature vector by using graph convolution layers. A fixed length graph level feature vector (global representation) is obtained by graph-level pooling operation of each node feature vector. The discriminator takes inputs both global representation and patch representation to decide whether they are from the same skeleton graph. In this toy example, there will be 14 global-patch pairs

MI is the mutual information estimator modeled by the discriminator *T*. The Jensen-Shannon MI estimator proposed in [19] is

$$I(\mathbf{h}_i^j, \mathbf{H}_i) = \mathbb{E}\left[-sp(-T(\mathbf{h}_i^j, H_i)\right] - \mathbb{E}\left[-sp(-T(\mathbf{h}_{i'}^j, H_i)\right] \tag{8}$$

where $\mathbb{E}$ is the expectation (here it is just average operation) and $sp(z) = log(1 + e^z)$. i and i' denote two graph samples from the dataset **G**. The discriminator *T* estimates global-patch representation pairs by passing two representations to different non-linear transformations and then takes the dot product of the two transformed representations. Both non-linear transformations consist 3 linear layers with ReLU activation functions. Therefore, the discriminator will output a score between $[0, \infty)$ to represent whether the input patch (node) is from the input graph. If the input global/patch pairs are from the same graph, we refer to them as positive samples, otherwise negative samples. We randomly sample pairs as input to the discriminator.

Dataset

Ciona **neuron EM dataset**

The first dataset (Dataset 1) contains two *Ciona* larva 3D TEM images [20]. The section thickness for TEM images varies between 35 nm and 100 nm. For each section, xy resolution is 3.85×3.85 nm. Animal 1 contains 7671 sections and animal 2 contains about 8000 sections. In each *Ciona* larva, 187 neurons are annotated. Those 187 neurons can be grouped into 31 types. For animal 2, *Ciona* neuron skeletons are traced using TrackEM2 [27], an ImageJ [28] plugin. This dataset is summarized in Table 1, and we refer to [20] for more details.

C.elegans **Neuron dataset from NeuroMorpho**

NeuronMorpho [22] is a publicly available dataset that is used for neuron morphology research. It has dozens of different animals' neurons. So far, it is the largest neuron skeletons dataset with associated metadata. In this paper, we take a subset of *C.elegans* dataset (Dataset 2) from the whole NeuroMorpho dataset to verify our sub-cellular feature

Jiang *et al. BMC Bioinformatics*   (2023) 24:366

Page 10 of 19

Table 1 Details of *Ciona* Dataset

Animal	xy resolution (nm)	Section thickness (nm)	Number of sections	Annotations
Animal 1	3.85×3.85	35–100	7671	3D surface point cloud of neurons are provided
Animal 2	3.85×3.85	35–100	6928	3D neuron skeletons without skeleton points' radii

It contains two *Ciona* animals, one with surface point cloud annotated and one with skeleton annotated

extraction method and neuron skeleton comparison method. Dataset 2 consists of 1240 neuron skeletons (with radii) and it is classified into 3 different types. Each neuron with detailed metadata information such as number of branches and length of neuron.

Major type neurons from NeuroMorpho

We collect 5 major cell types from NeuromMorpho dataset (Dataset 3) including 20614 pyramidal neurons, 956 ganglion neurons, 2674 granule neurons, 1617 medium spiny neurons, and 320 motoneurons. These neurons come from 3 species, human, rat, and mouse.

In additional to the above datasets, we also use the benmark ShapeNet [29] and detailed experimental results are provided in the supplemental material.

Results

Skeleton model from 3D surface point cloud

We apply our method on animal 1 neurons from Dataset 1 for the purpose of building a shape model to analyze neuron morphology. To get the fixed number of 3D surface input points, we use the sampling strategy described in [30]. The main idea of the sampling strategy is to give each point a weight based on its distance to neighbor points. Then we sample points based on the weights until we reach the number of desired points. Details of defining the neighbor points and computing the weights are described in [30].

