

Reducing Shape-Graph Complexity with Application to Classification of Retinal Blood Vessels and Neurons

Benjamin Beaudett^{1*} and Anuj Srivastava¹

^{1*}Statistics Department, Florida State University, 117 N. Woodward Ave, Tallahassee, 32306, Florida, USA.

*Corresponding author(s). E-mail(s): bbeaudett@fsu.edu;
Contributing authors: anuj@stat.fsu.edu;

Abstract

Shape graphs are complex geometrical structures commonly found in biological and anatomical systems. A shape graph is a collection of nodes, some connected by curvilinear edges with arbitrary shapes. Their high complexity stems from the large number of nodes and edges and the complex shapes of edges. With an eye for statistical analysis, one seeks low-complexity representations that retain as much of the global structures of the original shape graphs as possible. This paper develops a framework for reducing graph complexity using hierarchical clustering procedures that replace groups of nodes and edges with their simpler representatives. It demonstrates this framework using graphs of retinal blood vessels in two dimensions and neurons in three dimensions. The paper also presents experiments on classifications of shape graphs using progressively reduced levels of graph complexity. The accuracy of disease detection in retinal blood vessels drops quickly when the complexity is reduced, with accuracy loss particularly associated with discarding terminal edges. Accuracy in identifying neural cell types remains stable with complexity reduction.

Keywords: Shape graph, Graph simplification, Graph summarization, Elastic shape analysis

1 Introduction

This paper studies structures of shape graphs with a focus on their statistical analysis and classification. A shape graph is a set of curves in space (edges) along with the points where they intersect or terminate (nodes). Shape graphs generalize traditional graphs by assigning locations in space to each node and specifying paths in space for each edge instead of just binary adjacency values or scalar weights. In other words, one is also interested in the shapes formed by the edge curves when analyzing and comparing shape graphs. Shape graphs contain important information in the context of objects such as networks of

blood vessels or neurons, branches and roots of plants, and networks of roads or waterways. Spatial tree graphs that prohibit cycles are a special case of shape graphs.

In natural systems, the roles and functionalities of network structures are directly reflected in the details of connectivity and spatial relations that shape graphs capture. Thus, tools for statistical analysis of shape graphs provide a novel avenue for characterizing their roles or detecting irregularities. One case of interest is the retinal blood vessel (RBV) networks of the human eye. A great deal of work has been done in recent years to create machine learning (ML) algorithms that can extract these networks from retinal fundus

images and diagnose ocular diseases [7, 25, 5, 12, 16, 23]. These systems can either assist human diagnosticians or perform independent diagnoses, thereby reducing the cost of screening for diseases. Several of these systems are already in clinical use, including at least one approved by the Food and Drug Administration [16]. A common approach in these ML classifiers is to feed color fundus images directly to an artificial neural network which makes classifications based on pixel intensities. While some of these produce very accurate results, details about the decision-making are opaque and do not provide insight as to why a classification was made. Instead of relying on pixel intensities in the fundus images, we focus on the networks of veins and arteries, represented as shape graphs, and study their structural variability. This approach allows us to seek interpretable statistics of geometric details and use these for analysis.

A significant challenge in the analysis of shape graphs comes from their complexity. Direct comparison of any two graphs requires the registration of nodes across graphs, termed graph matching. In the context of retinal blood vessels, graph matching can be used to track changes over time [19] or to quantify the dissimilarity between subjects [3]. Current approaches to graph matching have computational complexity that grows as a third-order polynomial in the numbers of nodes and edges [39], making it difficult to quantify shape differences when the numbers of nodes and edges become even modestly large. One way to facilitate the matching is by reducing graph complexity, i.e. producing simpler graphs with fewer nodes and edges but which maintain the gross structure of the originals. This relieves the computational burden and causes the matching to focus on large-scale structures without being slowed by finer details. One can also improve results of matching graphs with differing complexity levels by simplifying the more complex graph to match the other [3]. A related issue is the level of detail in graphs necessary for classification. Do the smaller, peripheral structures carry relevant information, or is it mainly restricted to the core skeletal parts? Graph simplification can also be used to investigate this question.

A number of methods for graph simplification have been developed, with a recent survey provided by Liu et al. [18]. However, very little work has been done that applies to graphs where

both nodes and edges have spatial attributes. The only such method we are aware of is that of Basu Bal et al. [3], which clusters nodes and replaces each cluster with a single new node. That purely clustering-based approach gave some intuitive results but left visible room for improvement, having a tendency to create bushy collections of terminal edges emanating from a common node. In this paper we refine that approach, presenting a method for simplifying shape graphs that performs separate hierarchical clusterings of both nodes and edges as well as directly removing small or redundant terminal edges. The clustered nodes and edges are replaced by sparser representations that are based on means of the cluster members. The choice of metrics for clustering is important and dictates the outcomes.

We assess our method through visualizations and statistics-based classification experiments. These use shape graphs in \mathbb{R}^2 extracted from retinal fundus photographs and shape graphs in \mathbb{R}^3 formed from digital reconstructions of neurons. We lay out the approach used for shape graph reduction in sections 2, 3, and 4, including demonstrations using RBV and neuron graphs in section 4. In section 5 we discuss some alternative ideas. We describe and give results from classification experiments in section 6.

2 Mathematical Background

Shape Graphs: We define a shape graph G in \mathbb{R}^d as a set of nodes V represented by points $\{v_i \in \mathbb{R}^d\}_{i=1}^n$ and a set of edges B represented by parameterized curves $\{\beta_{ij} : [0, 1] \rightarrow \mathbb{R}^d\}_{i,j=1}^n$ with $\beta_{ij}(0) = v_i$ and $\beta_{ij}(1) = v_j$ for adjacent nodes or $\beta_{ij} = 0$ for nonadjacent nodes. We use the letter m to indicate the number of nontrivial edges in a graph and write β with a single index or none at all when the nodes are not being emphasized. We ensure that all graphs we work with consist of a single connected component. For most of this paper there are no other topological constraints. We refer to the special case where cycles are prohibited as tree graphs.

Curve Shape Means: To analyze shapes of edge curves, we utilize elements of elastic shape analysis of Euclidean curves. We briefly describe the approach here and refer to reader to [13, 15, 29, 26, 33, 34, 3, 11] for further details. This

approach is based on a mathematical representation called square-root velocity function or *srvf*. The *srvf* of an edge β is defined by an operator $q : \mathcal{C}([0, 1], \mathbb{R}^d) \rightarrow \mathbb{L}^2([0, 1], \mathbb{R}^d)$ with

$$q_\beta(t) := \dot{\beta}(t) / \sqrt{\|\dot{\beta}(t)\|}.$$

Here $\mathcal{C}([0, 1], \mathbb{R}^d)$ represents the set of all absolutely continuous, \mathbb{R}^d -valued functions on $[0, 1]$. Working with the derivative $\dot{\beta}$ immediately removes irrelevant information about location, and using the *srvf* has the additional advantage of providing a notion of shape distance between edges which is invariant to reparameterization. Let Γ be the group of orientation-preserving diffeomorphisms of $[0, 1]$. Reparameterization of an edge by some $\gamma \in \Gamma$ acts on the *srvf* as

$$(q_\beta, \gamma)(t) := q_{\beta \circ \gamma}(t) = q_\beta(\gamma(t)) \sqrt{\dot{\gamma}(t)}.$$

A key property of *srvf* representation is the invariance $\|(q_\beta, \gamma)\|_2 = \|q_\beta\|_2 = \sqrt{\ell}$, the square root of the arc length of β , for any $\gamma \in \Gamma$. Changing the subscript notation to reflect indices, transforming two *srvfs* using the same reparameterization yields an unchanged $\|(q_1, \gamma) - (q_2, \gamma)\|_2 = \|q_1 - q_2\|_2$. Next, one defines a quotient space in the *srvf* space according to

$$[q] := \{(q, \gamma) | \gamma \in \Gamma\}.$$

The quotient point $[q_\beta]$ represents the set of all curves that are within translations and reparameterizations of β , and constitutes a shape equivalence class. The set of all equivalence classes is the shape space $\mathcal{S} \equiv \mathbb{L}^2([0, 1], \mathbb{R}^d) / \Gamma$. This shape space is endowed a metric d_s according to

$$d_s([q_1], [q_2]) := \min_{\gamma \in \Gamma} \|q_1 - (q_2, \gamma)\|_2.$$

This d_s is a proper distance on \mathcal{S} and is invariant to both rigid translation and reparameterization of curves [27]. The optimizing γ can be approximated using the dynamic programming algorithm.

We can now define a representative shape for a set of *srvfs* $\{q_i\}_{i=1}^I$ as the Karcher or intrinsic mean

$$[\mu_q] := \underset{[q]}{\operatorname{argmin}} \sum_{i=1}^I d_s([q], [q_i])^2.$$

We find a value μ_q using a slight alteration of the *srvf* Karcher mean algorithm presented in [28]. After selecting one of the q_i as an initial μ_q^1 , the algorithm proceeds iteratively, at each step k reparameterizing the *srvfs* as $\{q_i^k\}_{i=1}^I$ to optimally match μ_q^k and updating $\mu_q^{k+1} = \frac{1}{I} \sum_{i=1}^I q_i^k$. This algorithm is known to converge with monotonically decreasing error, but it can converge to a local optimum rather than a true mean. We use the μ_q produced by the algorithm without further reparameterization.

Having found a mean *srvf* μ_q for a set of edges $\{\beta_i\}_{i=1}^I$, an unpositioned mean edge is formed by converting it into a curve by integrating $\beta_\mu(t) := \int_0^t \mu_q(s) \|\mu_q(s)\| ds$. In our applications we use edge sets $\{\beta_i\}_{i=1}^I$ to form a mean edge which always has a prespecified pair of destination nodes v_a and v_b it is intended to connect. In doing this, we first form a new set of edges $\{\beta_i^*\}_{i=1}^I$ by rotating, translating, and scaling the originals so that they run from v_a to v_b . These repositioned curves are then used to define β_μ which is in turn rotated, translated, and scaled to fit the destination node pair.

