ABSTRACT. Recent advances in brain clearing and imaging have made it possible to image entire mammalian brains at sub-micron resolution. These images offer the potential to assemble brain-wide atlases of projection neuron morphology, but manual neuron reconstruction remains a bottleneck. In this paper we present a probabilistic method which combines a hidden Markov state process that encodes neuron geometric properties with a random field appearance model of the fluorescence process. Our method utilizes dynamic programming to efficiently compute the global maximizers of what we call the “most probable” neuron path. We applied our algorithm to the output of image segmentation models where false negatives severed neuronal processes, and showed that it can follow axons in the presence of noise or nearby neurons. Our method has the potential to be integrated into a semi or fully automated reconstruction pipeline. Additionally, it creates a framework for conditioning the probability to fixed start and endpoints through which users can intervene with hard constraints to, for example, rule out certain reconstructions, or assign axons to particular cell bodies.

E-mail addresses: tathey1@jhu.edu, DTward@mednet.ucla.edu, umuelle3@jhmi.edu, mim@cis.jhu.edu.

Date: August 9, 2021.
1. INTRODUCTION

Neuron morphology has been a central topic in neuroscience for over a century, as it is the substrate for neural connectivity, and serves as a useful basis for neuron classification. Technological advances in brain clearing and imaging have allowed scientists to probe neurons that extend throughout the brain to the other, and branch hundreds of times [34]. It is finally now becoming feasible to assemble a brainwide atlas of cell types in the mammalian brain which would serve as a foundation to understanding how the brain operates as an integrated circuit, or how it fails in neurological disease. One of the main bottlenecks in assembling such an atlas is the manual labor involved in neuron reconstruction. We propose a probabilistic model-based algorithm that operates on image segmentations to efficiently reconstruct neuronal processes.

Our approach draws upon two major subfields in Computer Vision, appearance modeling and hidden Markov-models to generate globally optimal solutions using dynamic programming. Our use of appearance models was highly motivated by Kass’s and Cohen’s early works [17, 8] on active shape modeling and their subsequent application by Wang et al. [33] for neuron reconstruction. For our own approach we exploit foreground-background models of image intensity for the data likelihood term in the hidden-Markov structure.

Our probabilistic models are hidden Markov random fields, but we reduce the computational structure to a hidden Markov model (HMM) since the latent structures are absolutely ordered. Hidden Markov modeling (HMM) involves two sequences of variables, one is observed and one is hidden. A popular application of HMM’s is in speech recognition where the observed sequence is an audio signal, and the hidden variables a sequence of words [26]. In our setting, the observed data is the image, and the hidden data is the contour representation of the axon or dendrite’s path. Maximum a posteriori (MAP) estimation of the hidden variables in an HMM can be done with the well-known Viterbi algorithm, an example of dynamic programming [13].

The HMM structures also provide us with the explicit opportunity to encode the geometric features of curves through the Frenet representation of curvature and torsion. The Markov chain prior distributions we describe incorporate these geometric features of axonal curves, which we have studied previously [2, 18].

In this work, we address the reconstruction of a single neuronal process, which fixes the dimension of the problem and allows us to compute globally optimal solutions. This is in contrast with conditional mean, or posterior sampling approaches such as in [16]. We limit our approach to an endpoint control problem choosing the initial and endpoint conditions associated to the soma and axon endpoint. The global optimality of our solutions arise from the fact that we can write the conditional distribution of the state sequence recursively and solve it as a shortest-path problem.

In this work, we apply our hidden Markov modeling framework to the output of image segmentation models. Convolutional neural networks have shown impressive results in image segmentation [28], but it only takes a few false negatives to sever neuronal processes that are often as thin as one micron (Figure 1). Our method can be used to string together connected components in a binary image mask in order to translate an image segmentation to a neuron reconstruction. We apply our method to data from the MouseLight project at Janelia Research Campus [34].

1.1. Related Works. Since neuron morphology plays such an important role in neuroscience, scientists have been reconstructing neurons in light microscopy images for a long time. Neuron reconstruction is time consuming, so many automated reconstruction algorithms have been proposed, especially over the last decade. In 2010, the DIADEM project brought multiple institutions together to consolidate existing algorithms, and stimulate further progress by generating open access image datasets, and organizing a contest for reconstruction algorithms [21]. Several years later, the BigNeuron project continued the legacy of DIADEM, this time
Previous approaches to automated neuron reconstruction have ranged from shortest path/geodesic based methods \cite{22, 33, 30}, minimum spanning trees \cite{37}, Bayesian estimation \cite{27}, tracking \cite{7, 10}, and deep learning \cite{14, 38, 20}. Methods have also been developed to enhance, or extend existing reconstruction algorithms \cite{32, 31, 24, 6}. Also, some works focus on the subproblem of resolving different neuronal processes that pass by closely to each other \cite{35, 19, 25}.

Our method is best classified as a shortest path based method, but it differs from state of the art in two main ways. First, unlike the active shape models in \cite{33}, we have chosen to exploit the conditional independence of the random field given the ordered Markov state sequences to generate globally optimal estimates \cite{3, 11}. Thus, our optimization method is not susceptible to local optima in the way that the gradient methods in active shape modeling are. The same distinction applies between our work and the elegant work on filtering \cite{27} in which the authors recursively push forward the conditional mean estimator. Our maximally probable solutions have the recursive property that for each transition through the state trellis graph we are also exploring only the most probable solutions. This shares the property of generating globally optimal solutions which are the goal of filtering methods which generate sequentially conditional mean solutions.

Our solution is hierarchical, generating an intermediate data structure which are segments of axons termed fragments, allowing us to leverage state of the art methods in machine learning for image segmentation. Our global optimization generating maximally probably paths solve for the optimal selection and optimal permutation for ordering the local axons fragments conditioned on soma and endpoint selection.
2. METHODS

2.1. The Bayesian Appearance Imaging Model. Our Bayes model for axon reconstructions models the axons as geometric objects on which we define the prior distribution with the image formation process defining the likelihood of the observation.

We model the axons as simply connected curves in \( \mathbb{R}^3 \) written as a function of arclength \( c(\ell), \ell \geq 0, c(\ell) \in \mathbb{R}^3 \).