To evaluate our skeleton extraction method on Dataset 1, we carefully repair surface mesh using screened poisson surface reconstruction method [31] with spherical harmonics to smooth the surface. Figure 7 shows the qualitative comparison of our methods and other state-of-the-art methods [18, 32]. Our method has better visual results. Reference [32] can generate unstructured skeleton points but it lacks topological constraint. We sample the number of points using [30] to be the same with the other methods for a fair comparison. It performs well when neuron has tube like structure but it is not good when neuron has a more circular shape. Compared with [18], our method can capture more detailed structures which are important for sub-cellular feature extraction, such as branches. For a quantitative evaluation of our method on Dataset 1, we compute the strictly defined MAT and use the handcrafted method in [18] to remove insignificant spikes to get the simplified MAT. We sample points on the simplified MAT as the ground truth skeleton points. Then we compute Chamfer Distance (CD) and Hausdorff distance

Table 2 Quantitative Comparison with state-of-the-art skeleton model extraction method on Dataset 1

	CD-recon	HD-recon	CD-skel	HD-skel	vol-pct (%)
DPC [32]	0.102	0.298	0.303	0.311	10.1
Point2Skeleton [18]	0.081	0.207	0.155	0.191	8.2
Our method	0.067	0.183	0.090	0.185	5.6

Table 3 Sub-cellular feature evaluation results on Dataset 1, Dataset 2, and Dataset 3

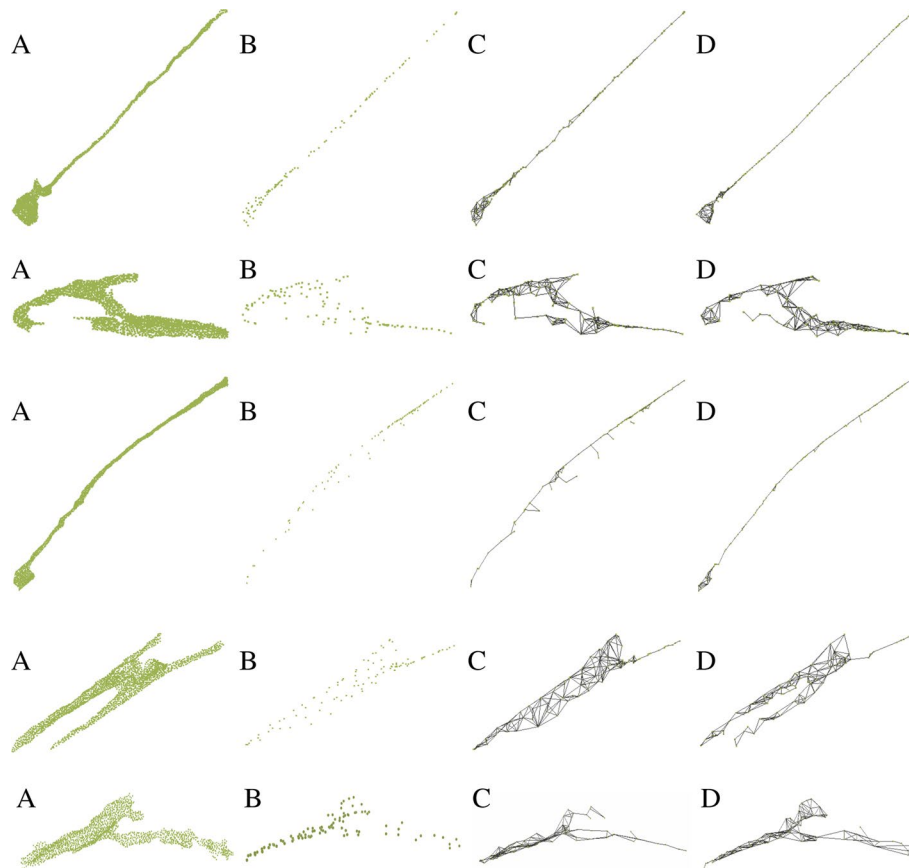|	len-pct on Dataset 1	num-pct on Dataset 1	len-pct on Dataset 2	num-pct on Dataset 2	len-pct on Dataset 3	num-pct on Dataset 3
NAVIS [3]	7.1	10.5	**3.3**	6.5	4.3	5.1
Our method	**5.5**	**7.6**	3.6	**5.9**	**3.3**	**3.9**

Fig. 7 The figure shows skeleton extraction results from different methods. **A** Input 3D surface points; **B** sampled skeleton points from surface points using DPC [32]; **C** skeleton mesh from surface points using Point2Skeleton [18]; **D** skeleton mesh from our method with surface norm cost function

