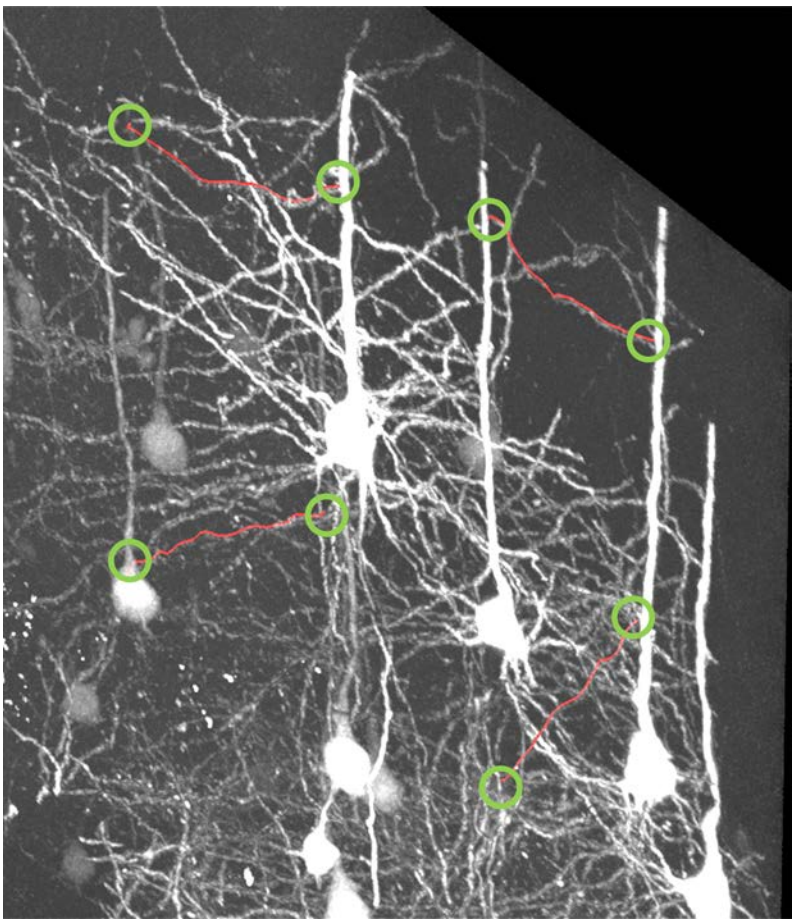
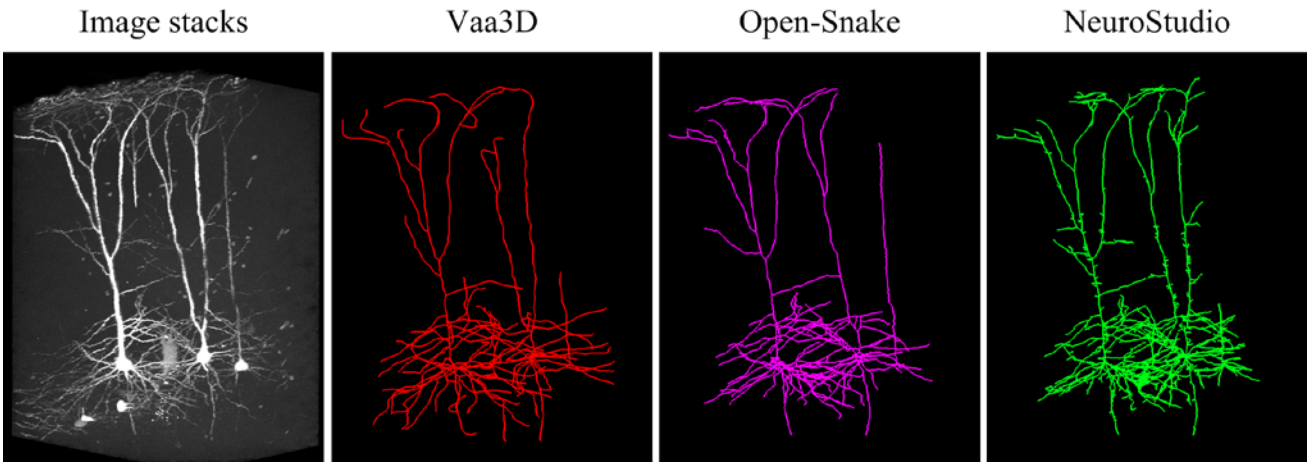
Supplementary Figure 1