We denote the entire axon curve in space as \( c_{\mathbb{R}^3} \): 

\[
\text{To interface the geometric object with the imaging volume we represent the underlying curve } c_{\mathbb{R}^3} \text{ as a delta-dirac impulse train in space. We view the imaging process as the convolution of the delta-dirac impulse train with the point-spread kernel of the imaging platform. We take the point-spread function of the system to be roughly one micron in diameter, implying that the axons are well resolved. The fluorescence process given the axon contour is taken as a relatively narrow path through the imaging domain (\( \approx 1 \mu m \) diameter) with relatively uniform luminance.}
\]

We take the image to be defined over the voxel lattice \( D = \bigcup_{i \in \mathbb{Z}^3} \Delta y_i \subset \mathbb{R}^3 \) with centers \( y_i \in \Delta y_i \). We model the image as a random field \( I_{y_i} \), \( \Delta y_i \in D \) which is an inhomogeneous Poisson process [29] with mean field \( E\{I_{y_i}\} = \lambda_{y_i} \). The Poisson assumption implies the random field has conditionally independent spatial increments across the image, implying the marginal probability at any fixed location encodes the joint probability for the fragments. This is consistent with the covariance plot in Figure 4. We adopt a foreground-background model for the images giving the marginal probability of a voxel intensity as:

\[
\begin{align*}
(1a) \quad p(I_{y_i}) &= \alpha_1(I_{y_i}), y_i \in \text{foreground} \\
(1b) \quad p(I_{y_i}) &= \alpha_0(I_{y_i}), y_i \in \text{background}
\end{align*}
\]

We denote the image random field associated to any subset of sites \( Y \subset D \), as

\[
I_Y = \{I_{y_i} : \Delta y_i \in Y\}.
\]

Modelling the image as a random field with independent increments, the appearance probability conditioned on the set being in foreground or background is given by the product of probabilities of (1a):

\[
\begin{align*}
(2a) \quad p(I_Y) &= \prod_{y \in Y} \alpha_1(I_{y_i}), \quad Y \subset \text{foreground} , \\
(2b) \quad p(I_Y) &= \prod_{y \in Y} \alpha_0(I_{y_i}), \quad Y \subset \text{background} ;
\end{align*}
\]

the shorthand notations for the joint probability of the set in foreground or background

\[
(2c) \quad \alpha_k(Y) := \prod_{y \in Y} \alpha_k(I_{y_i}), \quad k = 0, 1 .
\]

There can be many simple transformations between the original luminance measurement and the preprocessed data including scaling and shifting which affects the Poisson probability. To accomodate arbitrary scaling and shift we estimate the foreground-background intensity distributions from the data itself including all of its moments. We estimate \( \alpha_0(\cdot), \alpha_1(\cdot) \) by labeling a subset of the data as foreground/background then fitting Gaussian kernel density estimates to the labeled data (see Figure 3 below). Though we no longer use Poisson distributions, we maintain the independent increments assumption, which is consistent with the second order covariance curve depicted in Figure 4. This is the key property that allows us to factor joint probabilities into products of marginals for sets of voxels.

The foreground-background imaging model allows us to estimate the error rate of classifying a voxel as either foreground or background. In the Neyman-Pearson framework, foreground-background classification is a simple two hypothesis testing problem and the most powerful test at a given type 1 error rate is the log
likelihood ratio test. The Kullback-Leibler (KL) divergence between the foreground and background distributions gives the exponential rate at which error rates converge to zero as the number of independent, identically distributed samples increases [9]. In the case of using Gaussians to model the foreground-background distributions, the KL divergence reduces to the squared signal to noise ratio. In the absence of normality, the KL Divergence is the general information theoretic measure of image quality for arbitrary distributions.

2.2. The Prior Distribution via Markov State Representation on the Axons Fragments. Our representation of the observed image is as a hidden Markov random field with the axon as the hidden latent structure, exploiting the simplicity of conditional independence conditioned on the axon. Given the complexity of sub-micron resolution images, we build an intermediate data structure at the micron scale that represents pieces of axons called fragments \( F \subset D \) defined as collections of voxels without any assumed global ordering between them. Depicted in Figure 1 are the fragments shown via different colors. The fragments are a coarser scale voxel representation that gives us an efficient data structure upon which we reassemble and can be viewed analogously as higher order features.

The axon reconstruction problem becomes the reassembly of the fragments along with the imputation of the censored fragments. In the MouseLight images, the complexity of the space of fragments is approximately \(|F| = 100,000\) for a cubic millimeter of projection neuron image data.

The probability of a fragment under the foreground-background appearance model has the following compact notation; for \( F \subset \text{foreground or backround} \),

\[
\alpha_k(I_F) := \prod_{y \in F} \alpha_k(I_y), \quad k = 0, 1.
\]  

We exploit the computational structures of hidden Markov models (HMMs) when the underlying latent structure is absolutely ordered so that dynamic programming can efficiently compute globally optimal state sequence estimates. For each fragment \( F \in F \) we assume that there is a natural identification or correspondence to the states, \( F : s \in S \mapsto F(s) \in F \) with the state containing the minimum information required to specify the HMM. The state is augmented for all of the calculations to include the annotation of the endpoints of the fragments \( x^0, x^1 \in \mathbb{R}^3 \) so that we can compute "deletion" or "gap" probabilities along with unit length tangents \( \tau^0, \tau^1 \in \mathbb{R}^3 \) associated to the endpoints so we can compute curvature (Figure 2).

The collection of states \( s \in S \) are the hidden variables in the HMM.

Our algorithms exploit two splitting properties, the Markov nature of the state sequence and the splitting of the random field image conditioned on the state sequence. We use the notation \( s_{i:j} := (s_i, s_{i+1}, \ldots, s_j) \) for partial state sequences. We model the state sequence \( s_1, \ldots, s_n \) as Markov with splitting property:

\[
p(s_{i+1:n}, s_{1:i-1}|s_i) = p(s_{i+1:n}|s_k)p(s_{1:i-1}|s_i), \quad i = 2, \ldots, n - 1,
\]

which implies the 1-order Markov property \( p(s_i|s_{i-1}, s_{1:i-2}) = p(s_i|s_{i-1}) \).

We construct the Markov chain with the Boltzmann distribution for transition probabilities:

\[
p(s_i|s_{i-1}) = \frac{e^{-U(s_{i-1}, s_i)}}{Z(s_{i-1})}, \quad \text{with} \quad Z(s_{i-1}) = \sum_{s_i \in S} e^{-U(s_{i-1}, s_i)},
\]

and energy

\[
U(s_{i-1}, s_i) = \alpha_d ||x^0_i - x^1_{i-1}||^2 + \alpha_\kappa \kappa(s_{i-1}, s_i)^2.
\]

where \( || \cdot || \) is the standard Euclidean norm. The term \( \kappa(s_{i-1}, s_i) \) approximates the curvature of the path connecting \( s_{i-1} \) to \( s_i \) as follows:
Figure 2. Top row of panels shows how fragments and states are generated. Panel a shows an image subvolume and panel b shows a subset of foreground voxels which we call a fragment. From the fragment we estimate the endpoints (blue points in panel c) and the tangents at the endpoints (blue arrows in panel d). Panel d shows the rasterization of the line segment connecting the endpoints. Panels e and f illustrate that two states are generated from each fragment in the algorithm’s implementation, one for each orientation. Panels g and h shows how an imputed fragment $\bar{F}$ is computed when connecting two fragments.