(HD) between computed skeleton points and ground truth skeleton points. We refer to them as CD-skel and HD-skel, respectively. To compute CD-skel and HD-skel, we shift and rescale each skeleton so that the skeleton center is located at (0,0,0) and their x,y,z

coordinates are all between −1 and 1 for every skeleton. We also measure the difference between the shapes reconstructed from the skeletons and ground truth surface points using CD and HD. We refer to them as CD-recon and HD-recon, respectively. Similarly, we also shift and rescale the ground truth points so that each neuron is centered at (0,0,0) and each neuron's surface coordinates are between −1 and 1. Other than those four aforementioned evaluation metrics, we also use the reconstructed neuron volume difference as the evaluation metric, considering neuron volume as one of the important property of a neuron. We denote it as $vol - pct$. Mathematically, it is defined as $vol - pct = \frac{|vrecon - vg|}{vg}$ where $vrecon$ is the volume of the reconstructed neurons from the skeleton model, and vg is the ground truth volume. Table 2 gives the detailed results of different methods. It shows our method has the best performance compared to other methods on Dataset 1 in terms of all of the above evaluation metrics.

Sub-cellular feature extraction from skeleton model

We apply our sub-cellular feature extraction method on Dataset 1, Dataset 2, and Dataset 3. We define the length difference percentage (len-pct) and number of branches difference percentage (num-pct) to measure the neuron length error and number of branches error of the computation methods. We compare our sub-cellular feature extraction method with the state-of-the-art sub-cellular feature extraction method proposed in [3]. Table 3 gives details of sub-cellular feature computation results. len-pct and num-pct for Dataset 1 is for both animals. Our method provides the better sub-cellular feature extraction results in most cases and the percentage error is no more than 8 percent. Also, as we see, our method is comparable to [3] on "curve" skeletons but has better performance on the skeleton meshes ("curve" skeleton is a special case of the skeleton mesh).
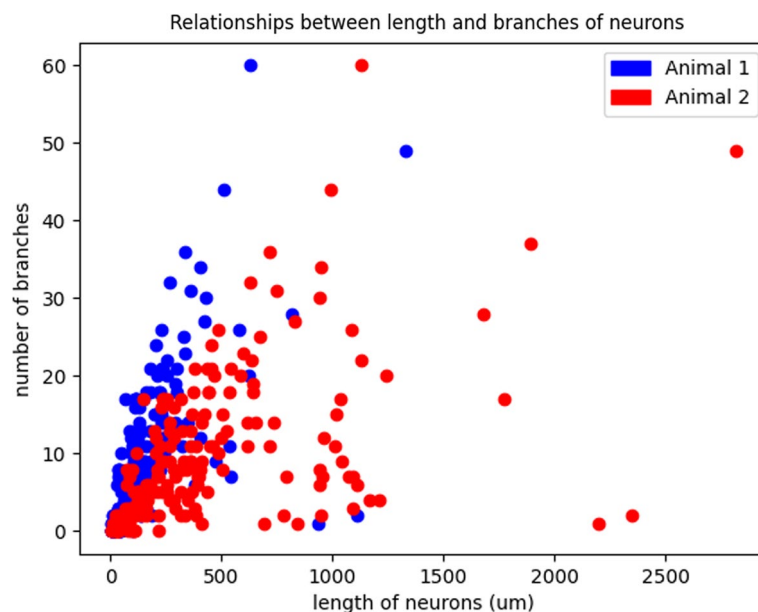

Fig. 8 Relationships between length and number of branches of neurons using two animals of Dataset 1. Blue dots represent neurons from animal 1 and red dots represent neurons from animal 2