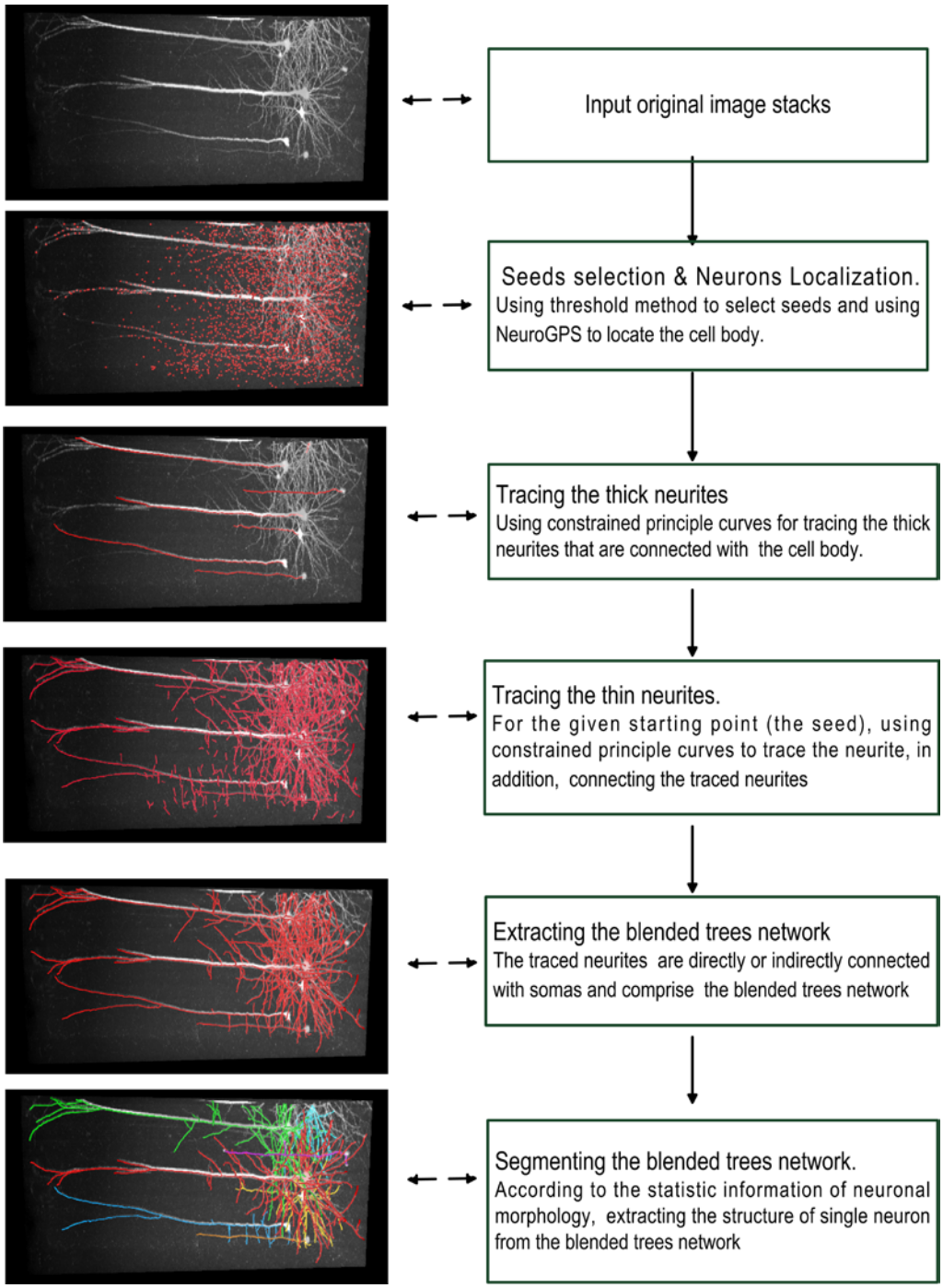
Supplementary Figure 1 Spurious connections between neurites make the neuronal trees blend together. Both ends of some neurites (red curve) have two links (light green circle) with other neurites, such that the reconstructed neuronal trees cannot be separated.

Supplementary Figure 2



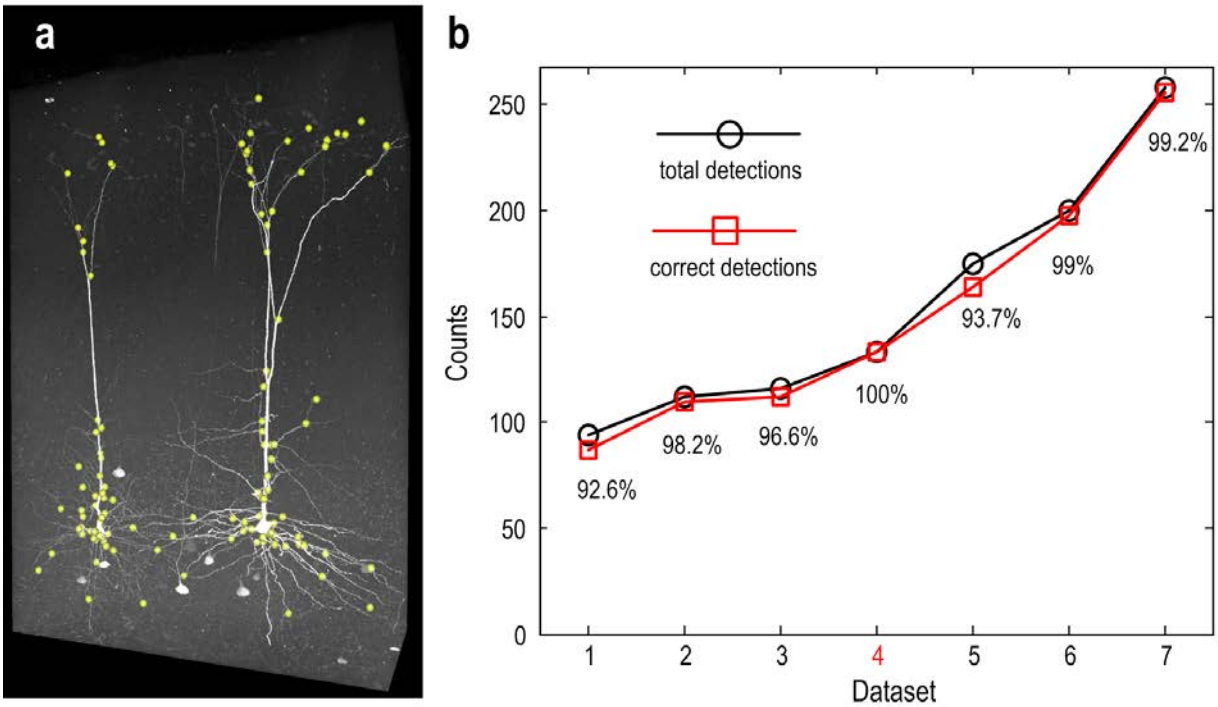
Supplementary Figure 2 Using the image stack of a small-scale neuronal population to test some of the best available methods. These reconstructed results indicate that these methods wrongly regarded three neuronal trees as a single neuronal tree and failed to identify individual neuronal trees of the neuronal population, as they are not designed to separate individual trees.