Chamfer Distance: Some steps in our method require measurements of location-based discrepancy between edges, for which the shape distance above cannot be applied. In these cases we use the chamfer distance as defined in [37]. In our setting, an edge β_k is represented by T consecutive points $\{p_i^{(k)}\}_{i=1}^T$. The chamfer distance between two edges is

$$d_C(\beta_1, \beta_2) = \frac{1}{T} \sum_{i=1}^T \min_{p \in \beta_2} \|p_i^{(1)} - p\| + \frac{1}{T} \sum_{i=1}^T \min_{p \in \beta_1} \|p_i^{(2)} - p\|. \quad (1)$$

We note that this is not a proper distance since it does not obey the triangle inequality.

Effective Resistance: Our node clustering method requires a distance between nodes that is attentive to the network structure. We use the effective resistance defined in [8]. Rather than simply finding the shortest edge-based paths between nodes, this distance measures nodes as being progressively closer as the overall connectivity between them is increased. Take a graph composed of nodes $\{v_i\}_{i=1}^n$ and edges $\{\beta_{ij}\}_{i,j=1}^n$ between them. Let ℓ_{ij} be the length of β_{ij} and define the weight $w_{ij} = \frac{1}{\ell_{ij}}$ if nodes v_i, v_j are adjacent and $w_{ij} = 0$ otherwise. Let \mathbf{W} be the $n \times n$ matrix of weights, \mathbf{S} be a diagonal matrix of the strengths

$S_{ii} = \sum_{j=1}^n w_{ij}$, and \mathbf{Q} be the weighted Laplacian $\mathbf{Q} = \mathbf{S} - \mathbf{W}$. For a connected graph, this matrix is positive semidefinite with a single zero eigenvalue which has the equiangular vector $\mathbf{1}$ as its eigenvector [14]. The effective resistance between two nodes is then defined as

$$d_E(v_i, v_j) = (\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{Q}^{-1} (\mathbf{e}_i - \mathbf{e}_j)^T \quad (2)$$

where \mathbf{e}_i is the column vector with 1 in its i th location and zeros elsewhere, and \mathbf{Q}^{-1} is a matrix which operates as an inverse of \mathbf{Q} on $\text{span}\{\mathbf{1}\}^\perp$ and as the zero map on $\text{span}\{\mathbf{1}\}$. This d_E is a proper distance function [14].

3 Data Processing

There are two types of datasets used in this paper: retinal blood vessels (RBVs) and neurons. We convert input data into shape graphs where each edge is discretized as a piecewise linear curve defined with arbitrary granularity. The results presented here used $T = 30$ points per edge.

The RBV shape graphs are extracted from retinal fundus photographs. The photographs are first converted to binary segmentations where each pixel is either black or white to indicate whether a vessel is present in that location. This segmentation was performed prior to our accessing the data. We then use the Vessel Tech software [36] to convert the binary segmentations to shape graphs in \mathbb{R}^2 . The nodes of the graph are the points where vessel branches intersect or terminate. The edges connecting the nodes are the stretches of blood vessel between them. We do not distinguish between veins and arteries, and the representation does not consider continuity of vessel segments across intersection points. We do not consider vessel thickness.

The neuron shape graphs are obtained directly from `.swc` encodings of neuron structures in \mathbb{R}^3 .

Our approach assumes that input graphs consist of a single connected component. This is a reasonable assumption for our data types. However, imperfections in the data gathering and processing can result in disconnected graphs, so we apply a simple method of automatically connecting the graphs: A connected component \tilde{G}^i of a disconnected graph \tilde{G} is represented as the union of all of the points in its nodes and edges. The smallest Euclidean distance is found between any

point $p \in \tilde{G}^i$ and any point $q \in \tilde{G}^j \subseteq (\tilde{G}^i)^c$ in any other component, such that at least one of p and q is a node. Components \tilde{G}^i and \tilde{G}^j are then joined by creating an edge between nodes at points p and q , creating a new node at one of the points if necessary. The shape of the new edge is determined using the Karcher mean of *srufs* of nearby edges using the method described in section 2. This is repeated selecting a new \tilde{G}^i from the redefined components until a single connected graph G is produced.

An example of a binary segmented fundus image and the connected graph extracted from it can be found in the top two images in the leftmost column of Fig. 8. The leftmost column of Fig. 9 shows three views of a neuron shape graph with the full complexity of its original encoding. Our visualizations use human RBV data taken from the ARIA dataset [9, 10] and human neurons from the Narkilahti dataset [24] on NeuroMorpho.org [20, 2].

4 Multi-Resolution Shape Graph Representation

In this section we describe a method that reduces graph complexities while maintaining their general shapes. We use the term ‘resolution’ to refer to the complexity. Thus, representing a graph at multiple resolutions by progressively decreasing its complexity leads to a ‘multi-resolution’ representation. The process runs as a cycle consisting of two kinds of steps: (1) Terminal edge removal: Terminal edges identified as very short or as similar to nearby terminal edges do not contribute much to fundamental structure and are removed. This helps prevent formation of bushy groups of terminal edges that was seen in earlier work. (2) Local summarization via clustering: Separate hierarchical clustering steps for nodes and edges identify clusters throughout the graph and replace them with simpler structures. It is common in our data to find long, unbroken edge segments running alongside each other which are redundant in terms of the skeletal shape of the graph. Including edge-based clustering allows the algorithm to identify and merge these in cases where node clustering alone would leave them unchanged.

All of the trimming and clustering steps, as well as the graph connection in section 3, can

produce degree-two nodes. These can often be removed without affecting the graph’s spatial structure by simply deleting the node and concatenating the two adjacent edges. In some cases though, the two adjacent nodes are themselves adjacent, forming a ‘triangle’ such that removing the degree-two node would leave the others directly connected by two separate edges. We remove the former type as they appear throughout the process, but leave the latter unchanged. For conciseness, we usually omit mention of these removals in the details below.

All of our hierarchical clustering is performed using the native `linkage` and `cluster` functions in MATLAB [31].

4.1 Terminal Edge Removal

We consider two types of terminal edges that can be removed with only minor changes in graph structure: short terminal edges and groups of terminal edges that are similar to each other in terms of shape and location. We perform two separate processes to remove these kinds of edges from a graph. The first uses information about the graph to define a length threshold for a short edge then removes terminal edges that are shorter. The second uses clustering to identify groups of similar terminal edges then removes all but the longest from each group. We outline these processes in algorithms 1 and 2. For the similar terminal removal we use the chamfer distance d_C from equation 1 and single-linkage (shortest distance) hierarchical clustering. Whenever a round of terminal trimming is applied, it includes short terminal removal followed by similar terminal removal. Application of the terminal removal steps is illustrated in Figs. 1 and 2.

4.2 Node Clustering

The node-based clustering performed here is the same as used in [3]. We outline the process in algorithm 3. For distance we use the effective resistance d_E from equation 2, and our distance-based clustering method is complete-linkage (largest distance) hierarchical clustering. Figs. 3 and 4 illustrate the node clustering produced by this method.

Algorithm 1 Short terminal edge trimming

- 1: **Input:** A shape graph with n edges, and tuning parameters θ_{tag} and θ_{til} .
 - 2: Compute the lengths of all n edges in the graph.
 - 3: Set $\ell_{\theta_{til}}$ as the θ_{til} percentile of the lengths.
 - 4: Set a length threshold $\ell_\theta = \theta_{tag} * \ell_{\theta_{til}}$ as a percentage of the percentile.
 - 5: **while** the graph contains terminal edges with length less than ℓ_θ :
 - 6: Remove terminal edges with length less than ℓ_θ along with their terminal nodes. Removing terminal edges redefines which remaining edges are terminal, so this must be done repeatedly until the condition is met. Also, removing a terminal edge often reduces the degree of a node from three to two. In order to avoid removing chains of short terminal edges which are more properly considered a single long one, degree-two nodes are detected and elided as appropriate between rounds of edge removal.
 - 7: **end while**
 - 8: **Output:** A shape graph with short terminal edges and associated nodes removed, and all other nodes and edges identical to the input.
-

4.3 Edge Clustering

The method used for edge-based clustering is similar to that used for the nodes, with the distinction that an edge cluster is not directly replaced by an edge but rather is replaced by a node. Thus note that while the clustering uses a target number m_r of edge clusters, this value is more directly associated with the number of nodes in the output graph than the number of edges. We outline the process in algorithm 4. For distance we use the chamfer distance d_C from equation 1, and our distance-based clustering method is average-linkage (arithmetic mean) hierarchical clustering. Figs. 5, 6, and 7 illustrate the edge clustering produced by this method.

4.4 The Multi-resolution Reduction Cycle

Through exploratory experiments and visual assessment of results of the resolution reduction methods described above, we noticed two trends: First, applying resolution reduction as a sequence

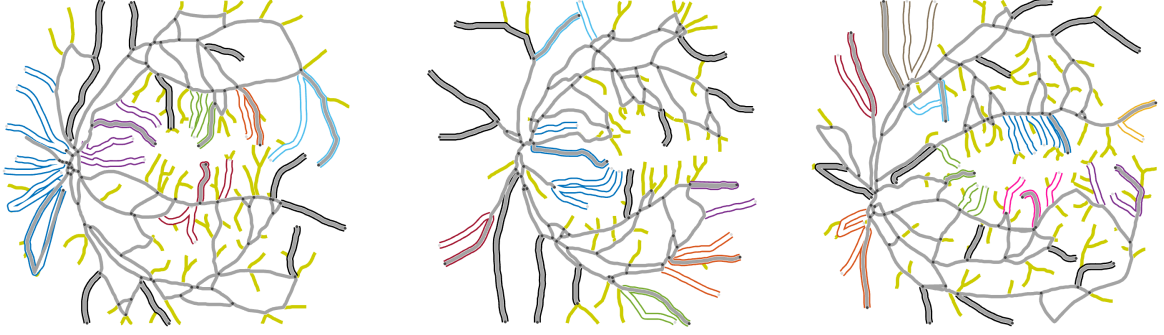


Fig. 1: Terminal edge removal applied to three RBV networks. Yellow edges are removed during short-terminal trimming. This is followed by similar-terminal trimming. Colored highlights indicate similar-terminal clusters. Black highlights indicate terminal edges not assigned to any cluster. Only the longest edge in each cluster is retained, with the white edges being removed. Gray edges show the graph that remains after trimming.