For Dataset 1, we also analyze the relationships between length and branches of neurons computed from our method. For animal 1 in Dataset 1, we first use our skeleton extraction method to get the skeleton representation of each neuron. Next, we use our sub-cellular feature extraction method to get length of the neuron and number of branches of the neuron. Figure 8 shows the relationships between length and branches of neurons. Blue dots represent neurons from animal 1 and red dots represent neurons from animal 2. Overall, longer neurons tend to have more branches. This relationship between neuron length and number of branches is consistent between animals 1 and 2.
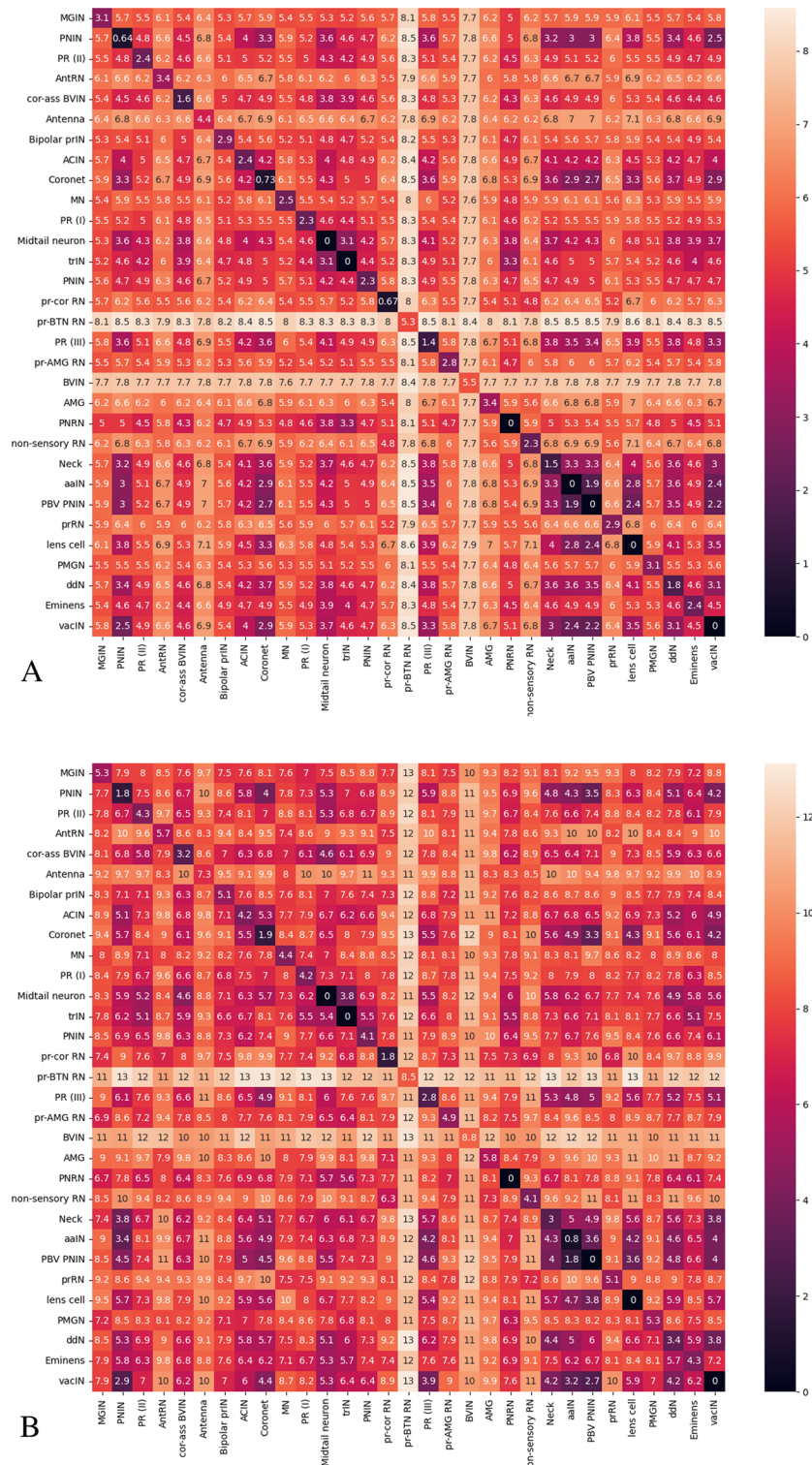
Skeleton model comparison

We apply our skeleton model comparison method on Dataset 1 for the purpose of analyzing *Ciona* neuron morphology of different neuron types. Given the skeleton graph, we embed it into a 100-dimension vector. Then we use K-means++ to cluster vector representations of skeleton graphs. For K-means++, the number of clusters is set to be the same as number of neuron classes. After K-means++, the cluster label is given by using the majority vote of neuron types within the cluster. We use animal 1 neurons to train the K-means++ model and get the cluster (neuron type) centers. Next, we use animal 2 as the test set. For each neuron in the test set, we assign the label based on its closest cluster center. The distance metric we use is the euclidean distance in the vector embedding space. Table 4 shows the comparison of clustering (classification) accuracy on both training and test sets using different neuron classification method. The neuron classification methods include graph spectrum method, graph2vec [33], s-rep [17] and our graph level representation method. The graph spectrum method uses the eigen values of the graph's adjacency matrix to form the vector representation. Similar to our method, graph2vec method is another way to convert the skeleton graph to the graph level vector representation. For the s-rep method, it uses the skeleton points' features such as spoke direction, spoke length, and skeleton points' locations to classify neurons. From Table 4, we observe that grouping neurons by our graph embedding provide the best classification results on both train and test sets. It shows that neuron types are closely related to its morphology. Also, our method is a better way to represent skeleton graphs in terms of clustering accuracy.