Supplementary Figure 3



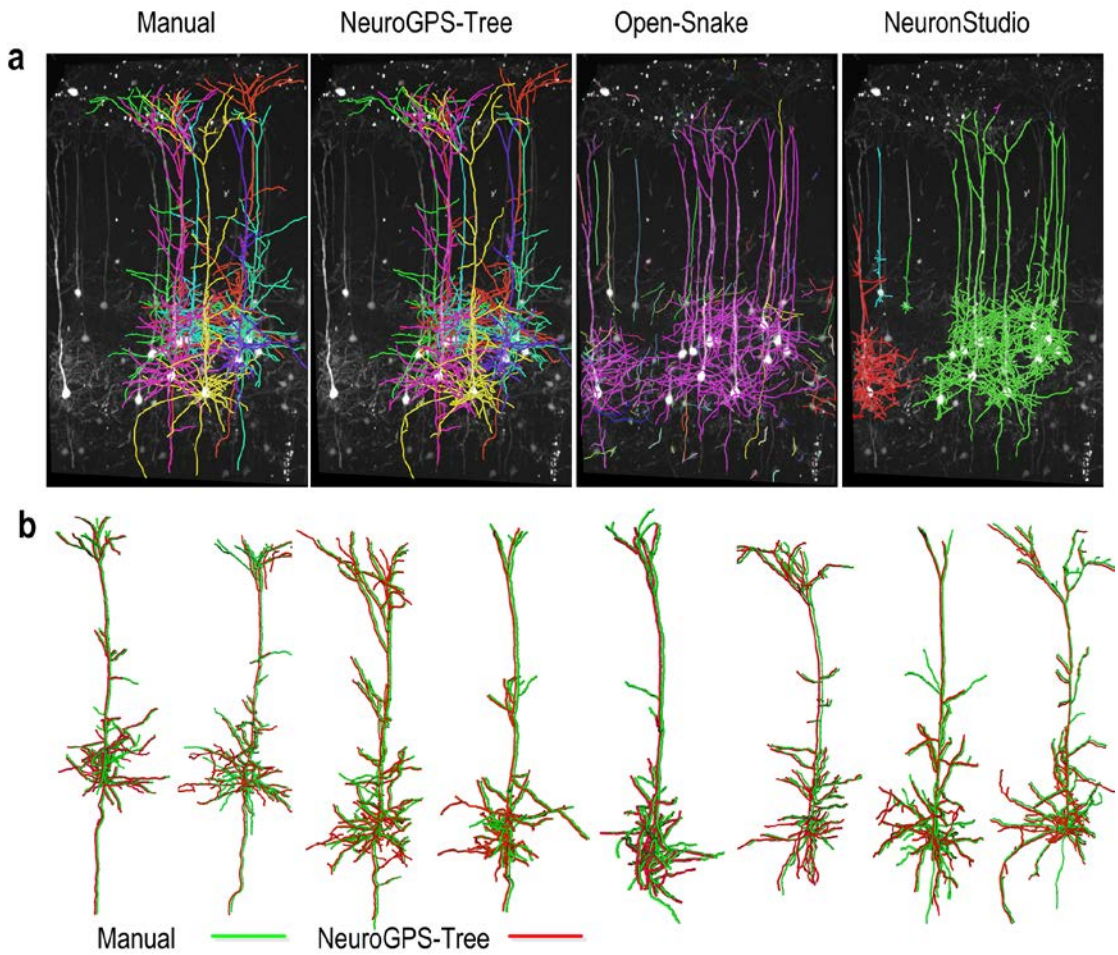
Supplementary Figure 3 The entire routine of NeuroGPS-Tree for the reconstruction of the neuronal population.