Define $\tau_c := \frac{x_{i-1}^1 - x_i^0}{||x_{i-1}^1 - x_i^0||}$ which is the normalized vector connecting $s_{i-1}$ to $s_i$ and $\tau_i$, $\tau_{i-1}$ the tangent vectors, then

$$\kappa(s_{i-1}, s_i)^2 = \frac{\kappa_1(s_{i-1}, s_i)^2 + \kappa_2(s_{i-1}, s_i)^2}{2},$$

with $\kappa_1(s_{i-1}, s_i)^2 = 1 - \tau_{i-1} \cdot \tau_c$, $\kappa_2(s_{i-1}, s_i)^2 = 1 - \tau_c \cdot (-\tau_0^0)$. We derive these terms from the formula for curvature as modeled in [2] in Appendix B.

2.3. The MAP State Sequence. Define the maximizer of the posterior probability (MAP) on the state sequence $s_{1:n} \in S^n$, with $|S|$ finite:

$$\hat{s}_{1:n} := \arg\max_{s_{1:n} \in S^n} \log p(s_{1:n} | I_D).$$
2.4. Global Maximally Probable Solutions via Viterbi $O(n|S|^2)$ Calculation. The solution space has cardinality $|S|^n$ so it is infeasible to compute the global maximizer by evaluating all possibilities. Our approach is to rewrite the probability recursively in order to use the Viterbi algorithm and dynamic programming with $O(n|S|^2)$ time complexity. We rewrite the MAP estimator in terms of the joint probability:

$$\hat{s}_{1:n} = \arg\max_{s_{1:n} \in S^n} p(s_{1:n}|I_D) = \arg\max_{s_{1:n} \in S^n} p(s_{1:n}, I_D).$$

The image random field is split or conditionally independent conditioned on the fragment states:

$$p(I_{F(s_i)}; I_{D\setminus F(s_i)}|s_i) = p(I_{F(s_i)}|s_i)p(I_{D\setminus F(s_i)}|s_i),$$

which implies $p(I_{F(s_i)}|s_i, I_{D\setminus F(s_i)}) = p(I_{F(s_i)}|s_i)$.

We introduce the shorthand notation identifying fragment sequences with the state sequence:

$$F_{1:n} := (F(s_1), \ldots, F(s_n)).$$

Define the indicator function $\delta_A(x) = 1$ for $x \in A$, 0 otherwise.

Lemma 1. For $n > 1$ we have the joint probability:

$$p(s_{1:n}, I_D) = \prod_{i=2}^{n} \left( \frac{\alpha_{1}(I_{F_i})}{\alpha_{0}(I_{F_i})} \right)^{\delta_{D,F_{1:i-1}}(F_i)} p(s_i|s_{i-1}) p(s_1, I_D).$$

See Appendix C for the proof which rewrites the probability recursively giving the factorization.

There is often an analogy made between the negative logarithm of the joint probability and the total cost of a path through a directed graph [13], and several algorithms exist that can solve the shortest path problem. However, we cannot use them in this context because the cost function is not “sequentially-additive” due to the dependence of the indicator function on all previous states in the sequence.

In the results, section 3.6, we offer a example demonstrating that directly applying the Viterbi algorithm to this problem does not generate the MAP estimate.

Instead, we adjust our probabilistic representation the $|S|^n$ paths in order to utilize shortest path algorithms such as Bellman-Ford or Dijkstra’s [3][11]. For this we note that the $\frac{\alpha_{1}(I_{F_i})}{\alpha_{0}(I_{F_i})}$ term in equation 5 may often be greater than 1. In the analogy to shortest paths in directed graphs, this could lead to negative cycles in the graph of states. When negative cycles exist, the shortest path problem is ill-posed. To avoid this phenomenon we remove the background component of the image from the joint probability, which converts $\frac{\alpha_{1}(I_{F_i})}{\alpha_{0}(I_{F_i})}$ to $\alpha_{1}(I_y)$, and converts our global posterior probability to our path probability.

Theorem 1. Define the most probable solution $s_{1:n} \in S^n$ by the joint probability

$$\arg\max_{s_{1:n} \in S^n} p(s_{1:n}, I_{F_{1:n}}).$$

Then we have

$$\max_{s_{1:n} \in S^n} p(s_{1:n}, I_{F_{1:n}}) = \max_{s_{1:n} \in S^n} \prod_{i=2}^{n} \left( \alpha_{1}(I_{F_i}) \right)^{\delta_{D,F_{1:i-1}}(F_i)} p(s_i|s_{i-1}) p(s_1, I_{F_1}).$$

See Appendix C for proof.

We note that since the path defines the image in the joint probability we can define the probability only a function of state $\tilde{p}(s_{1:n}) := p(s_{1:n}, I_{F_{1:n}})$ emphasizing we are solving the most probable path problem.

We can maximize this expression using dynamic programming if $\alpha_{1}(\cdot)$ satisfies certain conditions: is a probability.

Proposition 1. Suppose that $\alpha_{1}(I_y) \leq 1$ for all $y$. Then the globally optimal solution to the fixed start and end point problem is a nonrepeating sequence and can be obtained by computing the shortest path in a directed graph where the vertices are the states, and the edge weight from state $s_{i-1}$ to $s_i$ is given by:
Proof. First, we want to show that the globally optimal sequence is nonrepeating. This is clear because if $\alpha_1(I_y) \leq 1$, then every term in the product in equation (6) is bounded by 1. Thus, for any state sequence that repeats states, if we remove all elements between the two instances of the repeated states, then this new sequence will have at least as much likelihood $p(s_{1:n}, I_{F_{1:n}})$.

For nonrepeating state sequences, the probability $p(s_{1:n}, I_{F_{1:n}})$ of (6) can be simplified:

$$p(s_{1:n}, I_{F_{1:n}}) = \prod_{i=2}^{n} (\alpha_1(I_{F_i})) p(s_i|s_{i-1}) p(s_1, I_{F_1}).$$

Taking the logarithm yields a sequentially additive cost function that can be solved with shortest path algorithm on a graph with edge weights given by Equation (7). \qed

2.5. Implementation.

2.5.1. Fragment generation: Fragments are collections of voxels, or “supervoxels," and can be viewed analogously as higher order features such as edgelets or corners. As described in section 2.2, identifying the subset of fragments that compose the axon, then ordering them becomes equivalent to reconstructing the axon contour model.

The first step of fragment generation is obtaining a foreground-background mask, which could be obtained, for example, from a neural network, or by simple thresholding. In this work, we use an Ilastik model that was trained on three image subvolumes, each of which has three slices that were labeled. During prediction, the probability predictions from Ilastik are thresholded at 0.9, a conservative threshold that keeps the number of false positives low.

The connected components of the thresholded image are split into fragments of similar size by identifying the voxel $v$ with the largest predicted foreground probability and placing a ball $B_v$ with radius $5\mu m$ on that voxel. The voxels within $B_v$ are removed and the process is repeated until the component is covered. The component is then split up into pieces by assigning each voxel to the center point $v$ that is closest to it. This procedure ensures that each fragment is no larger than a ball with radius $5\mu m$ (Figure 2 panel b). At this size, it is reasonable to assume that each fragment is associated with only one axon branch (no fragment is large enough to extensively cover multiple branches).