Based on previous observations, we do further morphology analysis based on our graph level representation results on Dataset 1. After we get the vector representation of each graph, we compute euclidean distance between each pair of vectors. Then we compute the inter class and intra class distance based on pairwise neuron distance as Fig. 9 shows. Diagonal entries tend to be smaller than other values, confirming a strong correlation between structure and function. More specifically, neurons within a neuron type tend to have a smaller morphological distance than neurons between different groups. Also, two animals inter and intra distance look very similar.

Table 4 Neuron Classification Accuracy on Dataset 1 with skeleton meshes

	Graph spectrum	Graph2vec [33]	S-rep [17]	Our method
Train	0.691	0.767	0.791	0.893
Test	0.632	0.718	0.773	0.871

Fig. 9 Inter and intra class neuron morphology distance on animal 1 (**A**) and animal 2 (**B**). Neuron morphology distance is computed by using euclidean distance between our graph level representation of the skeleton graph
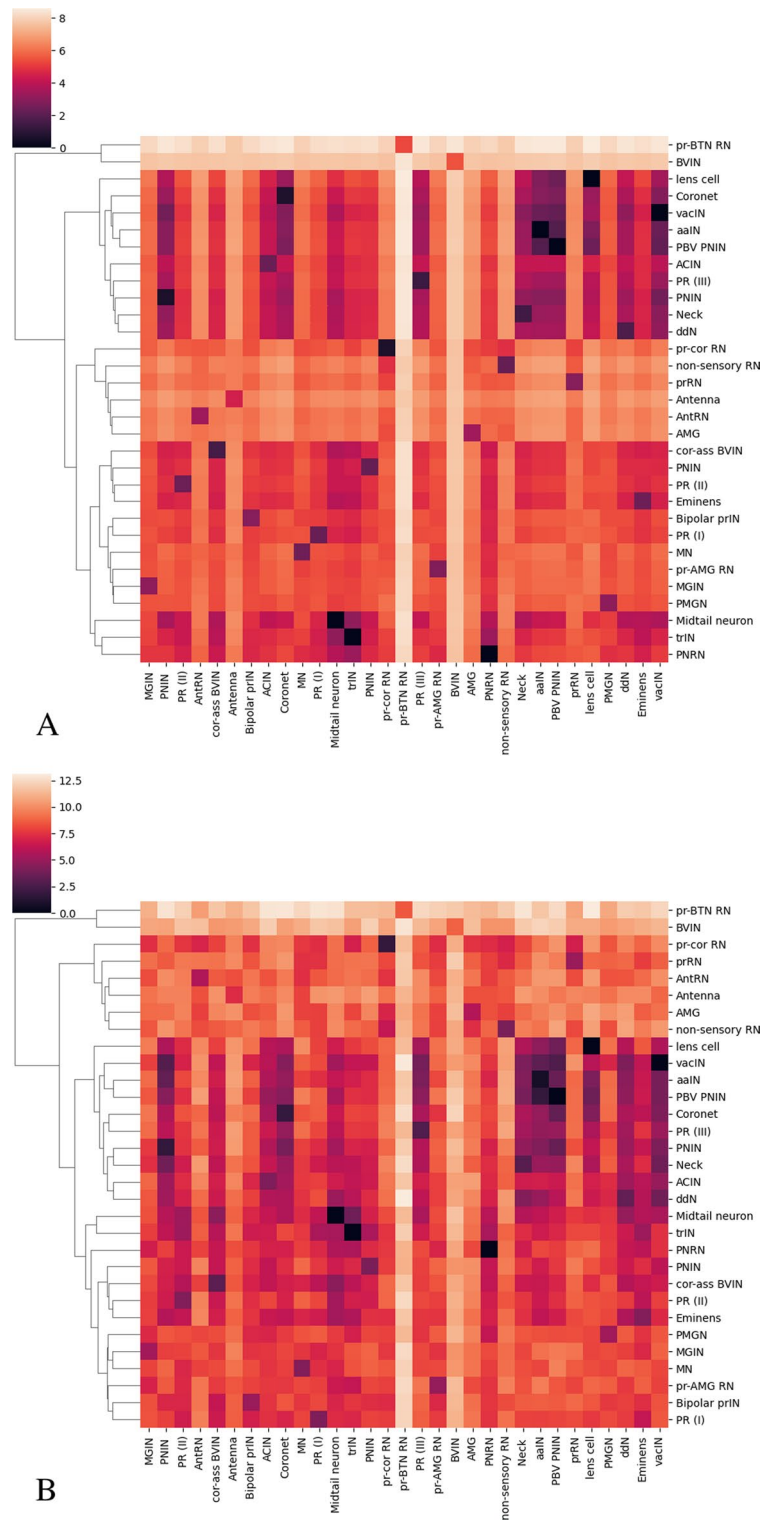
Fig. 10 Hierarchical clustering of neurons of animal 1 (**A**) and animal 2 (**B**)