Supplementary Figure 4



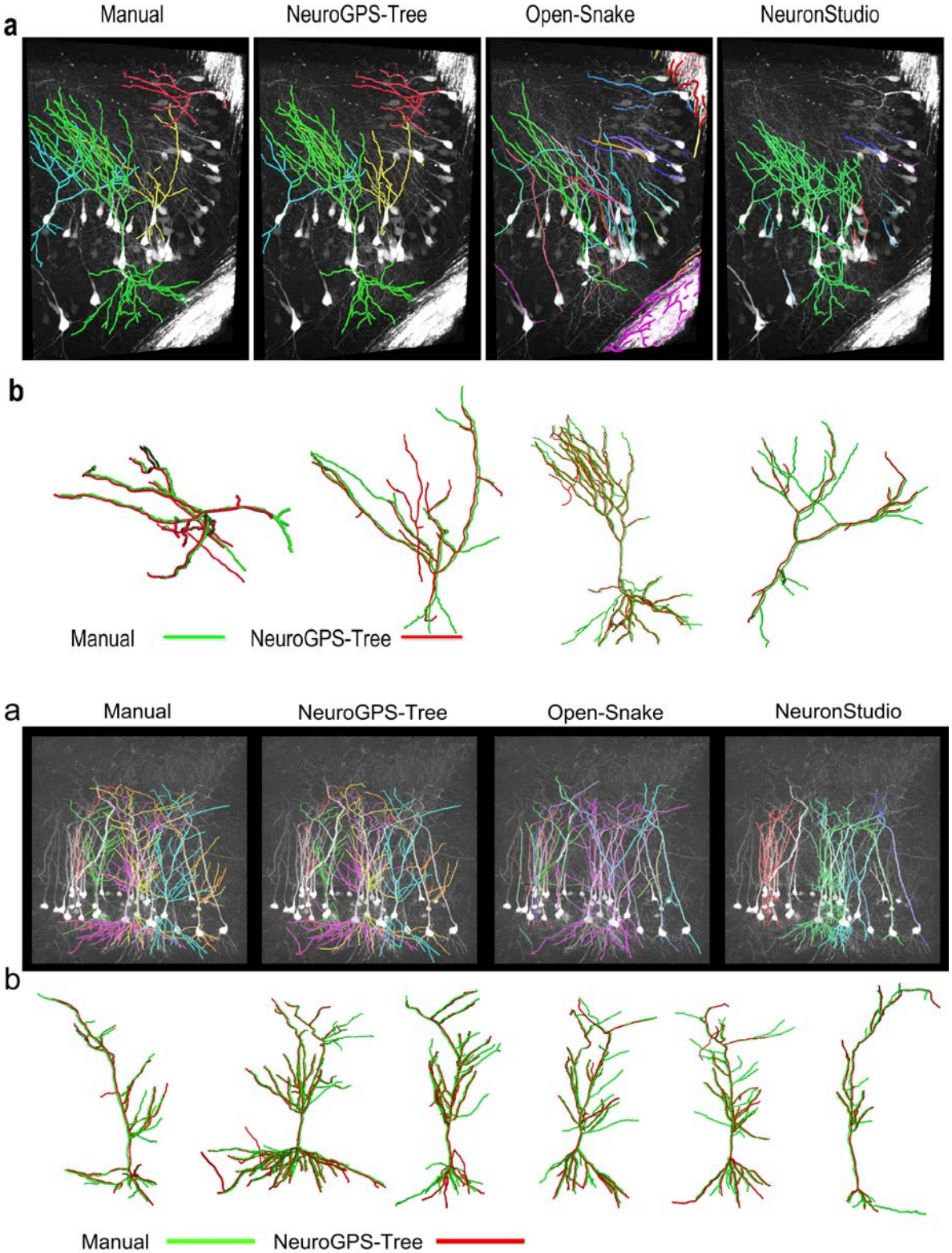
Supplementary Figure 4 Using the discriminant rule to identify the direction of a neurite with high accuracy. (**a**) The direction of neurites, which points towards the terminal end of a neurite, is determined using the discriminant rule. The yellow points represent the identified starting ends of neurites. (**b**) Seven experimental datasets used to verify the effectiveness of the discriminant rule. The red number 4 on the horizontal axis represents the dataset shown in (**a**). 'Total detections' and 'correct detections' represent the number of neurites whose directions can be determined using the discriminant rule (**Online Methods**), and the number of correctly determined neurites verified by manual inspection, respectively.

Supplementary Figure 5



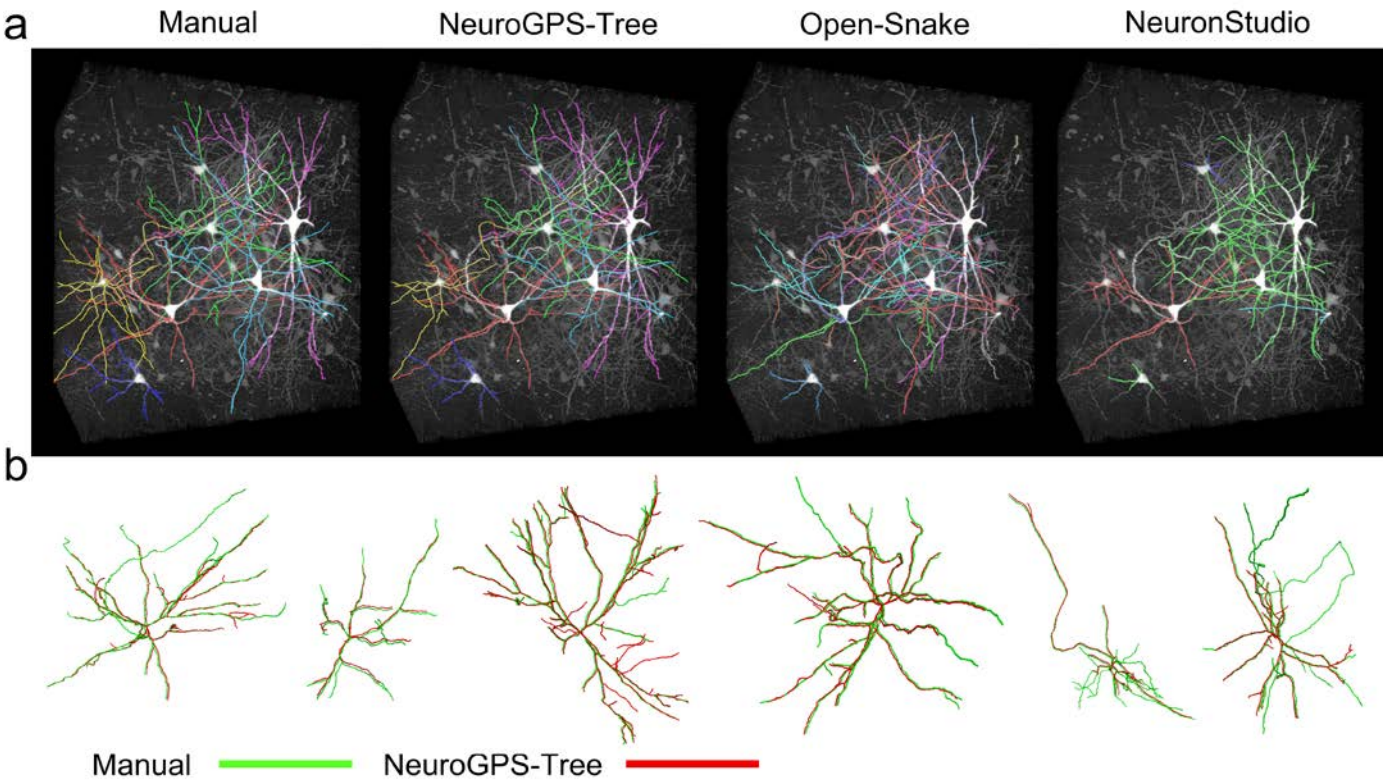
Supplementary Figure 5 Comparison of reconstructions derived manually and from NeuroGPS-Tree, Open-Snake and NeuronStudio (I). (**a**) The testing dataset originatedfrom the cortical region of a mouse brain. From the reconstructed neuronal population, we selected and displayed the neuronal trees with plentiful neuritis; these selected neuronal trees are identified in different pseudo colors. The results derived from Open-Snake and NeuronStudio show that individual neuronal trees cannot be identified. (**b**) The reconstructed results derived manually and from NeuroGPS-Tree. Note that the corresponding precision, recall rates and DIADEM score are shown in **Fig. 2d** dataset 3.