Next, the endpoints $x^0, x^1$ and tangents $\tau^0, \tau^1$ are computed as described in Appendix A (Figure 2 panel c).

Finally, each fragment is simplified to the line segment between its endpoints which is rasterized using the Bresenham algorithm (Figure 2 panel d). Briefly, the Bresenham algorithm identifies the image axis along which the line segment has the largest range and samples the line once every voxel unit along that axis. Then, the other coordinates are chosen to minimize the distance from the continuous representation line segment.

2.5.2. One State for Both Orientations. A fragment represents part of a neuronal process, and has no orientation. Orientation is required, however, to generate an ordering of the fragments for reconstruction. Our implementation generates two states for each fragment, one for each orientation of the fragment. The states are identical, except their endpoints and tangents are swapped (Figure 2 panels e and f).
2.5.3. **Imputing Fragment Deletions.** In practice the imaging data may exhibit significant dropouts leading to significant fragment deletions. While computing the likelihood of the image data, we augment the gaps between any pair of connected fragments in $F_1, F_2, \ldots$ by augmenting the sequence with imputed fragments $\bar{F}_1, \bar{F}_2, \bar{F}_1, \ldots$, with $\bar{F}_i \subset D$ the imputed line of voxels which forms the connection between the pair $F_i, F_{i+1}$. For this define the start and endpoint of each fragment as $x^0(F) \in \mathbb{R}^3, x^1(F) \in \mathbb{R}^3$ (see Figure 2) with line segment connecting each pair:

$$L_{i,i+1} = \{y : y = ax^1(F_i) + (1-a)x^0(F_{i+1}), a \in [0,1]\}.$$

The imputed fragment $\bar{F}_i \subset D$ for each pair $(F_i, F_{i+1})$ is computed by rasterizing $L_{i,i+1}$ with the Bresenham algorithm (Figure 2 panels g and h).

The likelihood of the sequence of fragments augmented by the imputations becomes

$$p(s_{1:n}, I_{F_{1:n}}) = \prod_{i=2}^{n} \alpha_1(I_{F_i}) \delta_{D,F_{i-1}}(F_i) \alpha_1(I_{\bar{F}_i}) p(s_i | s_{i-1}) p(s_1,I_{F_1})$$

2.5.4. **Initial and Endpoint Conditions.** We take the initial conditions to represent

$$p(s_1, I_{F_1}) = \pi(s_1)p(I_{F_1} | s_1),$$

with $\pi$ the prior on initial state. For all of our axon reconstructions we specify an axonal fragment as the start state $s_{\text{start}}$ and set $\pi(s_1) := \delta_{s_{\text{start}}}(s_1)$.

The endpoint conditions are defined via a user specified terminal state $s_{\text{term}}$ where the path ends giving the maximization:

$$\max_{s_{1:n} \in \mathcal{S}^n} p(s_{1:n}, I_{F_{1:n}} | s_n = s_{\text{term}}).$$

The marginal probability on the terminal state always transitions to itself, so that $p(s_n = s_{\text{term}}) = \delta_{s_{\text{term}}}(s_n)$. Thus, a state sequence solution of length $n$ may end in multiple repetitions of $s_{\text{term}}$, such as

$$s_{1:n} = \{s_1, s_2, \ldots, s_n', s_{\text{term}}, s_{\text{term}}, \ldots, s_{\text{term}}\}.$$

2.5.5. **Modified Prior for Computational Efficiency.** Rather than computing prior probabilities for all $|\mathcal{S}|^2$ possible state transitions, we rule out (set probability to 0) any transitions where the distance between endpoints is greater than 15$\mu$m or the angle between states is greater than 150 degrees.

2.6. **Existing Reconstruction Algorithms and Comparison to Manual Traces.** We applied several state of the art reconstruction algorithms to our data for comparison. Out of the 26 reconstruction algorithms available in Vaa3D version 3.2 [23], we chose the three algorithms that have accompanying publications, and most closely resemble our approach. The first method is APP2 which starts with an oversegmentation of the neuron using shortest path algorithm, then prunes spurious connections [35]. The second is Snake which is based on active contour modeling [33]. The third is Advantra which is based on the Bayesian filtering work from [27]. We also included the G-Cut algorithm, which disentangles oversegmentations of neuron clusters by maximizing the extent to which neuronal processes travel directly away from the soma [19]. We applied G-Cut to the reconstruction by APP2. We denote our algorithm “Maximum Probability Neuron Path” (MPNP).

We applied the algorithms to the initial segments of 10 different neurons in one of the brain samples from the MouseLight Project from HHMI Janelia [34]. Briefly, projection neurons were sparsely labeled then imaged with a two-photon microscope at a voxel resolution of $0.3 \times 0.3 \times 1\mu$m. Each axon reconstruction is generated semi-automatically by two independent annotators.

We quantified reconstruction accuracy using three metrics, the first of which is Frechet distance. Frechet distance is commonly described in the setting of dog walking, where both the dog and owner are following
different predetermined paths. The Frechet distance between the two paths then is the minimum length dog leash needed to complete the walk, where both dog and owner are free to vary their walking speeds but are not allowed to backtrack. In our setting we compute the Frechet distance between two discrete paths \( P : \{1, \ldots, L_p\} \rightarrow \mathbb{R}^3, Q : \{1, \ldots, L_q\} \rightarrow \mathbb{R}^3 \) as defined in [12]. Assume without loss of generality that \( L := L_p = L_q \), since the shorter sequence can always have the final coordinate repeated at the end.

\[
\delta_{dF}(P, Q) = \min_{\sigma_p, \sigma_q \in \Sigma_L} \max_{n \in \{1, \ldots, L\}} \| P[\sigma_p[n]] - Q[\sigma_q[n]] \|
\]

where \( \Sigma_L \) is the space of reparameterizations of sequences of length \( L \) i.e. it is a map from \( \{1, \ldots, L\} \rightarrow \{1, \ldots, L\} \) that is surjective and non-decreasing. We use the standard Euclidean norm for \( \| \cdot \| \). The discrete Frechet distance is an upper bound to the continuous Frechet distance between polygonal curves, and it can be computed more efficiently. Further, if we take a discrete Frechet distance of zero to be an equivalence relation, then \( \delta_{dF} \) is a metric on this set of equivalence classes and thus is a natural way to compare non-branching neuronal reconstructions. In this work, all reconstruction segments are upsampled to one point per micron when necessary.

Various other performance metrics have been proposed, including an arc-length based precision and recall [33], a critical node matching based Miss-Extra-Scores (MES) [36] and a vertex matching based substantial spatial distance (SSD) [23]. We chose to compute a modified SSD since it gives a picture of the average spatial distance between two reconstructions deviate. This complements the Frechet distance described earlier, which computes the maximum spatial distance between two reconstructions.