Table 5 Neuron Classification Accuracy on Dataset 2 with "curve" skeleton

	L-Measure [9]	TMD [11]	NeuroPath2Path [12]	Our method
Motoneuron	0.913	0.945	0.921	**0.953**
Interneuron	0.931	0.911	0.953	**0.962**
Somatic	0.896	0.928	**0.963**	0.957
Average accuracy	0.912	0.931	0.948	**0.955**

Table 6 Neuron Classification (animal species) Accuracy on Dataset 3 with "curve" skeleton

	L-Measure [9]	TMD [11]	NeuroPath2Path [12]	Our method
Human	0.852	0.854	0.877	**0.883**
Mouse	0.837	0.880	**0.889**	0.876
Rat	0.848	0.875	0.891	**0.923**
Average accuracy	0.845	0.876	0.890	**0.898**

Based on this inter and intra class distance, we do hierarchical clustering as shown in Fig. 10. The hierarchical clustering results show that BVIN and pr-BTN RN have larger morphology distances from other neuron types. The BVIN neurons are a broad group of intrinsic interneurons located in the brain vesicle of *Ciona*. The main role of this group is to connect the sensory neurons to other groups within the brain vesicle. The BVIN neurons have partial subclassification based on sensory input [20]. Receiving specific sensory information is an indication of functional role, therefore, the BVIN can be further subdivided into different groups based on the sensory group(s) from which they receive input. Using the sensory input as criteria, the entire group was split up into four groups: prIN if receiving photoreceptor input, antIN if receiving antenna cell input, pr-ant IN if receiving from both, and BVIN if not receiving from either. The pr-BTN RN only have two neurons and their functions are mostly unknown. According to the connectome [20], they receive input from both the photoreceptors and the BTN neurons (neurons involved in processing touch stimuli in the tail), so it's possible they play a role in integrating the two inputs. Any functional differences that may exist between the two are currently unknown, however, the hierarchical clustering suggests that this may be the case.

To show the generality of our skeleton model comparison method, we apply it on Dataset 2 and Dataset 3 which both contain "curve" skeleton neurons for classification. For Dataset 2, we try to classfiy *C.elegans* neurons into 3 predefined cell types (somatic, motoneuron, and interneuron). For Dataset 3, we do two experiments. First, we classify each type of neurons into classes based on which species they belong to. Second, we classify neurons within same species to different neuron types. For all three experiments, we embed each skeleton mesh into a 100-dimension vector and then use K-means++ for the classification, which is the same process for Dataset 1. We compare the performance of our method with available state-of-the-art methods on NeuroMorpho dataset, L-Measure [9], TMD [11], and NeuroPath2Path [12]. We use 10-fold cross validation for our classification task. Tables 5, 6 and 7 show our method has the best performance in most cases. Therefore, in general, our experiment shows that although our method is

Jiang *et al. BMC Bioinformatics*   (2023) 24:366

Page 17 of 19

Table 7 Neuron Classification (cell types) Accuracy on Dataset 3 with "curve" skeleton

	L-Measure [9]	TMD [11]	NeuroPath2Path [12]	Our method
Pyramidal	0.876	0.887	0.902	**0.916**
Ganglion	0.819	0.847	**0.895**	0.884
Granule	0.828	0.879	0.868	**0.912**
Medium spiny	0.820	0.856	0.875	**0.902**
Motoneuron	0.838	0.843	0.874	**0.912**
average accuracy	0.865	0.882	0.896	**0.914**

specifically designed for neuron skeleton mesh representation, it also has comparable performance in terms of "curve" skeleton neuron classification.