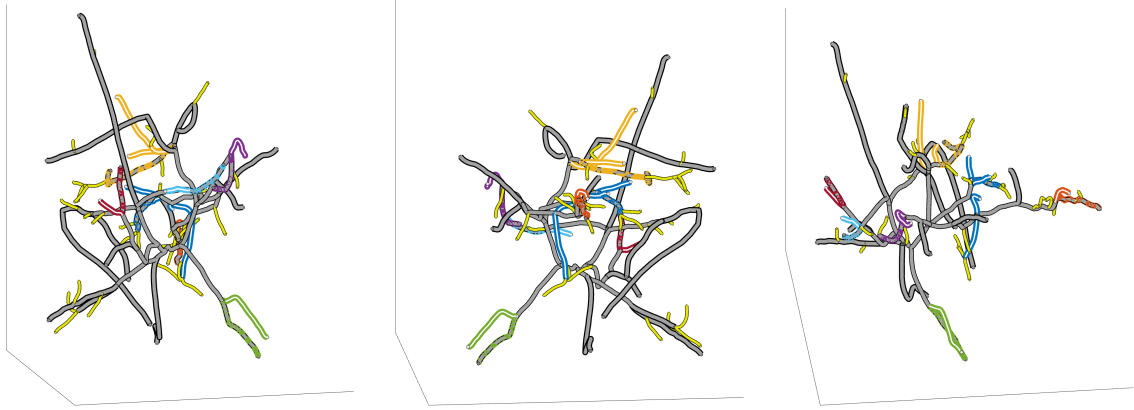


Fig. 2: Terminal edge removal applied to a neuron. Three views of the same graph. Bright yellow edges are removed during short-terminal trimming. This is followed by similar-terminal trimming. Colored highlights indicate similar-terminal clusters. Thicker black highlights indicate terminal edges not assigned to any cluster. Only the longest edge in each cluster is retained, with the white edges being removed. Gray edges show the graph that remains after trimming.

of terminal edge trimming, edge clustering, node clustering, and a further terminal trimming in that order was effective in producing skeletal reductions at the target complexity. Second, when the degree of complexity reduction was large, the skeletal structure was better maintained if the reduction was performed as a sequence of gradual reductions as opposed to fully dropping the resolution in a single step. In regard to the first observation, we note that while the node clustering step is guaranteed to produce a reduced graph with no more than the number n_r of target nodes, the edge clustering reduction generally produces

graphs with somewhat more nodes than the number m_r of target clusters, and can possibly even produce graphs larger than the input. While this would apparently be a flaw if using edge clustering as a standalone reduction method, it is a useful feature when used in conjunction with the node clustering. The edge clustering finds collections of similar edges and replaces them with highly-interconnected nodes which are then condensed by the node clustering, which enforces the desired node count in the reduced graph. We found that using edge clustering followed by node clustering

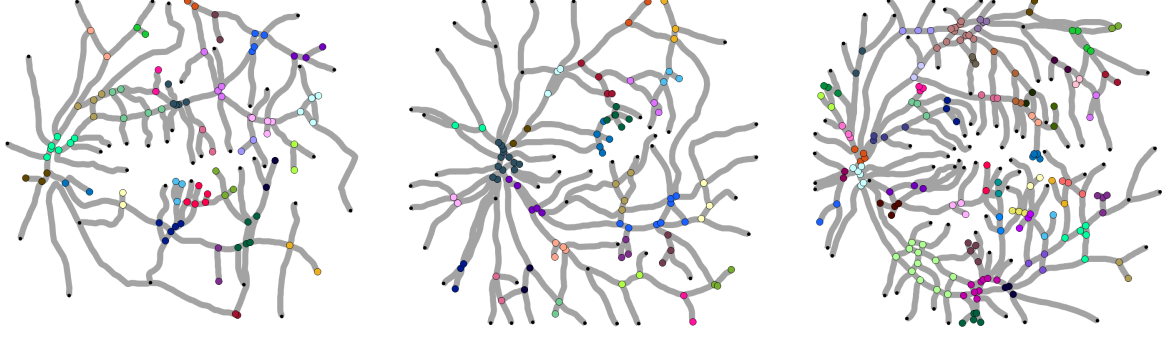


Fig. 3: Node clusters identified in three RBV networks using a .5 resolution target $n_r = 0.5n$. Nodes marked with the same color form a cluster. Smaller black nodes are not assigned to any cluster. The reduced graphs are not shown in this figure.

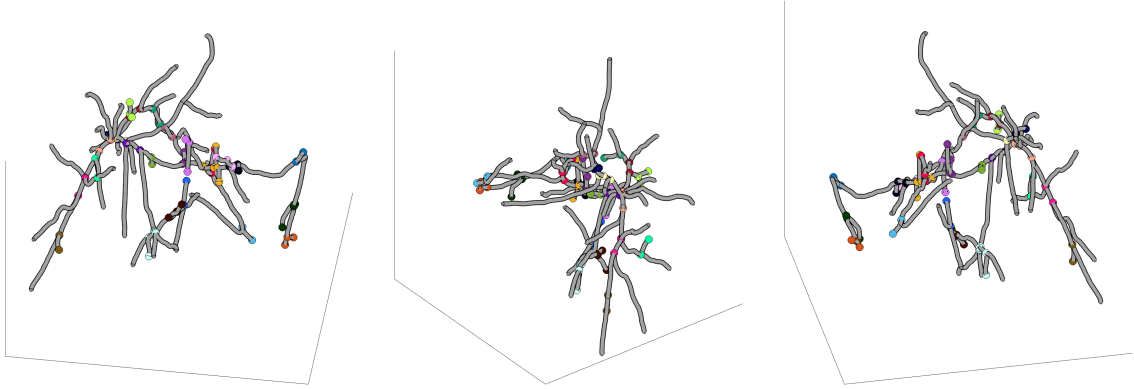


Fig. 4: Node clusters identified in a neuron using a .5 resolution target $n_r = 0.5n$. Three views of the same graph. Nodes marked with the same color form a cluster. Smaller gray nodes are not assigned to any cluster. The reduced graph is not shown in this figure.

gave more visually convincing results than using node clustering alone.

We perform complexity reduction using a multi-resolution sequence. This consists of performing edge clustering, node clustering, and terminal edge trimming cyclically to produce reduced graphs at progressively decreasing resolution levels $\rho_0 = 1 > \rho_r > \rho_{r+1} > 0$. The complete process of producing a multi-resolution reduction of a shape graph using a set of resolutions $\{\rho_r\}_{r=0}^R$ is detailed below, summarized in algorithm 5, and illustrated in Figs. 8 and 9.

The input shape graph G first undergoes a preliminary round of terminal edge removal, producing a graph G_0 . The numbers m_0 and n_0 of

nontrivial edges and nodes in G_0 are used to determine edge cluster targets $\{m_r = \lceil \rho_r * m_0 \rceil\}_{r=1}^R$ and node count targets $\{n_r = \lceil \rho_r * n_0 \rceil\}_{r=1}^R$ for the clustering steps at each resolution level. Then the resolution reduction sequence is performed for each resolution ρ_r , $r \geq 1$. First, edge clustering is performed on G_{r-1} with an edge cluster target of m_r to produce \tilde{G}_r^{edge} . Next, node clustering is performed on \tilde{G}_r^{edge} with a node count target of n_r to produce \tilde{G}_r^{node} . The sequence is completed by applying terminal edge trimming to \tilde{G}_r^{node} to produce G_r , the reduced-complexity graph for that resolution.

The graph produced at a given reduced resolution is dependent on the sequence of resolutions included in the process. For example, a graph produced by reducing directly from resolution 1.0 to

Algorithm 2 Similar terminal edge trimming

- 1: **Input:** A shape graph with m edges, a tuning parameter ϕ_{til} , a distance function defined on shape graph edges, and a distance-based clustering method that determines the number of clusters based on the data and can restrict cluster formation according to a distance ceiling.
 - 2: Compute the distances between all m edges in the graph.
 - 3: Set a maximum cluster radius rad_ϕ equal to the ϕ_{til} percentile of the distances.
 - 4: Perform clustering on only the terminal edges. Clustering is performed to the extent possible without allowing cluster formation at distances above rad_ϕ .
 - 5: Identify the longest edge in each cluster (including singletons) and remove all other terminal edges along with their terminal nodes.
 - 6: **Output:** A shape graph with visually redundant terminal edges and associated nodes removed, and all other nodes and edges identical to the input.
-

0.6 will generally differ from the one produced by reducing the resolution progressively from 1.0 to 0.8 to 0.6. For our data, we found that progressive reduction using resolution step sizes of 0.2 usually produced satisfactory results for reduction down to resolution 0.4 or lower.

5 Other Reduction Methods

There is no general best approach for simplifying a graph. This is especially true for shape graphs with curve-valued edge attributes. We investigated a variety of methods for reducing complexity, with the one exhibited in most of this paper being the one we preferred. In this section we describe some others we tried.

5.1 k -means Clustering

Rather than using the greedy approach of hierarchical clustering, one might like to have a method that is guided by a global optimization objective.