The first step in computing the SSD between reconstruction \( P \) and reconstruction \( Q \) is, for each point in \( P \), finding the distance to closest point in \( Q \). Substantial directed divergence of \( P \) from \( Q \) is then defined as the average of all these distances that exceed a certain threshold. Then, the same process is repeated, except flipping the roles of \( P \) and \( Q \). SSD is then defined as the average of these two substantial directed divergence values. The manual reconstructions from the Mouselight data are not always sampled frequently enough to capture the twists and turns of axons (e.g. Figure 10 panel 4), so the directed divergence of the algorithmic reconstruction from the manual reconstruction is misleadingly high. For this reason, we only report substantial directed divergence of manual reconstruction from algorithmic reconstruction.

3. Results

3.1. Appearance Model. Figure 3 shows kernel density estimates of the foreground and background image intensity distributions. The distributions vary greatly between the three image subvolumes, implying that modeling the image process as homogeneous throughout the whole brain would be inappropriate. Additionally, many of the distributions, especially for the manual labels, do not appear to be either Gaussian or Poisson. Indeed, Kolmogorov-Smirnov tests rejected the null hypothesis for both Gaussian and Poisson goodness of fit in all cases, with all p-values below \( 10^{-4} \).

The key motivation for introducing the Poisson assumption is that it implies independent increments. For that reason, the K-L divergence remains the important test statistic governing performance of the fragment generation through the appearance modeling.

We also investigated the covariance between nearby voxels in the image background (Figure 4) and found that covariance decayed rapidly with increasing distance, lending support to our conditionally independent spatial increment assumption.
Figure 3. Kernel density estimates of foreground and background intensity distributions of three different subvolumes (three columns) of one of the MouseLight images. A subset of the voxels in each subvolume were manually labeled (top row), then used to train an Ilastik model to classify the remaining voxels (bottom row). In each panel, we computed the KL Divergence between the foreground and background kernel density estimates. These panels show that the intensity distributions vary across different regions of the image.

3.2. Path Reconstruction. We pose our reconstructions as an endpoint control problems where initial and end states are fixed. We demonstrate our algorithm on a 2D image, an image with a single axon segment, and several images that contain many neuronal processes.

3.2.1. 2D Image Example. Shown in Figure 5 is a demonstration on a satellite image of part of the Great Wall of China. Left panel shows the image with the fixed start (green) and end (red) points. The middle panel shows the results of an image segmentation. Pixels that were classified as foreground are pictured in a color, and the different colors represent different connected components of the foreground pixels. Our algorithm identifies a sequence of components from the start to the end point. This sequence is shown in the right panel, where the fragments are colored in red and overlaid with a blue line connecting the endpoints of the fragments.

3.2.2. Maximally Likely Axon Reconstructions with Fragments. Figure 6 illustrates the algorithm being applied to part of an axon. In this example, there is very little signal dropout, and no other neuronal processes, and the algorithm successfully follows the axon from start point to end point.

3.3. Missing Fragments due to Luminance Dropout. Figure 7 examines the essential importance of the fragments. The fragment process determines the state space that the algorithm explores. If the density of tracing exhibit luminance drop out there will be no states which guarantees the algorithm will not finds those areas. The reconstruction algorithm fails when when the fragment generation process neglects significant
Figure 4. Covariance of image intensities between voxels at varying displacements from each other, normalized by total variance. Different displacements (all in the direction of $[1, 1, 1]$) are plotted along the horizontal axis. In each case, 1000 voxel pairs were sampled. The three lines represent three different image subvolumes, and in all three cases, the covariance drops to zero rapidly, suggesting that image intensity between neighboring background voxels are not correlated.

Figure 5. Satellite image of part of the Great Wall of China with left panel showing the image with the fixed start (green) and end (red) points. The middle panel shows the results of an image segmentation by, in this case, a random forest model in Ilastik with the space of fragments, $\mathcal{F}$, pictured in color. The maximally probable sequence is shown in the right panel, where the fragments are colored in red and overlaid with a blue line connecting the endpoints of the fragments.

portions ($>\sim 10\mu m$) of the neuronal process. The left panel shows a neuron with the manual reconstruction in green. Close up views in panels 1 and 2 show that the signal from the axon under the reconstruction is weak (and barely visible). Indeed, panel 3 shows that fragments were not generated for this stretch of axon.
The reconstruction from the algorithm shown by the blue line in panel 4 starts in the right direction, then turns onto a nearby dendrite.

3.4. **Hyperparameter Selection.** Figure 8 examined the reconstruction for different choices of the hyperparameters controlling distances between states $\alpha_d$ and curvature of the path $\alpha_k$. In the example shown here, the three choices of hyperparameters lead to three different reconstruction results. The left panel shows a neuron with the manual reconstruction in green. Panels 1-3) show the reconstructions from the algorithm, with the sequence of fragments shown in unique colors, and the path connecting them shown as a blue line. Panel 1 shows the result with $\alpha_d = 10$ and $\alpha_k = 1000$. The dotted red circle indicates where the reconstruction deviated from the true path by jumping $\sim 5 \mu m$ to connect the purple fragment to the light blue fragment. Panel 2 shows the result with $\alpha_d = 100$ and $\alpha_k = 1000$. The higher $\alpha_d$ value avoids the jump in panel 1, but this time, the dotted red circle shows where the reconstruction takes a sharp turn to deviate from the true path. Finally, in panel 3, $\alpha_d = 100$ and $\alpha_k = 10000$ avoids both the jump from panel 1 and the sharp turn from panel 2 and follows the true path of the axon back to the cell body.

3.5. **Several Maximally Likely Axon Reconstructions.** Figure 9 shows several examples of our maximally probable reconstruction algorithm. Left panels show an image subvolume containing one or more neurons. The middle panels show the same image overlaid with the fragments where each fragment has a unique color. The green arrow indicates the starting fragment and the red arrow indicates the ending fragment. The right panels show the path selected by the algorithm. The fragments in the path are shown in unique colors and the blue line shows the path that connects the endpoints of the fragments.

Figure 10 shows successful reconstruction of several projection neuron axons from the MouseLight dataset. The endpoints were fixed with the paths shown identified by the maximally probable algorithm depicted with by the red line, with the path from the manual reconstruction shown by the green line. The algorithm was run with the same hyperparameters in each case: $\alpha_d = 10$ and $\alpha_k = 1000$. 
Shown in Figure 7 are examples of failures to reconstruct the projection neurons to be consistent with the hand tracing. In all cases the luminance evidenced by the intensity image $I(y), y \in R^3$ appeared to have extended low-luminance sections resulting in fragment generation dropout.