Conclusion

In this paper, we propose a novel neuron morphology analysis pipeline. It mainly includes three parts. First, we propose a robust shapre representation using skeleton mesh. Next, we compute sub-cellular features from the skeleton mesh. Finally, we compare different neuron shapes using skeleton mesh. To the best of knowledge, this is the first time that such an approach is used to represent and classify neuronal shapes.

The introduction of the estimated surface norm penalty results in a robust mesh representation that achieves the state-of-the-art performance using well defined metrics.

Based on skeleton graph, we formulate sub-cellular feature computation problem as a longest simple path problem that can be easily computed. To compare different neuron morphology, we use a novel unsupervised graph level representation method to get the vector representation of each skeleton graphs. We provide detailed experimental results to demonstrate the effectiveness of our method. Specifically, our analysis of the *Ciona* dataset demonstrates that shape could be used as a significant feature for classifying neuronal types.

Author contributions
JJ designed the overall neuron morphology analysis pipeline, carried out experiments on different datasets, and drafted manuscript. MG helped on the clustering/classification part of the experiments and helped draft the manuscript. CB provided the *Ciona* neuron class information and helped draft the paper. WS coordinated *Ciona* dataset acquisation, helped analyzed the *Ciona* data, and reviewed the manuscript. BSM coordinated the overall design, development and evaluations of the neuron morphology analysis method, and helped prepare the manuscript.

Availability of data and materials
The shape analysis code is available https://github.com/matlabbaltam/Neuron-Morphology. Dataset 1 is available https://drive.google.com/drive/u/1/folders/1jCGek7U9e7dddsmDz0JKCkWfU0esaiJR and Dataset 2 and 3 are available https://neuromorpho.org/.

Declarations

Ethics approval and consent to participate
Ethical approval was not required as confirmed by the license attached with the open-access data.