¹ A family of such methods is offered by k -means

¹Although methods of hierarchical clustering defined by optimality conditions have been devised, most work to date on

Algorithm 3 Node-clustering reduction of a shape graph

- 1: **Input:** A shape graph with n nodes, a target reduced node count $n_r < n$, a distance function defined on shape graph nodes, and a choice of distance-based clustering method that allows the user to specify the number of clusters produced.
 - 2: Compute the distances between all n nodes in the graph.
 - 3: Perform clustering on the nodes such that the number of multiple-node clusters plus the number of singletons is equal to n_r . These two types are treated equivalently as clusters.
 - 4: For each cluster c_i , form a new node v_i^* located at the Euclidean mean position of the nodes in the cluster.
 - 5: Determine adjacencies between the new nodes. For each cluster c_i , the new node v_i^* representing it is made adjacent to all nodes v_j^* representing clusters c_j for which there was any prior adjacency between some nodes $v_i \in c_i$ and $v_j \in c_j$.
 - 6: Form edges between adjacent new nodes v_i^* and v_j^* using the *srvf* Karcher mean described in section 2. The mean shape is defined using all edges connecting a node in c_i to a node in c_j , and v_i^* and v_j^* are the destination nodes.
 - 7: **Output:** A shape graph defined by the new nodes $\{v_i^*\}_{i=1}^{n_r}$ and the edges connecting them.
-

clustering. Given some dissimilarity measure on point sets, our objective is to produce a graph with k nodes such that the sum of the dissimilarities between the edges in the input graph and their corresponding nodes in the reduced graph is minimized. The reduction is performed as follows: To begin, k points in \mathbb{R}^d within a region around the input graph are randomly chosen as initial cluster centers. Viewing the discretized edges as point clouds, dissimilarities are computed between each edge and each center. Each edge is assigned to the least dissimilar center. A new center for each of the resulting edge clusters is defined by finding the single point in the nearby region which produces the minimum sum of dissimilarities within its cluster. The reassign and re-center steps are

the topic has focused on applying algorithms without attention to mathematically formalized objectives [6].

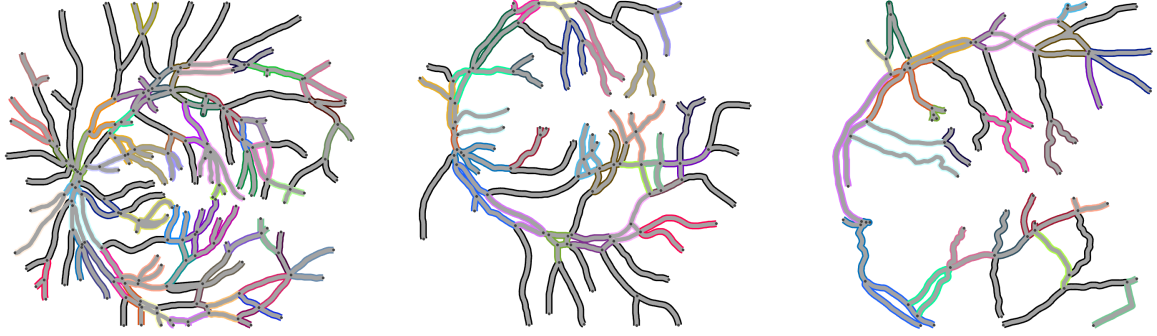


Fig. 5: Edge clusters identified in three RBV networks using a .4 resolution target $m_r = 0.4m$. Edges marked with the same color form a cluster. Black highlights indicate edges not assigned to any cluster. The reduced graphs are not shown in this figure.

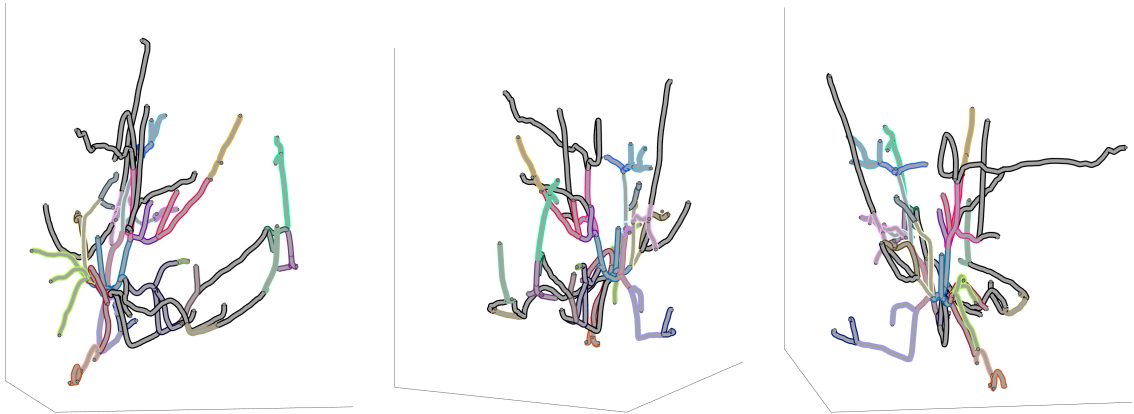


Fig. 6: Edge clusters identified in a neuron using a .4 resolution target $m_r = 0.4m$. Three views of the same graph. Edges marked with the same color form a cluster. Black highlights indicate edges not assigned to any cluster. The reduced graph is not shown in this figure.

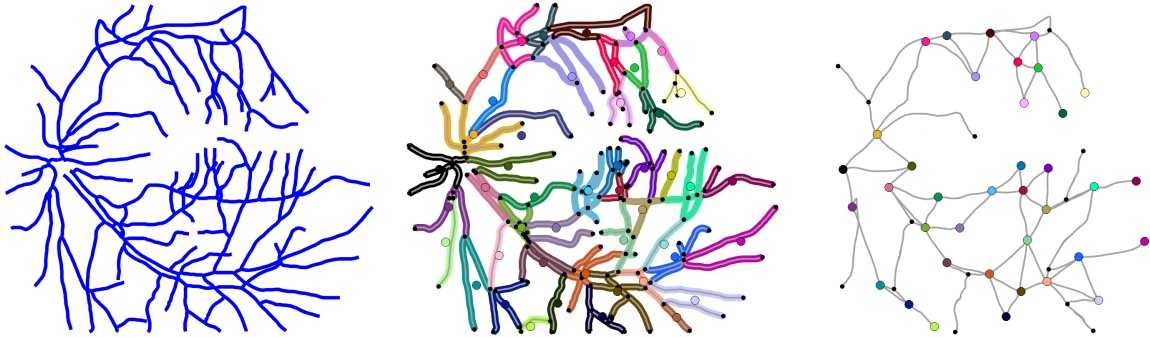


Fig. 7: Illustration of the edge clustering reduction step applied to a single RBV network. *Left:* The unreduced graph. *Center:* Edge clustering of the unreduced graph. Colored highlights indicate identified clusters. Colored dots indicate the node mean for the corresponding cluster. *Right:* The reduced graph produced by the edge clustering step, used as an intermediate step in the reduction cycle. Colored dots are inherited from the center plot and indicate nodes that have been created to represent edge clusters. Smaller black dots are nodes retained from singleton edges.

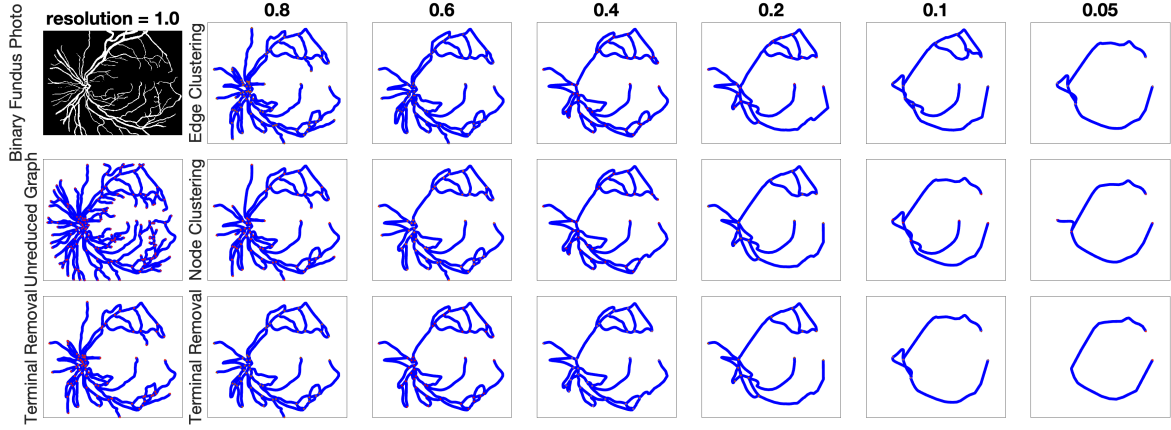


Fig. 8: The multi-resolution reduction process applied to one RBV network using resolutions $\rho_r \in \{1, .8, .6, .4, .2, .1, .05\}$, including intermediate steps. Columns progress from left to right and rows within each column progress from top to bottom. *First column:* Binary segmented retinal fundus photograph (top), connected graph G extracted from it (middle), and G_0 produced by preliminary trimming of terminal edges (bottom). *Columns with resolution $\rho_r < 1$:* Graphs \tilde{G}_r^{edge} (top), \tilde{G}_r^{node} (middle), and G_r (bottom) produced by sequential edge clustering, node clustering, and terminal trimming.