Shown in Table 12 are the performance metrics evaluating how the different automatic reconstruction algorithms compared to the manual ground truth reconstructions. In example 3 in the successes group, our algorithm had a Frechet distance greater than 14$\mu$m from the manual reconstruction, which is suspiciously high for a reported success. After investigating, we found that the manual reconstruction deviated around 10$\mu$m from the axon around the axon hillock. After removing this portion of the reconstruction, the Frechet distance fell to around 6$\mu$m.

3.6. **Viterbi Counter-example.** Here is a simple counter example demonstrating that the Viterbi algorithm fails to identify the globally optimal state sequence in our setting:
Figure 8. Figure showing different selections of hyperparameters $\alpha_d$ and $\alpha_\kappa$. The left panel shows a neuron with the manual reconstruction in green. Panel 1 shows $\alpha_d = 10, \alpha_\kappa = 1000$, panel 2 $\alpha_d = 100, \alpha_\kappa = 1000$, panel 3 $\alpha_d = 100$ and $\alpha_\kappa = 10000$. The sequence of fragments are shown in unique colors with the path connecting them shown as a blue line. The red circle in Panel 1 indicates where the reconstruction deviated from the true path by jumping $\sim 10 \mu m$ to connect the gray fragment to the light blue fragment. Panel 2 shows the result with the higher $\alpha_d$ value avoiding the jump in panel 1, but this time, the red circle shows where the reconstruction takes a sharp turn to deviate from the true path. Finally, in panel 3, $\alpha_d = 100$ and $\alpha_\kappa = 10000$ avoids both the jump from panel 1 and the sharp turn from panel 2 and follows the true path of the axon back to the cell body.

$$S = \{1, 2\}, n = 4, \pi(s_1 = 1) = 1$$

$$\frac{\alpha_1(I_1)}{\alpha_0(I_1)} = 1$$

$$\frac{\alpha_1(I_2)}{\alpha_0(I_2)} = 1/100$$

The transition probabilities are given in Figure 13.

After three iterations, the Viterbi algorithm stores (1, 1, 1, 2) as the highest probability path of length 4 that ends in state 2 (Figure 14a). The joint probability of this path is 1/800. However, there is an alternative path that the algorithm missed, (1, 2, 1, 2), which has higher joint probability 1/400 (Figure 14b). The algorithm did not identify the globally optimal sequence because it did not “see” that the cost of transitioning from state 1 to 2 would drop from 1/200 to 1/2 after visiting state 2 the first time.

4. Discussion

In this work we present hidden Markov models for reconstructing neuronal processes. This process involves converting the connected components of an image mask into a set of “fragments” that are no longer
From these fragments we estimate the end points, and the tangent of the fragment at those endpoints in order to construct a state space that supports connection strength and curvature. We derive the Bayes posterior for maximum a-posteriori distribution estimation of the hidden state sequence encoding the axon reconstruction. We show that $O(n|S|^2)$ viterbi calculation is not possible for the MAP estimator since the state grows for the MAP solution. We modify our procedure to defining the path probabilities rather than the posterior probability which make it feasible to calculate efficiently the global maximum.

We apply the algorithm in the case of fixed start and end points on two-photon image data of neurons in mice. In a dataset of 10 partial axon reconstructions, our algorithm is competitive with existing state of the art algorithms (Table 12). We observed that the most common failure mode in this dataset was when there
FIGURE 10. Showing maximally probable reconstruction of projecting neurons from the MouseLight dataset which are successful; identified path is shown by the red line; path from the manual reconstruction is shown by the green line. The algorithm was run with the same hyperparameters in each case: $\alpha_d = 10$ and $\alpha_\kappa = 1000$.

FIGURE 11. Showing failures of maximally probable reconstruction of projecting neurons from the MouseLight dataset; identified path is shown by the red line; path from the manual reconstruction is shown by the green line. The algorithm was run with the same hyperparameters in each case: $\alpha_d = 10$ and $\alpha_\kappa = 1000$.

are extended stretches of the axon where no fragments have been generated (Figure 7). Thus, this algorithm is only effective if paired with an effective voxel classification tool. The algorithm can also fail in areas densely populated with neuronal processes. However, tuning the hyperparameters according to the density of the fragments and the geometry of the underlying neuron can solve this issue (Figure 8). Our algorithm is specifically adapted for reconstructing axons in projection neurons in datasets such as MouseLight, in two ways. First, the high image quality as indicated by large KL-divergence values (Figure 3), makes it straightforward to build an effective foreground-background classifier. Secondly, our algorithm can be tuned to different levels of curvature, which can adapt to the occasional sharp turns in projection axons (Figures 10 and 11).

We propose that this method can be used in semi-automatic neuron reconstruction where a tracer could click on two different fragments along a neuronal process, and this algorithm would fill in the gap. Or, this algorithm could be used in a completely automated neuron reconstruction pipeline. The biggest hurdle in extending this algorithm to fully-automated setting would be designing the distributions in the model so they accurately model the underlying neuron generation process. Our future work will be focused on this endeavor,
**Figure 12.** Shown are performance metrics comparing the manual ground truth reconstructions to those from several algorithms including MPNP (our algorithm), APP2, G-Cut, Snake, and Advantra. The samples are organized into successes and failures according to a visual evaluation of whether our algorithm correctly reconstructed the axon (see Figures 10 and 11). The best performances for each subvolume according to a given metric are highlighted in green.

| Example No. (Figures 10, 11) | Successes | Failures |
|-------------------------------|------------|----------|
| Frechet Distance ($\mu m$)    |            |          |
| MPNP                          | 4.7        | 4.0      |
| APP2                          | 4.7        | 7.0      |
| G-Cut                         | 22.9       | 9.6      |
| Snake                         | 121.3      | 4.0      |
| Advantra                      | 82.3       | 101.6    |
| Substantial Directed Divergence ($\mu m$) |            |          |
| MPNP                          | 2.2        | 2.4      |
| APP2                          | 2.2        | 5.3      |
| G-Cut                         | 13.0       | 7.5      |
| Snake                         | 2.3        | 3.1      |
| Advantra                      | 24.4       | 13.9     |

**Figure 13.** Transition probabilities of a two state Markov model that demonstrates the Viterbi algorithm is invalid in our setting.

ensuring that the state transition distribution and image emission distribution lead to robust reconstruction of neuronal processes.