Consent for publication
Not applicable

References

1. Halavi M, Hamilton KA, Parekh R, Ascoli GA. Digital reconstructions of neuronal morphology: three decades of research trends. Front Neurosci. 2012;6:49.
2. Jiang S, Pan Z, Feng Z, Guan Y, Ren M, Ding Z, Chen S, Gong H, Luo Q, Li A. Skeleton optimization of neuronal morphology based on three-dimensional shape restrictions. BMC Bioinf. 2020;21(1):1–16.
3. Costa M, Manton JD, Ostrovsky AD, Prohaska S, Jefferis GS. Nblast: rapid, sensitive comparison of neuronal structure and construction of neuron family databases. Neuron. 2016;91(2):293–311.
4. Abdellah M, Hernando J, Eilemann S, Lapere S, Antille N, Markram H, Schürmann F. Neuromorphovis: a collaborative framework for analysis and visualization of neuronal morphology skeletons reconstructed from microscopy stacks. Bioinformatics. 2018;34(13):574–82.
5. Li S, Quan T, Xu C, Huang Q, Kang H, Chen Y, Li A, Fu L, Luo Q, Gong H, et al. Optimization of traced neuron skeleton using lasso-based model. Front Neuroanat. 2019;13:18.
6. Saha PK, Xu Y, Duan H, Heiner A, Liang G. Volumetric topological analysis: a novel approach for trabecular bone classification on the continuum between plates and rods. IEEE Trans Med Imag. 2010;29(11):1821–38.
7. Panichev O, Voloshyna A. U-net based convolutional neural network for skeleton extraction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops; 2019
8. Lee T-C, Kashyap RL, Chu C-N. Building skeleton models via 3-d medial surface axis thinning algorithms. CVGIP: Gr Models Image Process. 1994;56(6):462–78.
9. Scorcioni R, Polavaram S, Ascoli GA. L-measure: a web-accessible tool for the analysis, comparison and search of digital reconstructions of neuronal morphologies. Nat Protoc. 2008;3(5):866–76.
10. Wan Y, Long F, Qu L, Xiao H, Hawrylycz M, Myers EW, Peng H. Blastneuron for automated comparison, retrieval and clustering of 3d neuron morphologies. Neuroinformatics. 2015;13:487–99.
11. Kanari L, Dłotko P, Scolamiero M, Levi R, Shillcock J, Hess K, Markram H. A topological representation of branching neuronal morphologies. Neuroinformatics. 2018;16:3–13.
12. Batabyal T, Condron B, Acton ST. Neuropath2path: classification and elastic morphing between neuronal arbors using path-wise similarity. Neuroinformatics. 2020;18:479–508.
13. Li Y, Wang D, Ascoli GA, Mitra P, Wang Y. Metrics for comparing neuronal tree shapes based on persistent homology. PLoS ONE. 2017;12(8):0182184.
14. Schubert PJ, Dorkenwald S, Januszewski M, Jain V, Kornfeld J. Learning cellular morphology with neural networks. Nat Commun. 2019;10(1):2736.
15. Li Z, Fan X, Shang Z, Zhang L, Zhen H, Fang C. Towards computational analytics of 3d neuron images using deep adversarial learning. Neurocomputing. 2021;438:323–33.
16. Zhao J, Chen X, Xiong Z, Zha Z-J, Wu F. Graph representation learning for large-scale neuronal morphological analysis. IEEE Trans Neural Netw Learn Syst 2022.
17. Pizer SM, Hong J, Vicory J, Liu Z, Marron J, Choi H-y, Damon J, Jung S, Paniagua B, Schulz J, et al. Object shape representation via skeletal models (s-reps) and statistical analysis. Riemannian Geomet Stat Med Image Anal. 2020;1:233–71.
18. Lin C, Li C, Liu Y, Chen N, Choi Y-K, Wang W. Point2skeleton: learning skeletal representations from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021;4277–4286
19. Sun F-Y, Hoffman J, Verma V, Tang J. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In: International Conference on Learning Representations 2019.
20. Ryan K, Lu Z, Meinertzhagen IA. The cns connectome of a tadpole larva of ciona intestinalis (l.) highlights sidedness in the brain of a chordate sibling. Elife. 2016;5:16962.
21. Scheffer LK, Xu CS, Januszewski M, Lu Z, Takemura S-y, Hayworth KJ, Huang GB, Shinomiya K, Maitlin-Shepard J, Berg S, et al. A connectome and analysis of the adult drosophila central brain. Elife. 2020;9:57443.
22. Ascoli GA, Donohue DE, Halavi M. Neuromorpho.org: a central resource for neuronal morphologies. J Neurosci. 2007;27(35):9247–51.
23. Qi CR, Yi L, Su H, Guibas LJ. Pointnet++: deep hierarchical feature learning on point sets in a metric space. Adv Neural Inf Process Syst. 2017;30:1.
24. Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017;652–60.
25. Zhou Q-Y, Park J, Koltun V. Open3d: a modern library for 3d data processing. 2018. arXiv preprint arXiv:1801.09847.
26. Tran PV. Learning to make predictions on graphs with autoencoders. In: 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA), 2018;237–245. IEEE
27. Cardona A, Saalfeld S, Schindelin J, Arganda-Carreras I, Preibisch S, Longair M, Tomancak P, Hartenstein V, Douglas RJ. Trakem2 software for neural circuit reconstruction. PLoS ONE. 2012;7(6):38011.
28. Collins TJ. Imagej for microscopy. Biotechniques. 2007;43(S1):25–30.
29. Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, Savarese S, Savva M, Song S, Su H, et al. Shapenet: An information-rich 3d model repository. 2015. arXiv preprint arXiv:1512.03012.
30. Yuksel C. Sample elimination for generating poisson disk sample sets. Comput Gr Forum. 2015;34:25–32.
31. Kazhdan M, Hoppe H. Screened poisson surface reconstruction. ACM Trans Gr (ToG). 2013;32(3):1–13.
32. Wu S, Huang H, Gong M, Zwicker M, Cohen-Or D. Deep points consolidation. ACM Trans Gr (ToG). 2015;34(6):1–13.
33. Narayanan A, Chandramohan M, Venkatesan R, Chen L, Liu Y, Jaiswal S. graph2vec: Learning distributed representations of graphs. 2017. arXiv preprint arXiv:1707.05005.

Jiang *et al. BMC Bioinformatics*   (2023) 24:366

Page 19 of 19

Publisher's note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.