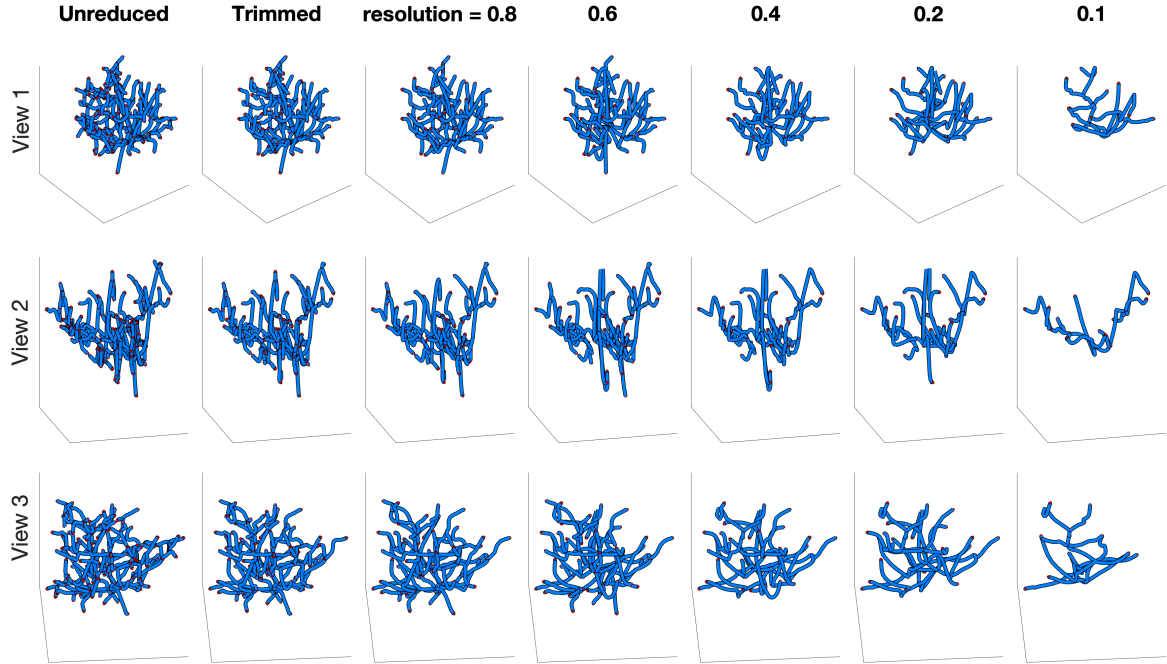


Fig. 9: The multi-resolution reduction process applied to a neuron using resolutions $\rho_r \in \{1, .8, .6, .4, .2, .1\}$. Each column shows three views of the same graph G_r for the indicated resolution level. The intermediate substeps of the reductions are not displayed in this figure.

Algorithm 4 Edge-clustering reduction of a shape graph

- 1: **Input:** A shape graph with m edges, a target number of edge clusters $m_r < m$, a distance function defined on shape graph edges, and a choice of distance-based clustering method that allows the user to specify the number of clusters produced.
 - 2: Compute the distances between all m edges in the graph.
 - 3: Perform clustering on all edges in the graph such that the number of true multiple-edge clusters plus the number of singletons is equal to m_r . The singletons are treated differently from the true clusters in what follows.
 - 4: For each singleton edge, retain its two endpoints as nodes v_i^* connected by the original edge without change.
 - 5: For each true edge cluster c_i , form a new node v_i^* located at the Euclidean mean position of all unique nodes that served as an endpoint for any of the edges in the cluster.
 - 6: Determine adjacencies involving newly-defined nodes. For new nodes v_i^* and v_j^* both of which are associated with true clusters, if there is any node v_{ij} in the input graph which serves as an endpoint for an edge in c_i and an edge in c_j , the new nodes are made adjacent. If v_i^* is retained from a singleton and v_j^* is formed from a true cluster, they are made adjacent if there is any edge connected to v_i which is a member of c_j .
 - 7: Form edges associated with newly-defined nodes. Each new edge is an *sruf* Karcher mean as described in section 2. When both v_i^* and v_j^* are defined from true clusters, the mean edge shape is defined using all edges in c_i and c_j connected to any node v_{ij} which serves as an endpoint to at least one edge in each cluster. If v_i^* is retained from a singleton and v_j^* is formed from a true cluster, the mean edge shape is defined using the singleton edge and all edges in c_j which have v_i as an endpoint. In both cases, v_i^* and v_j^* are the destination nodes.
 - 8: **Output:** A shape graph defined by the new and retained nodes $\{v_i^*\}_{i=1}^{\tilde{m}_r}$ and the edges connecting them, where \tilde{m}_r is equal to the number of true edge clusters plus the number of nodes retained from singleton edges.
-

Algorithm 5 Multi-resolution reduction of a shape graph

- 1: **Input:** Connected graph G , resolutions $\{\rho_r\}_{r=1}^R$, and trimming parameters $\theta_{tag}, \theta_{til}$, and ϕ_{til} .
 - 2: Trim G to produce G_0 and collect its edge count m_0 and node count n_0 .
 - 3: Define edge count targets $\{m_r = \lceil \rho_r * m_0 \rceil\}_{r=1}^R$ and node count targets $\{n_r = \lceil \rho_r * n_0 \rceil\}_{r=1}^R$ for clustering.
 - 4: **for** r from 1 to R :
 - 5: Perform edge clustering on G_{r-1} to produce \tilde{G}_r^{edge} .
 - 6: Perform node clustering on \tilde{G}_r^{edge} to produce \tilde{G}_r^{node} .
 - 7: Perform terminal edge trimming on \tilde{G}_r^{node} to produce G_r .
 - 8: **end for**
 - 9: **Output:** Multi-resolution graph sequence $\{G_r\}_{r=0}^R$
-

repeated until the dissimilarity sum converges to a local minimum. The final cluster centers are used as a node set for a reduced graph, and the edge clusters are used to construct a new edge set as described in section 4.3.

We tried this approach with various dissimilarity methods including chamfer distance, maximum set distance, minimum set distance, and Euclidean distance between centroids. All of these were tested with and without using edge lengths as weights for the dissimilarities. We judged the results to be uniformly poor by visual assessment. Having edges cluster based on affinity to a common point rather than to each other creates groupings with some symmetry about that point. This means that edges on opposite sides of the cluster center, despite having only their endpoints close to each other, are frequently grouped together. This often resulted in clusters including edges from distinct branches of the graph, joining together sections that should have remained separate.

5.2 Pairwise Clustering

The results from k -means clustering highlighted the importance of direct edge-to-edge comparisons. With this in mind, we turned back to the cyclic process from section 4 but trying alternative

methods at the edge clustering step. This included a number of variations of the hierarchical clustering in section 4.3 as well as a probabilistic pairwise clustering method detailed in [30]. We outline the probabilistic method here.

The algorithm takes a dissimilarity matrix as input, and after performing a random initial clustering it proceeds by performing two kinds of steps. The first randomly selects one of the input edges and randomly assigns it to one of the existing clusters. The assignment is based on to a probability vector with weights for each cluster corresponding to what the sums of all pairwise dissimilarities within all clusters would be if the edge were assigned to that cluster. Assignments with lower dissimilarity sums are given higher weights. The second step randomly selects two edges from different clusters and randomly either swaps their cluster assignments or does nothing. The decision to swap is based on a two-element probability vector with weights determined by what the sum of all pairwise dissimilarity sums within all clusters would be if the swap were or were not performed. Performing both of these steps in sequence constitutes one round of the algorithm. In each of the two steps, the probability values are modulated by a temperature parameter that makes the probability vectors more uniform when the temperature is high and accentuates the higher probability values when the temperature is low. The temperature is decreased at each round of the algorithm, introducing an element of simulated annealing. The algorithm runs until completing a prespecified number of rounds (we used 12,000).

Given a clustering of a graph’s edges, we again took the approach of representing each edge cluster by a single node and connecting these nodes as in section 4.3. This still leaves the question of how to define these new nodes. In addition to using Euclidean means of preexisting nodes as before, we also tried a number of dissimilarity-based cluster centers as with the k -means approach, except here the centers were only computed after the clusters were already defined.

We tried both the hierarchical clustering and the probabilistic clustering using a variety of settings. They were run using dissimilarity measures of chamfer distance, maximum set difference, and Euclidean distance of centroids. Each of these were tried with new nodes defined using Euclidean

node means or edge dissimilarity centers. A number of these yielded visually satisfactory results. The hierarchical method with chamfer dissimilarity and Euclidean nodes that is the focus of this paper was among the best.

5.3 Edge Removal

A simple approach to reducing graph complexity would be to use a reductive process which works entirely by removing nodes and edges without ever creating new ones. This has the advantage that no unwanted features such as novel cycles can be introduced. The terminal edge trimming in section 4.1 meets this description and could be effective as a standalone method for simplifying tree graphs. However, using a method comprised exclusively of removals to simplify a graph where cycles are allowed presents some hurdles. In our data it was common to see cycles or series of cycles composed of edge segments running parallel to each other and joined by other edges bridging across them (see Figs. 1, 3, and 5 for reference). In these cases an ideal simplification would remove the cycles and represent the entire region with a single edge. Clearly, then, a removal algorithm must be allowed to consider non-terminal edges. Safeguards would need to be in place to prevent the graph from becoming disconnected. This would not be too difficult, but there is the more nuanced problem of how to select removals that leave a sensible graph behind. Removing edges from these cycles using length-based criteria would be prone to producing jagged zig-zag paths that jump back and forth across both sides of a gap, giving a poor representation of the underlying trend. A more sophisticated approach would be needed to deal with this kind of issue. We did not explore this direction any further, but feel it would be worthwhile to pursue.