Supplementary Figure 6



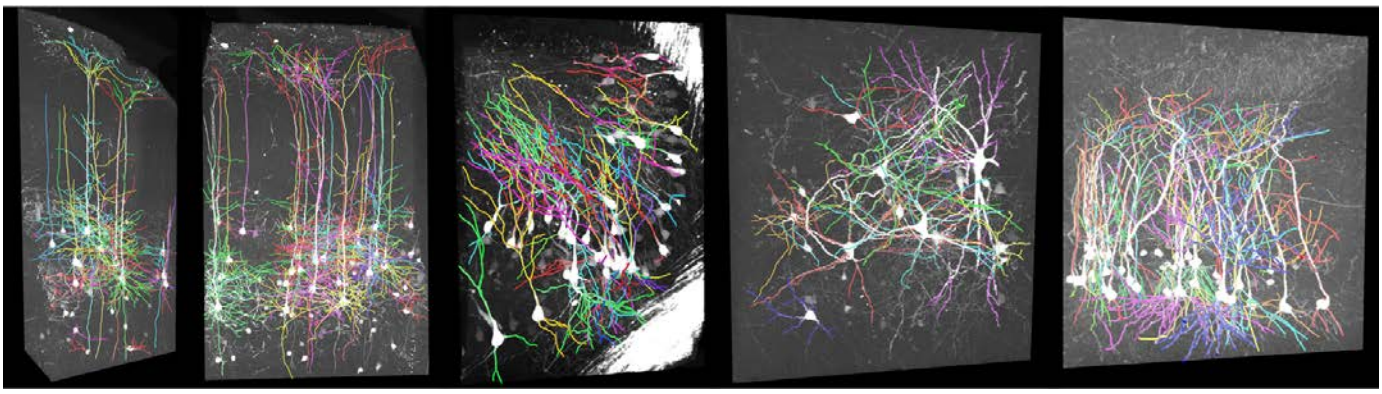
Supplementary Figure 6 Comparison of reconstructions derived manually and from NeuroGPS-Tree, Open-Snake and NeuronStudio (II). (**a**) The two testing datasets originated from the hippocampus of a mouse brain. The reconstructed neuronal trees with plentiful neurites are identified in different pseudo colors. (**b**) The reconstructed results derived manually and from NeuroGPS-Tree. Note that the corresponding precision, recall rates and DIADEM score are shown in **Fig. 2d** datasets 1 and 4.