**APPENDIX A. COMPUTING ENDPOINTS AND TANGENTS OF FRAGMENTS**

As explained in section 2.2, each fragment is a subset of the image $F \subset D$. Each fragment is assumed to be associated with a segment of an underlying neuron curve, and we want to estimate the locations of the two endpoints of the fragment. First, we identify which of the three voxel axes the fragment is longest, then measure its length along that axis. We divide this length by two to get $R$. Then, with each voxel $y$ in the fragment, we associate a set of voxels $N_y$ which is the intersection of the voxels in the fragment and the
Figure 14. Example showing that the Viterbi algorithm is invalid in our setting. The nodes represent the states and the line segments represent state transitions, as in Figure 8 of [13]. Panel a) illustrates three iterations of the Viterbi algorithm, with the lines representing the paths that are stored at each iteration. In this example, the blue path at the bottom of panel a) is identified by naive application of the Viterbi algorithm, but it is not the highest probability path of length 4 that ends in state 2 - the shortest such path is shown in panel b). Each transition has a number above it indicating its contribution to the joint probability - \( \alpha_1 p(x_k | q) \) from Equation 5. The number at the end of the path is the product of all such terms - the joint probability of the given path.

voxels within distance \( R \) from \( y \). The voxel with the smallest set \( N_y \) (by cardinality) is chosen to be the first endpoint. The second endpoint is the voxel that has the smallest set \( N_y \) but is also farther than \( R \) away from the first endpoint.

Currently the tangents at the endpoints is just approximated by the difference of the endpoints i.e. \( \tau^0 = \frac{x_0 - x_1}{\| x_0 - x_1 \|}, \tau^1 = -\tau^1 \). This method is based on the assumption that fragments are small enough that they are approximately straight. We also experimented with approximating the endpoint tangents by computing the one dimensional linear subspace that best fit the fragment voxels near the endpoints, but found it to be less robust.

Appendix B. Curvature Calculation for the Potential in the Markov Chain

The term \( \kappa(s_{i-1}, s_i) \) is an approximation of the curvature of the path that connects \( s_{i-1} \) to \( s_i \). For a curve with the tangent vector \( T(s) \), curvature is defined as \( \kappa(s) = \| \frac{dT}{ds} \| \). A finite difference approximation of curvature is then:
**Figure 15.** We applied the Vaa3D implementation of APP2 [35] to the partial axon dataset. Panels 1 and 2 show examples of APP2 successfully tracing the axon back to the soma. Panels 3 and 4 show examples of APP2 failing to follow the true axon path. In all panels, the true trace is shown in green, and the trace by APP2 is shown in blue. In panels 1 and 2, the full neuron reconstruction by APP2 is shown by the red skeletonization.

**Figure 16.** We applied the Vaa3D implementation of Open-Curve Snake neuron tracing [33] to the partial axon dataset. The true trace is shown in green, and the trace by this method is shown in blue. The underlying full reconstruction by Open-Curve Snake is shown by the red skeletonization. Panels 1 and 2 show examples of this method successfully tracing the axon back to the soma. Panel 3 shows an example where the red skeletonization does not extend up the axon, and therefore axon reconstruction failed.

\[
\left\| \frac{dT}{ds} \right\|^2 \approx \left\| T(s) - T(s - 1) \right\|^2 \\
= \left\| T(s) \right\|^2 + \left\| T(s - 1) \right\|^2 - 2T(s) \cdot T(s - 1) \\
= 2(1 - T(s) \cdot T(s - 1))
\]
Thus,

$$\left( \frac{dT}{ds} \right)^2 \propto 1 - T(s + 1) \cdot T(s)$$

We consider $\tau_i$, $\tau_{i-1}$ and the normalized vector between the states $\tau_c := \frac{x_i^1 - x_i^0}{|x_i^1 - x_i^0|}$ as samples of $T(s)$, then use Eq. (8) to estimate the curvature induced by connecting state $s_{i-1}$ to $s_i$.

$$(\kappa_1)^2 = 1 - \frac{1}{\tau_{i-1}} \cdot \tau_c$$

$$(\kappa_2)^2 = 1 - \tau_c \cdot (-\tau_i^0)$$

$$\kappa(s_{i-1}, s_i)^2 = \frac{(\kappa_1)^2 + (\kappa_2)^2}{2}$$

Arithmetic mean

**APPENDIX C. PROOF OF LEMMA 1 AND THEOREM 1**

It will be useful to introduce the following compact notation to represent the likelihood of a complete fragment under the foreground-background model:

$$\alpha_k(I_F) := \prod_{y \in F} \alpha_k(I_y), \quad k = 0, 1.$$ 

Notice $p(I_F|F) = \alpha_1(I_F)$. We define the indicator function $\delta_A(x) = 1$ for $x \in A$, 0 otherwise.

**Lemma 1**

For $n > 1$ we have the recursion probability

$$(10a) \quad p(s_{1:n}, I_D) = \left( \frac{\alpha_1(I_{F_n})}{\alpha_0(I_{F_n})} \right)^{\delta_D|F_{1:n-1}(F_n)} p(s_n|s_{n-1}) p(s_{1:n-1}, I_D),$$

implying the factored probability:

$$(10b) \quad p(s_{1:n}, I_D) = \prod_{i=2}^n \left( \frac{\alpha_1(I_{F_i})}{\alpha_0(I_{F_i})} \right)^{\delta_D|F_{1:i-1}(F_i)} p(s_i|s_{i-1}) p(s_1, I_D).$$

**Proof.** Factor the event $I_D = (I_{F_{1:n}}, I_{D\setminus F_{1:n}})$ as in [15] for road tracking, then

$$p(s_{1:n}, I_{F_{1:n}}, I_{D\setminus F_{1:n}}) = p(I_{F_n}|s_{1:n}, I_{F_{1:n-1}}, I_{D\setminus F_{1:n}}) p(s_n|s_{n-1}, I_{D\setminus F_{1:n}}) p(s_{1:n-1}, I_{F_{1:n-1}}, I_{D\setminus F_{1:n}})$$

$$= p(I_{F_n}|s_n) \delta_D|F_{1:n-1}(F_n) p(s_n|s_{n-1}) p(s_{1:n-1}, I_{F_{1:n-1}}, I_{D\setminus F_{1:n}})$$

$$= \alpha_1(I_{F_n}) \delta_D|F_{1:n-1}(F_n) p(s_n|s_{n-1}) p(s_{1:n-1}, I_{F_{1:n-1}}, I_{D\setminus F_{1:n}})$$

(11)

We rewrite the last term using the splitting property:

$$p(s_{1:n-1}, I_{F_{1:n-1}}, I_{D\setminus F_{1:n}}) = p(I_{F_{1:n-1}}|s_{1:n-1}, I_{D\setminus F_{1:n}}) p(s_{1:n-1}, I_{D\setminus F_{1:n}})$$

$$= p(I_{F_{1:n-1}}|s_{1:n-1}) p(I_{D\setminus F_{1:n}}|s_{1:n-1}) p(s_{1:n-1})$$

$$= p(I_{F_{1:n-1}}|s_{1:n-1}) \frac{p(I_{D\setminus F_{1:n-1}}|s_{1:n-1}) p(s_{1:n-1})}{p(I_{F_n}|s_n) \delta_D|F_{1:n-1}(F_n)}$$

$$= \frac{1}{\alpha_0(I_{F_n}) \delta_D|F_{1:n-1}(F_n)} p(s_{1:n-1}, I_{F_{1:n-1}}, I_{D\setminus F_{1:n}}).$$
with the last substitution following from the background model. Substituting into (11) yields the probability
written as a recursion (10a):
\[
p(s_{1:n}, I_D) = \left( \frac{\alpha_1(I_{F_n})}{\alpha_0(I_{F_n})} \right)^{\delta_{D,F_{1:n-1}}(F_n)} p(s_n | s_{n-1}) p(s_{1:n-1}, I_D),
\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
\[\]
[9] M. Cover and A. Thomas. Elements of information theory. Volume 2. Wiley, 1991.