5.4 Medial Axis

We also experimented with simplification using medial axes [17] of sets defined by a shape graph. The basic idea is to broaden the graph into a wider region and use the medial axis of the border of that region as a reduced graph. We begin by performing a distance transform of space relative to a graph G using the function $D_G : \mathbb{R}^d \rightarrow \mathbb{R}_{\geq 0}$ with $D_G(x) =$

$\min_{p \in G} d(x, p)$ where $d(\cdot, \cdot)$ is the Euclidean distance. This is used to define level sets $K_r = \{x \in \mathbb{R}^d | D_G(x) = r\}$. The medial axis \tilde{M}_r of K_r is defined by performing another distance transform $c_r(x) = \min_{q \in K_r} d(x, q)$ and the set-valued function $\theta_r(x) = \{q \in K_r | d(x, q) = c_r(x)\}$, then setting $\tilde{M}_r = \{x \in \mathbb{R}^d | \#\theta_r(x) \geq 2\}$. This set can contain both an interior subset where $D_G(x) < r$ and an exterior subset where $D_G(x) > r$. We are interested only in the interior portion, which we call M_r . A multi-resolution reduction of a shape graph can be produced by setting $G_0 = G$ and progressively simplifying $G_r = M_r$ as r increases.

We computed various approximations of M_r using the MATLAB functions `bwmorph` and `bwskel`. An example is shown in Fig. 10. Some of the results were compelling. For example, the graph in the top-right panel gives a convincing simplification of the original. Problems start to arise before long, though. The subsets of K_r around C-shaped curves in the graph eventually join across the gap and produce a medial axis with unwanted cycles, as in the passage from the top-right panel to the middle-left panel. The reverse topological change of removing cycles can also cause issues. As the circular subset of K_r within an existing cycle shrinks to a point and disappears, the medial axis collapses around it and suddenly jumps to pass through the center of the hole. This behavior is desirable for small-scale cycles, passing a single curve through the middle of a complex region as happens multiple times in the top row of the figure. When this happens in a large-scale cycle though, as in the two bottom-right panels, the medial axis departs entirely from the graph.

A few other problems arose which were less dramatic but still need consideration. In many cases, we found the medial axis edges to be significantly more angular and linear than the organic originals. Also, imperfections in the smoothness of the level sets frequently caused terminal edges to split at their ends and stretch beyond the original graph out to the level set boundary (this issue is not present in the figure).

In sum, this approach seems to hold some promise for moderate complexity reduction. The more serious topological issues don't arise until r becomes large, and smaller r values did produce some satisfying results. However, the value of r where things break down is specific to each individual graph and can vary widely. Defining criteria

to automatically determine an appropriate level of reduction could be difficult.

6 Image Classification

When using retinal images to detect disease, how much of the relevant information is carried by the fine details of the vessel networks, and how much can be found in the coarse large-scale structure alone? What is the relative importance of these two scales in recognizing retinal disease or distinguishing between neural cell types? We investigated these questions with classification experiments. To do this we reduced labeled graph datasets to a number of progressively lower resolutions, collected a set of graph statistics at each resolution level, and compared the accuracy of classification tasks performed using the statistics collected at different resolutions. Informally, this might be thought of as a search for a border between signal and noise in the granularity of detail for a given data set. Steady classification rates in the presence of decreasing resolution would be interpreted to mean that details at that level of complexity are essentially noise, whereas sudden drops in classification rates would indicate that signal has been removed.

6.1 Features Used for Classification

The classification was performed using easily-interpretable statistics of the nodes and edges of each graph. The node-based statistics are counts of nodes of various degrees and the overall total. The edge-based statistics used percentiles, averages, and totals of edge length, average curvature, and tortuosity computed for each edge. Table 1 defines values used for computing the edge-based statistics, and Table 2 lists all of the statistics considered for each graph. Fig. 11 gives an example of statistics collected for 30 individual RBV graphs. We performed our classification experiments twice: first using all 37 statistics listed in table 2, and again separately using only the first 17, excluding average curvature and tortuosity.

6.2 Data and Classification Methods

For the classification experiments using retinal blood vessels, we separately assessed small data sets from two sources: the Automated Retinal

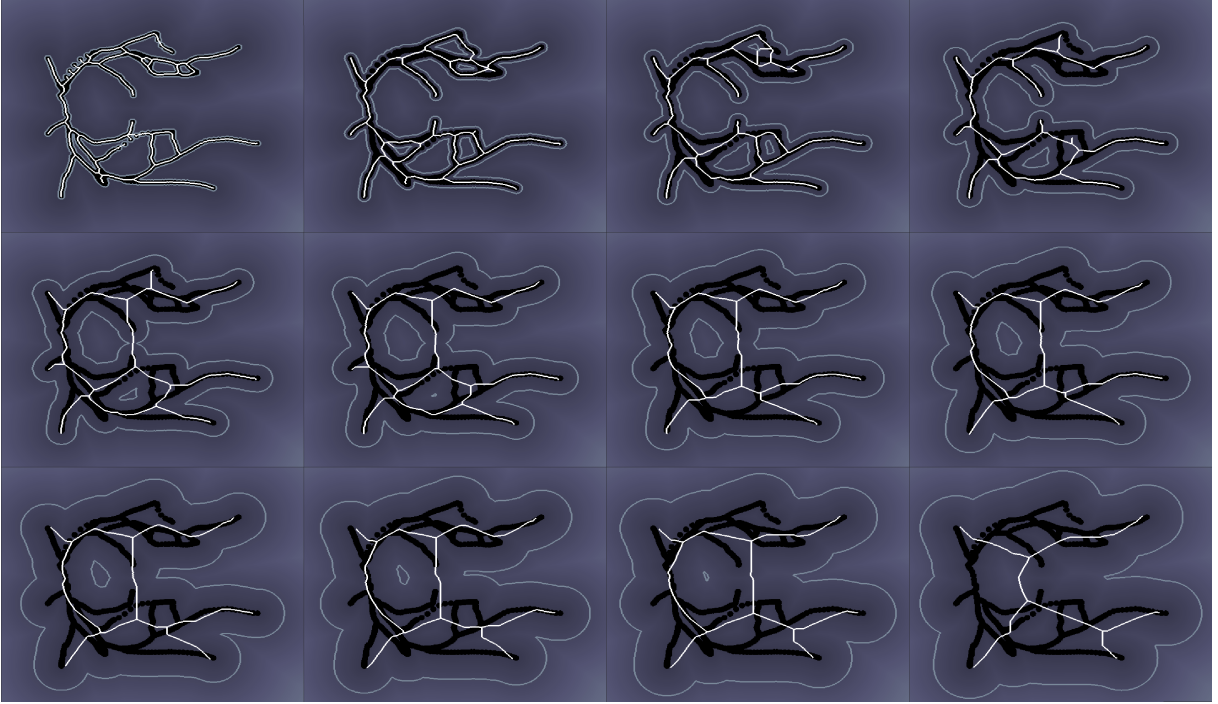


Fig. 10: Multi-resolution reduction using medial axes. Top left uses level set $r = 10$ and progresses from left to right across rows until $r = 110$ at bottom right. Black shows input graph. Grayscale background shows distance transform $D_G(x)$. Thin white and thick white respectively show level sets K_r and medial axes M_r .

Edge in a graph	$\beta : [0, 1] \rightarrow \mathbb{R}^d$
Length of an edge	$\ell = \int_0^1 \ \dot{\beta}(t)\ dt$
Curvature at a point along an edge	$\kappa = \left\ \frac{dT}{ds} \right\ $ (unit tangent vector T and arc length s)
Average curvature of an edge	$\frac{1}{\ell} \int_0^\ell \kappa(s) ds$
Tortuosity of an edge	$\tau = \ell / \ \beta(1) - \beta(0)\ $

Table 1: Values used to compute edge statistics

1	Total number of nodes
2-5	Nodes of degrees 1, 2, 3, and 4+
6	Number of edges
7	Sum of all edge lengths
8-16	Percentiles [0,5,10,25,50,75,90,95,100] of edge lengths
17	Average edge length
18-26	Percentiles [0,5,10,25,50,75,90,95,100] of average curvatures
27	Average average curvature
28-36	Percentiles [0,5,10,25,50,75,90,95,100] of tortuosities
37	Average tortuosity