Supplementary Figure 7



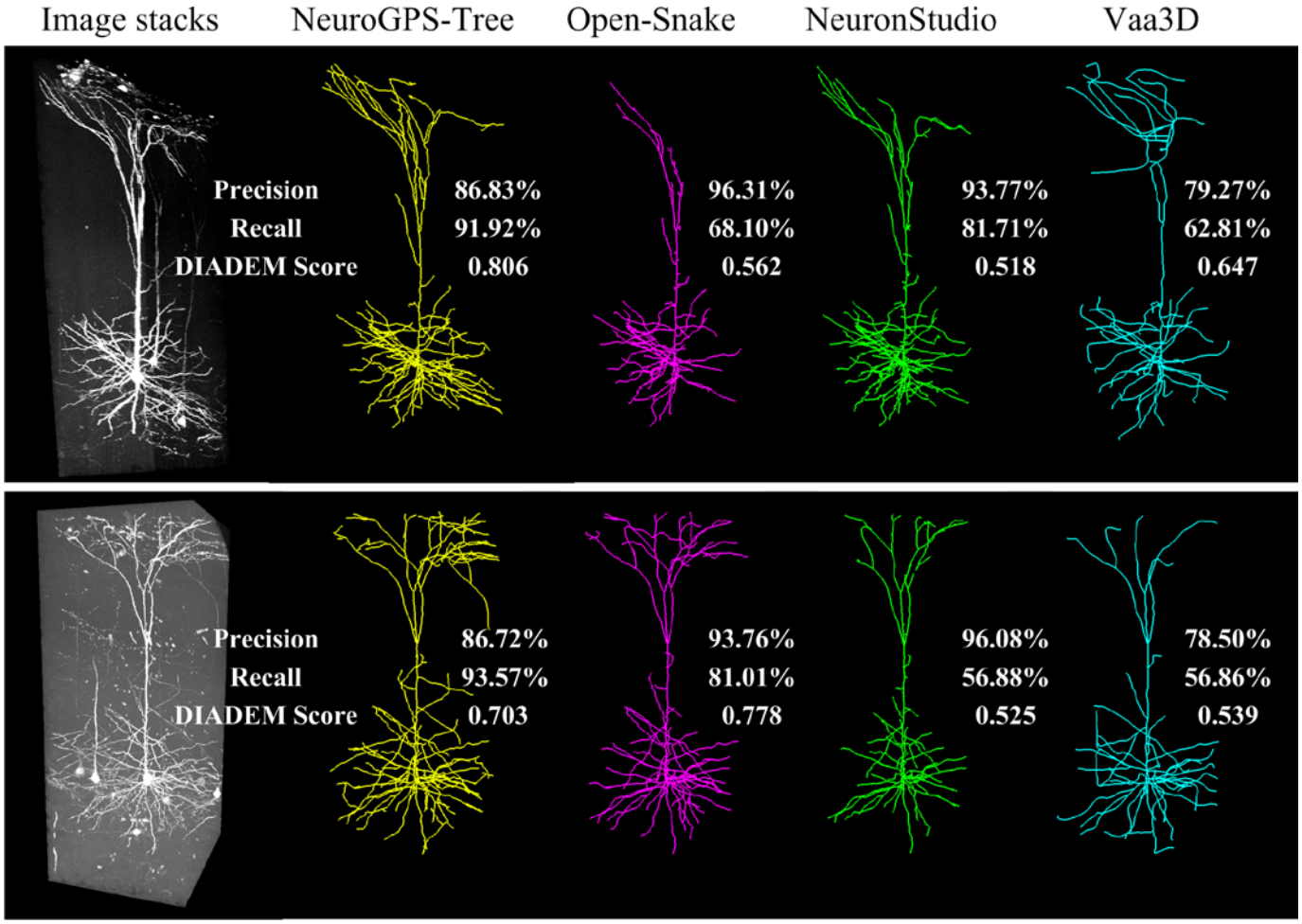
Supplementary Figure 7 Comparison of reconstructions derived manually and from NeuroGPS-Tree, Open-Snake and NeuronStudio (III). (**a**) The testing dataset originated from the thalamic nucleus of the mouse brain. The reconstructed neuronal trees with plentiful neurites are identified in different pseudo colors. (**b**) The reconstructed results derived manually and from NeuroGPS-Tree. Note that the corresponding precision, recall rates and DIADEM score are shown in **Fig. 2d** dataset 5.

Supplementary Figure 8



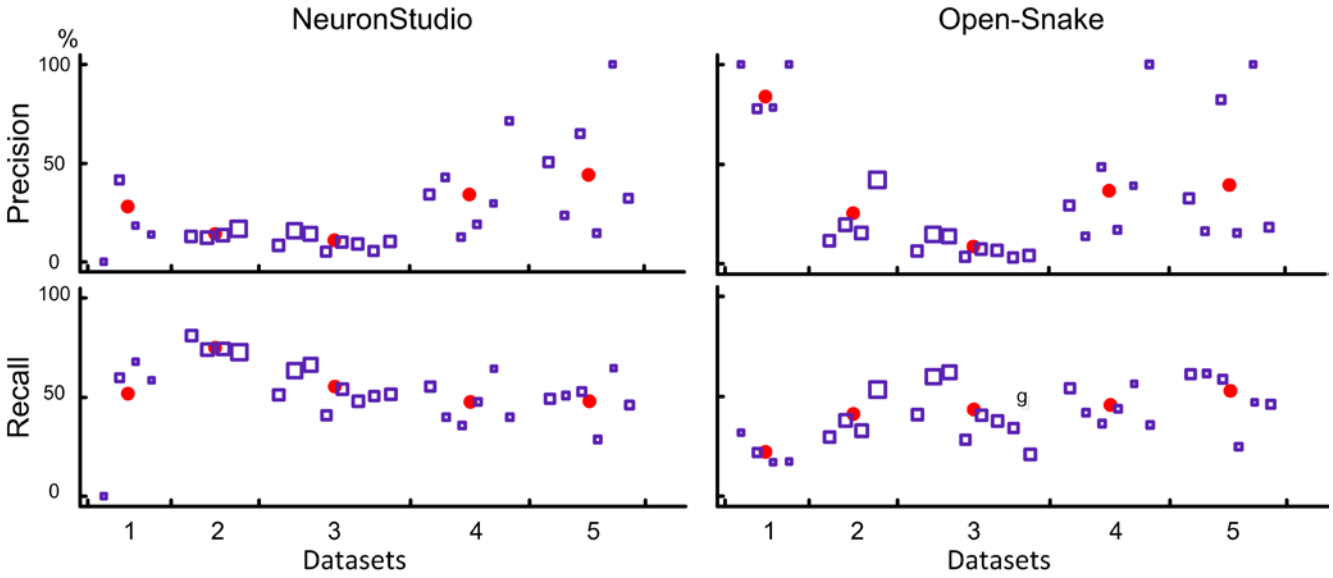
Supplementary Figure 8 The reconstruction of small neuronal populations imaged with two-photon laser scanning microscopy. Five groups of datasets as examples of the neuronal population reconstruction derived from NeuroGPS-Tree. Most of the neuronal trees in the reconstructed neuronal population are displayed and identified by pseudo colors.