[10] Tianhong Dai, Magda Dubois, Kai Arulkumaran, Jonathan Campbell, Cher Bass, Benjamin Billot, Fatmaturzehra Uslu, Vincenzo De Paola, Claudia Clopath, and Anil Anthony Bharath. Deep reinforcement learning for subpixel neural tracking. In International Conference on Medical Imaging with Deep Learning, pages 130–150. PMLR, 2019.

[11] Edsger W. Dijkstra. A note on two problems in connexion with graphs. Numerische Mathematik, 1(1):269–271, 1959.

[12] Thomas Eiter and Heikki Mannila. Computing discrete Fréchet distance. Technical report, Citeseer, 1994.

[13] G. David Forney. The viterbi algorithm. Proceedings of the IEEE, 61(3):268–278, 1973.

[14] Drew Friedmann, Albert Pun, Eliza L. Adams, Jan H. Lui, Justus M. Kebschull, Sophie M. Grutzner, Caitlin Castagnola, Marc Tessier-Lavigne, and Liqun Luo. Mapping mesoscale axonal projections in the mouse brain using a 3d convolutional network. Proceedings of the National Academy of Sciences, 117(20):11068–11075, 2020.

[15] Donald Geman and Bruno Jedynak. An active testing model for tracking roads in satellite images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(1):1–14, 1996.

[16] Ulf Grenander and Michael I. Miller. Representations of knowledge in complex systems. Journal of the Royal Statistical Society. Series B (Methodological), 56(4):549–603, 1994.

[17] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. International journal of computer vision, 1(4):321–331, 1988.

[18] N. Khaneja, M. I. Miller, and U. Grenander. Dynamic programming generation of curves on brain surfaces, 1998.

[19] R. Li, Muye Zhu, Junning Li, M. Bienkowski, Nicholas N. Foster, Hanpeng Xu, T. Ard, Ian Bowman, Changle Zhou, M. Veldman, X. W. Yang, Houri Hintiryan, Junsong Zhang, and H. Dong. Precise segmentation of densely intertwearing neuron clusters using g-cut. Nature Communications, 10, 2019.

[20] Rongjian Li, Tao Zeng, Hanchuan Peng, and Shuiwang Ji. Deep learning segmentation of optical microscopy images improves 3-d neuron reconstruction. IEEE transactions on medical imaging, 36(7):1533–1541, 2017.

[21] Hanchuan Peng, Erik Meijering, and Giorgio A Coslini. From diadem to bigneuron, 2015.

[22] Hanchuan Peng, Zongcai Ruan, Deniz Atasoy, and Scott Sternson. Automatic reconstruction of 3d neuron structures using a graph-augmented deformable model. Bioinformatics, 26(12):138–146, 2010.

[23] Hanchuan Peng, Zongcai Ruan, Fuhui Long, Julie H Simpson, and Eugene W. Myers. V3d enables real-time 3d visualization and quantitative analysis of large-scale biological image data sets. Nature biotechnology, 28(4):348–353, 2010.

[24] Hanchuan Peng, Zhi Zhou, Erik Meijering, Ting Zhao, Giorgio A Ascoli, and Michael Hawrylycz. Automatic tracing of ultra-volumes of neuronal images. Nature methods, 14(4):332–333, 2017.

[25] Tingwei Quan, Hang Zhou, Jing Li, Shiwei Li, Anan Li, Yuxin Li, Xiaohua Lv, Qingming Luo, Hui Gong, and Shaoqun Zeng. Neurogps-tree: automatic reconstruction of large-scale neuronal populations with dense neurites. Nature methods, 13(1):51–54, 2016.

[26] Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. IEEE asp magazine, 3(1):4–16, 1986.

[27] Miroslav Radojević and Erik Meijering. Automated neuron tracing using probability hypothesis density filtering. Bioinformatics, 33(7):1073–1080, 01 2017.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015.

[29] Donald L Snyder and Michael I Miller. Random point processes in time and space. Springer Science & Business Media, 2012.

[30] Engin Turetken, Fethallah Benmansour, Bjoern Andres, Hanspeter Pfister, and Pascal Fua. Reconstructing loopy curvilinear structures using integer programming. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1822–1829, 2013.

[31] Ching-Wei Wang, Yu-Ching Lee, Hilmil Pradana, Zhi Zhou, and Hanchuan Peng. Ensemble neuron tracer for 3d neuron reconstruction. Neuroinformatics, 15(2):185–198, 2017.

[32] Yimin Wang, Qi Li, Lijuan Liu, Zhi Zhou, Zongcai Ruan, Lingsheng Kong, Yaoyao Li, Yun Wang, Ning Zhong, Renjie Chai, et al. Teravr empowers precise reconstruction of complete 3-d neuronal morphology in the whole brain. Nature communications, 10(1):1–9, 2019.

[33] Yu Wang, Arunachalam Narayanaswamy, Chia-Ling Tsai, and Badrinath Roysam. A broadly applicable 3-d neuron tracing method based on open-curve snake. Neuroinformatics, 9:193–217, 03 2011.

[34] Johan Winnubst, Erhan Bas, Tiago A Ferreira, Zhuhao Wu, Michael N Economou, Patrick Edson, Ben J Arthur, Christopher Bruns, Konrad Rokicki, David Schauder, et al. Reconstruction of 1,000 projection neurons reveals new cell types and organization of long-range connectivity in the mouse brain. Cell, 179(1):268–281, 2019.

[35] Hang Xiao and Hanchuan Peng. App2: automatic tracing of 3d neuron morphology based on hierarchical pruning of a gray-weighted image distance-tree. Bioinformatics, 29(11), June 2013.
[36] Jun Xie, Ting Zhao, Tzumin Lee, Eugene Myers, and Hanchuan Peng. Anisotropic path searching for automatic neuron reconstruction. *Medical image analysis*, 15(5):680–689, 2011.

[37] Jian Yang, Ming Hao, Xiaoyang Liu, Zhijiang Wan, Ning Zhong, and Hanchuan Peng. Fmst: an automatic neuron tracing method based on fast marching and minimum spanning tree. *Neuroinformatics*, 17(2):185–196, 2019.

[38] Zhi Zhou, Hsien-Chi Kuo, Hanchuan Peng, and Fuhui Long. Deepneuron: an open deep learning toolbox for neuron tracing. *Brain informatics*, 5(2):1–9, 2018.