Table 2: Features of each graph used for classification

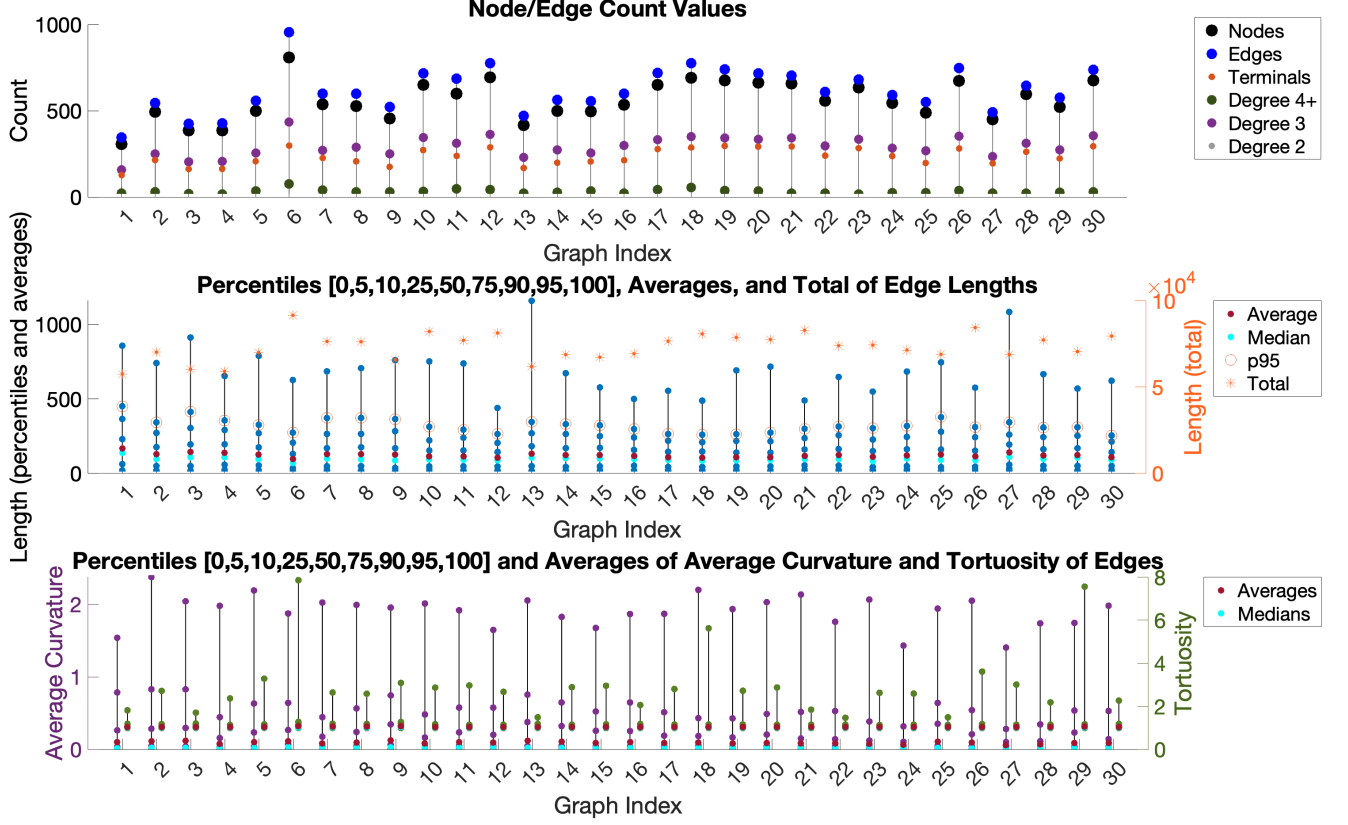


Fig. 11: Graph statistics used as features for classification. Each column (extended across rows) corresponds to a single graph. Statistics for 30 of the 45 unreduced graphs from the HRF data set are shown. Images 1-10 were labeled as healthy in the data set, images 11-20 were labeled as having diabetic retinopathy, and images 21-30 were labeled as glaucomatous.

Image Analysis (ARIA) Data Set [9, 10] and the High-Resolution Fundus (HRF) Image Database [21, 4]. These data sets include fundus images that were converted to binary segmentations prior to our download, which we converted to shape graph format as described in section 3. From the ARIA data set, we used 57 images from subjects labeled as healthy, 17 labeled as diabetic, and 18 labeled as having age-related macular degeneration (92 total). From the HRF data set, we used 15 images from subjects labeled as healthy, 15 labeled as having diabetic retinopathy, and 15 labeled as glaucomatous (45 total). Each image was given exactly one of these labels. Due to the limited sample sizes, we collapsed the disease categories and used the binary labels ‘healthy’ and ‘disease’ for our classification tasks.

For the classification experiments using neurons, we separately assessed two larger datasets

from NeuroMorpho.org [20, 2] consisting of various types of rat neurons: the Althammer archive [1] and the Markram archive [32]. The NeuroMorpho archives contain data in .swc file format which describes the data in terms of three-dimensional locations, and it is more straightforward to convert these into shape graphs. For the neuron datasets, the goal was to classify the cell type according to provided labels. For the Althammer data set, we performed binary classification using 685 microglia cells from the hypothalamus and 181 oxytocinergic neurons from the hypothalamus (866 total). From the Markram data set, we performed multi-class classification using 239 basket cells from the neocortex, 35 double bouquet cells from the neocortex, 83 Martinotti cells from the neocortex, 60 neurogliaform cells from the neocortex, 427 pyramidal neurons from the neocortex,

and 47 thalamocortical neurons from the thalamus (6 classes, 891 total).

For each RBV and neuron data set, we performed multi-resolution reduction as described in section 4 to produce five sets of graphs: unreduced graphs with no changes beyond preprocessing, graphs with preliminary terminal trimming but no further reduction, and graphs at resolutions 0.8, 0.6, and 0.4. Statistics of the graphs were collected at each level of complexity. We then performed classification using support vector machines (SVM) from the Scikit-learn [22] package in Python 3 [35]. We used SVMs with a Gaussian radial basis function kernel and tried a grid of values for parameters h and η which respectively control the penalty on the slack variables and the variance in the Gaussian kernel. We used 18 values of h including the numbers 0.5, 0.7, 0.9, and evenly-spaced values ranging from 1 to 29. For η we used 250 evenly-spaced values ranging from 0.0001 to 0.1001. This parameter region was selected based on coarser preliminary computations as producing the highest-accuracy results. The classification accuracy for the RBV data sets was determined using leave-one-out cross-validation and the accuracy for the neuron data sets used ten-fold cross-validation. We performed the experiments using the full set of 37 features as well as using only the 17 features excluding average curvature and tortuosity. For each graph set defined by data set, reduction level, and feature count, we computed the average, maximum, and standard deviation over all values in the grid of h and η .

6.3 Classification Results

Breakdowns of the datasets and the results from the classification experiments are presented in Tables 3 and 4 for the RBV and neuron data respectively. We emphasize that the primary goals here were to explore the use of interpretable geometry-based features and to analyze the relative importance of complex details versus general form in a shape graph representation. The aim was not to develop a state-of-the-art high-performance classifier. In particular, these results are not comparable to pixel-based classifiers. For reference, some recent approaches training neural networks on large data sets of color retinal fundus images have produced disease-detection classifiers that

rival human experts, with accuracy rates in the high 90% range for multiple retinal disorders in some cases [7, 25, 12, 16]. Concerning neurons, one recent paper tried a number of different deep neural network architectures to perform multi-class cell type classification on 35,000 rat neurons from NeuroMorpho.org and achieved accuracies around 90% [38].

The classification performance for the RBV data was better than chance in all cases and substantially better in most, with maximum accuracy for selected parameter values at or above .8 and reaching as high as .933. Accuracy rates for the unreduced graphs were reliably among the highest, only ever being surpassed by the trimmed-only results. Accuracy for the trimmed-only graphs was usually higher than for the graphs with reduced resolution. However, neither of these was universally the case. For the HRF data using 17 features, the trimmed-only graphs had much better results than any other resolution level. We note that the average and maximum here of .829 and .933 were the highest for any condition, and came from a setup using only 17 of the 37 features. For the ARIA data with 37 features, the graphs with resolution .6 performed nearly as well as the unreduced graphs and outperformed all other reduced resolutions, while the trimmed-only graphs had the lowest maximum and second-lowest average. There was no clear monotonic decrease in accuracy with resolution among the reduced-resolution RBV sets. The level .4 graphs had the lowest results among them in most cases, but these were usually not far from the next-lowest and included the highest average of the three for the HRF with 37 features.. The resolution .6 graphs had higher maximum accuracy than the resolution .8 graphs as often as not.

With the caveats that the results are not unanimous and that the data sets used are small, some general results are suggested. First, there is apparent potential in using these kinds of interpretable shape-based features to help detect retinal disease. Second, much of the information relevant for performing classification based on the disease labels in these data sets lies in the terminal edges. Third, reducing the complexity of the graphs using shapes that are similar-looking but not identical to the originals quickly removes most of the remaining information. That is, moderately reduced skeletons and heavily reduced skeletons

are equally poor in providing diagnostic information, despite the moderate skeletons being visually much more similar to the original graphs than are the heavily reduced ones.

There were some qualitative differences in the results for the neuron data. In the Althammer data set, the oxytocinergic neurons tended to have much denser graphs than the microglia and thus we considered the binary classification task to be an easy one. This was borne out in the results, with accuracy values ranging from mid to upper 90% values. There was monotonic decrease in accuracy values without exception as the graph resolution was reduced. However, the accuracy loss was not very severe, and even the .4 resolution set frequently produced accuracy values above 95%. Including the tortuosity and average curvature features improved the results everywhere except for the unreduced graphs, where the difference was very slight.

The multi-class classification in the Markram data set was a more difficult task. The classification rates were predictably lower than for the Althammer data, but were still much higher than chance. The classification rates gradually decreased from the trimmed-only graphs to the .4 resolution sets, but the lowest accuracies were not that much lower than the highest. Counterintuitively, the unreduced graphs had lower accuracies than the trimmed and .8 resolution sets. Only minor differences were seen between the results using 17 versus 37 features.

In both of the neuron datasets, the results suggest that gross shape is more relevant than fine details in distinguishing between these cell types.

7 Conclusions

This paper developed a method for reducing the detailed complexity of shape graphs while maintaining their fundamental shape. We demonstrated the method using shape graphs produced from real-world data in both two and three dimensions. We explored a variety of other reduction methods, a number of which apparently hold promise, and found the method spotlighted here to produce results that are visually more true to the originals.

The method was applied to disease detection in retinal fundus images and cell type classification of neurons by comparing classification

accuracies using graphs at different degrees of complexity reduction, basing the classifications on interpretable geometric features. For both types of data, we found that these features did contain information relevant for classification. For the retinal blood vessels, a great deal of that information was associated with the terminal edges of the graphs and classification accuracy dropped quickly when complexity was reduced. For the neural cells, classification rates were robust to complexity reduction even with a great deal of simplification.

In our view, the method’s subjective goal of extracting the basic form of a shape graph was convincingly achieved. It could be a useful tool in various applications where graph simplification is of interest.

Acknowledgements

The authors acknowledge and thank the producers of the Althammer, ARIA, HRF, Markram, and Narkilahti datasets for making them public. This research was supported in part by the NSF grants NSF CDS&E DMS 1953087 and NSF IIS 1955154.