Supplementary Figure 9



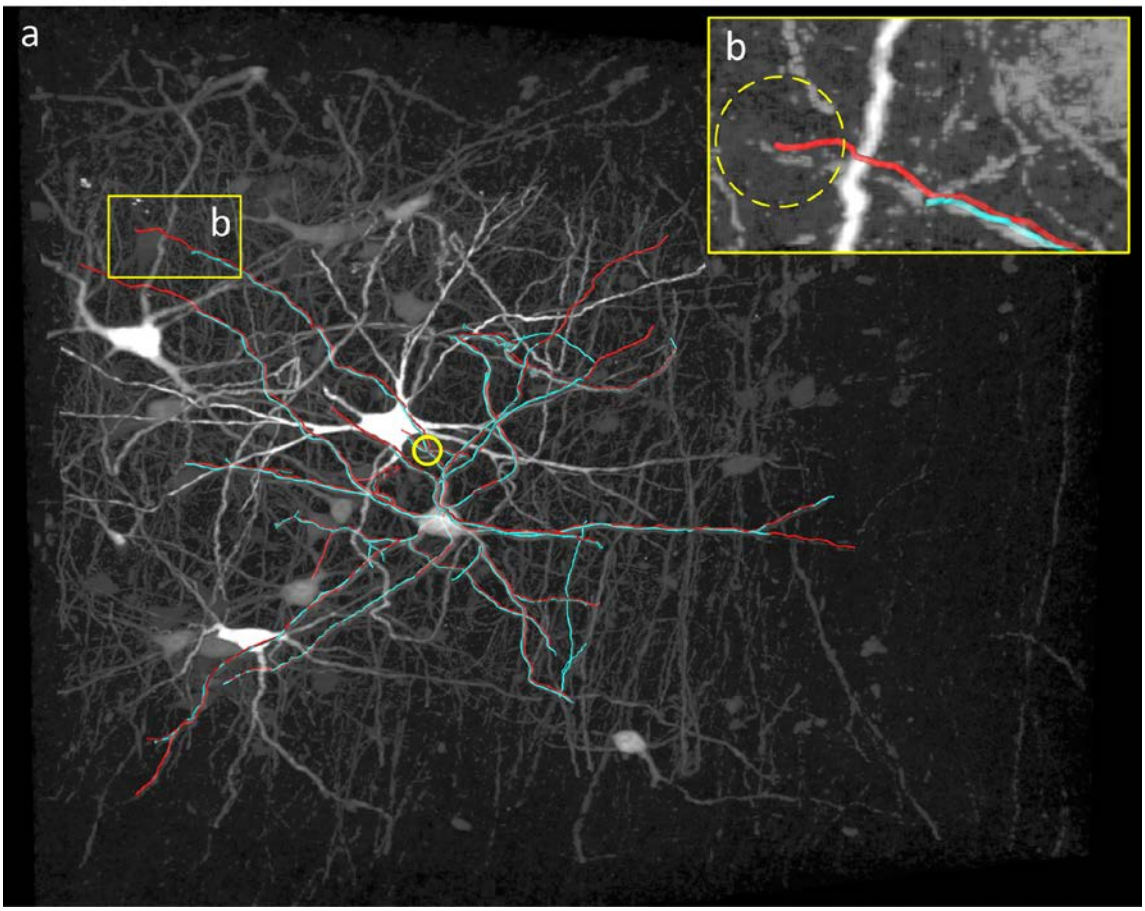
Supplementary Figure 9 Comparison of single neuron reconstructions derived from several methods. Row 1–2: two examples with neurites mostly from one neuron; these reconstructions are similar.

Supplementary Figure 10



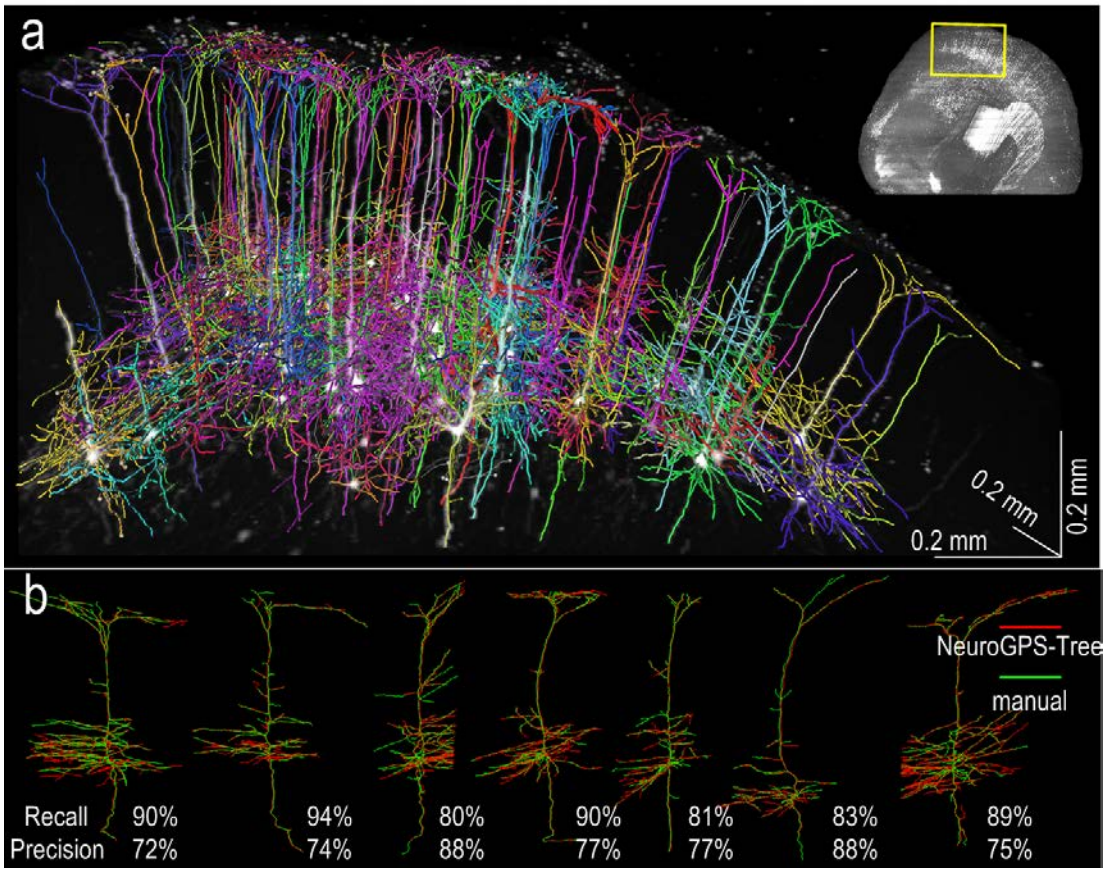
Supplementary Figure 10 Evaluation of the reconstructions derived from NeuronStudio and Open-Snake with precision and recall rates. The test image stacks are shown in **Supplementary Fig. 8**. There are two steps in computing the recall and precision rates: I) manually as the ground truth, searching for its optimally matched network in the reconstructions generated by NeuronStudio and Open-Snake; II) quantifying the skeletal difference between the matched network and the ground truth (see **Online Methods**). For the reconstruction of the neuronal population, NeuronStudio and Open-Snake failed to separate individual neurons, and thus generated a reconstruction with an extremely low precision rate. In addition, the recall rate was also far less than that of NeuroGPS-Tree (approximately 80%, as shown in **Fig. 2d**), mainly because NeuroGPS-Tree pays more attention to high-dynamic-range images than NeuronStudio and Open-Snake.