References

- [1] F. Althammer et al. “Three-dimensional morphometric analysis reveals time-dependent structural changes in microglia and astrocytes in the central amygdala and hypothalamic paraventricular nucleus of heart failure rats”. In: *Journal of Neuroinflammation* 17:221 (2020).
- [2] G. A. Ascoli. “Mobilizing the base of neuroscience data: the case of neuronal morphologies”. In: *Nature Rev. Neurosci.* 7 (2006), pp. 318–324.
- [3] A. Basu Bal et al. “Statistical Shape Analysis of Shape Graphs with Applications to Retinal Blood-Vessel Networks”. In: *arXiv preprint stat.ME* (2022), p. 2211.15514.
- [4] A. Budai et al. “Robust Vessel Segmentation in Fundus Images”. In: *International Journal of Biomedical Imaging* 2013 (2013).
- [5] J. Yul Choi et al. “Multi-categorical deep learning neural network to classify retinal images: A pilot study employing small database”. In: *PLoS ONE* 12(11) (2017), e0187336.

ARIA Data Classes			
Classification Label	Sample Size		
Healthy	57		
Disease	35		
Total	92		
ARIA Accuracy, 37 Features			
	Average	Max	St.Dev
Unreduced	.689	.783	.030
Trimmed only	.631	.696	.026
$\rho = .8$.633	.728	.039
$\rho = .6$.670	.783	.039
$\rho = .4$.605	.707	.031

HRF Data Classes			
Classification Label	Sample Size		
Healthy	15		
Disease	30		
Total	45		
HRF Accuracy, 37 Features			
	Average	Max	St.Dev
Unreduced	.776	.889	.059
Trimmed only	.753	.867	.038
$\rho = .8$.677	.822	.036
$\rho = .6$.650	.778	.030
$\rho = .4$.684	.733	.030

ARIA Accuracy, 17 Features			
	Average	Max	St.Dev
Unreduced	.736	.804	.035
Trimmed only	.590	.652	.026
$\rho = .8$.543	.652	.037
$\rho = .6$.575	.663	.026
$\rho = .4$.537	.641	.032

HRF Accuracy, 17 Features			
	Average	Max	St.Dev
Unreduced	.752	.822	.039
Trimmed only	.829	.933	.041
$\rho = .8$.645	.800	.066
$\rho = .6$.681	.756	.041
$\rho = .4$.625	.689	.036

Table 3: Data descriptions and classification results for RBV data sets. Accuracies were determined using leave-one-out cross-validation. Average, max, and standard deviation refer to repeated assessments with varying SVM parameters for slack penalty and Gaussian variance.

- [6] V. Cohen-Addad et al. “Hierarchical Clustering: Objective Functions and Algorithms”. In: *J. ACM* 66.4 (2019), Article 26.
- [7] L. Dong et al. “Artificial Intelligence for Screening of Multiple Retinal and Optic Nerve Diseases”. In: *JAMA Network Open* 5 (2022), e229960.
- [8] W. Ellens et al. “Effective Graph Resistance”. In: *Linear Algebra and its Applications* 435 (2011), pp. 2491–2506.
- [9] D.J.J. Farnell. *Automated Retinal Image Analysis (ARIA) Data Set*. https://www.damianjffarnell.com/?page_id=276 (accessed January 17, 2023). 2006.
- [10] D.J.J. Farnell et al. “Enhancement of blood vessels in digital fundus photographs via the application of multiscale line operators”. In: *Journal of the Franklin Institute* 345 (7 2008), pp. 748–765.
- [11] X. Guo et al. “Statistical Shape Analysis of Brain Arterial Networks (BAN)”. In: *arXiv preprint cs.CV* (2022), p. 2007.047932v2.
- [12] P. Jiang, Q. Dou, and L. Shi. “Ophthalmologist-Level Classification of Fundus Disease With Deep Neural Networks”. In: *Trans Vis Sci Tech* 9(2) (2020), p. 39.
- [13] S. H. Joshi et al. “A Novel Representation for Riemannian Analysis of Elastic Curves in \mathbb{R}^n ”. In: *Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit* (2007), pp. 1–7.
- [14] D.J. Klein and M. Randić. “Resistance Distance”. In: *Journal of Mathematical Chemistry* 12 (1993), pp. 81–95.
- [15] S. Kurtsek, A. Srivastava, and W. Wu. “Signal Estimation Under Random Time-Warpings and Nonlinear Signal Alignment”. In: *Advances in Neural Information Processing Systems (NIPS)* 24 (2011).

Althammer Data Classes	
Cell Type	Sample Size
Microglia	685
Oxytocinergic	181
Total	866

Markram Data Classes	
Cell Type	Sample Size
Basket	239
Double Bouquet	35
Martinotti	83
Neurogliaform	60
Pyramidal	427
Thalamocortical	47
Total	891

Althammer Accuracy, 37 Features			
	Average	Max	St.Dev
Unreduced	.996	.999	.004
Trimmed only	.992	.997	.005
$\rho = .8$.985	.993	.005
$\rho = .6$.976	.982	.005
$\rho = .4$.962	.975	.007

Markram Accuracy, 37 Features			
	Average	Max	St.Dev
Unreduced	.667	.708	.019
Trimmed only	.688	.719	.016
$\rho = .8$.683	.717	.017
$\rho = .6$.663	.707	.018
$\rho = .4$.649	.690	.026

Althammer Accuracy, 17 Features			
	Average	Max	St.Dev
Unreduced	.998	1.000	.006
Trimmed only	.982	.985	.007
$\rho = .8$.965	.972	.007
$\rho = .6$.957	.967	.008
$\rho = .4$.944	.955	.008

Markram Accuracy, 17 Features			
	Average	Max	St.Dev
Unreduced	.678	.699	.015
Trimmed only	.690	.712	.015
$\rho = .8$.682	.701	.015
$\rho = .6$.667	.685	.012
$\rho = .4$.660	.681	.016

Table 4: Data descriptions and classification results for neuron data sets. Accuracies were determined using ten-fold cross-validation. Average, max, and standard deviation refer to repeated assessments with varying SVM parameters for slack penalty and Gaussian variance.

- [16] A. Y. Lee et al. “Multicenter, Head-to-Head, Real- World Validation Study of Seven Automated Artificial Intelligence Diabetic Retinopathy Screening Systems”. In: *Diabetes Care* 44 (2021), pp. 1168–1175.
- [17] A. Lieutier and M. Wintraecken. “Hausdorff and Gromov-Hausdorff stable subsets of the medial axis”. In: *arXiv preprint cs.CG* (2023), p. 2303.04014v1.
- [18] Y. Liu et al. “Graph summarization methods and applications: A survey”. In: *ACM computing surveys (CSUR)* 51.3 (2018), pp. 1–34.
- [19] D. Motta, W. Casaca, and A. Paiva. “Vessel Optimal Transport for Automated Alignment of Retinal Fundus Images”. In: *IEEE transactions on image processing* 28(12) (2019), pp. 6154–6168.
- [20] *NeuroMorpho.org*.
- [21] Pattern Recognition Lab (CS5). *High-Resolution Fundus (HRF) Image Database*. <https://www5.cs.fau.de/research/data/fundus-images/> (accessed January 17, 2023). 2013.
- [22] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [23] R. Poplin et al. “Predicting Cardiovascular Risk Factors from Retinal Fundus Photographs using Deep Learning”. In: *Nature Biomedical Engineering* 2 (2018), pp. 158–164.
- [24] N. Räsänen et al. “Practical guide for preparation, computational reconstruction and analysis of 3D human neuronal networks

- in control and ischaemic conditions”. In: *Development* 149(20) (2022).
- [25] J. Son et al. “Development and Validation of Deep Learning Models for Screening Multiple Abnormal Findings in Retinal Fundus Images”. In: *The American Academy of Ophthalmology* 127(1) (2019), pp. 85–94.
- [26] A. Srivastava, I. Jermyn, and S. Joshi. “Riemannian Analysis of Probability Density Functions with Applications in Vision”. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition* (2007), pp. 1664–1671.
- [27] A. Srivastava and E. P. Klassen. *Functional and Shape Data Analysis*. New York: Springer-Verlag, 2016.
- [28] A. Srivastava et al. “Registration of Functional Data Using Fisher-Rao Metric”. In: *arXiv preprint math.ST* (2011), p. 1103.3817v2.
- [29] A. Srivastava et al. “Shape Analysis of Elastic Curves in Euclidean Spaces”. In: *IEEE transactions on pattern analysis and machine intelligence* 33 (7) (2011), pp. 1415–1428.
- [30] A. Srivastava et al. “Statistical Shape Analysis: Clustering, Learning and Testing”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.24 (2005), pp. 590–602.
- [31] The MathWorks Inc. *MATLAB*. Natick, Massachusetts, United States, 2022+. URL: <https://www.mathworks.com>.
- [32] M. Toledo-Rodriguez et al. “Neuropeptide and calcium-binding protein gene expression profiles predict neuronal anatomical type in the juvenile rat”. In: *J Physiol* 567.2 (2005), pp. 401–413.
- [33] J. D. Tucker, W. Wu, and A. Srivastava. “Analysis of proteomics data: Phase amplitude separation using an extended Fisher-Rao metric”. In: *Electronic Journal of Statistics* 8 (2014), pp. 1724–1733.
- [34] J. D. Tucker, W. Wu, and A. Srivastava. “Generative models for functional data using phase and amplitude separation”. In: *Computational Statistics and Data Analysis* 61 (2013), pp. 50–66.
- [35] G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.
- [36] X. Wang et al. “Vessel Tech: A high-accuracy pipeline for comprehensive mouse retinal vasculature characterization”. In: *Angiogenesis* 24 (2021), pp. 7–11.
- [37] T. Wu et al. “Density-aware Chamfer Distance as a Comprehensive Metric for Point Cloud Completion”. In: *35th Conference on Neural Information Processing Systems* (2021).
- [38] T. Zhang et al. “Neuron type classification in rat brain based on integrative convolutional and tree-based recurrent neural networks”. In: *Scientific Reports* (2021), 11:7291.
- [39] F. Zhou and F. De la Torre. “Factorized Graph Matching”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38(9) (2016), pp. 1774–1789.