Supplementary Figure 11



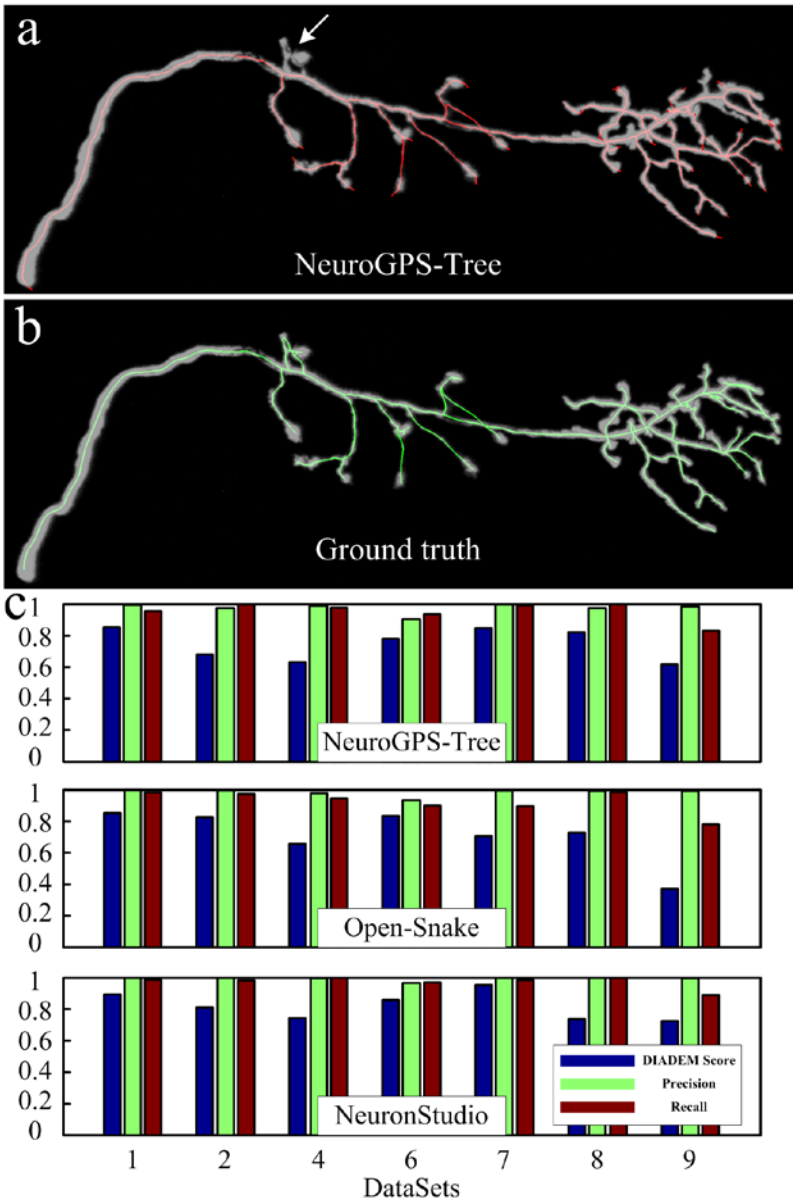
Supplementary Figure 11 DIADEM score is stricter than the recall and precision rate in quantifying the automated reconstruction. (**a**) For the neurite that starts at the location labeled with a circle, a majority of its automatically reconstructed skeleton curve (light blue) is well matched with manual reconstruction (red), but the automatically reconstructed termination end deviates from the truth as shown in (**b**). In this case, the automated reconstruction of this neurite can be accepted by the recall and precision rate, but not the DIADEM score.

Supplementary Figure 12



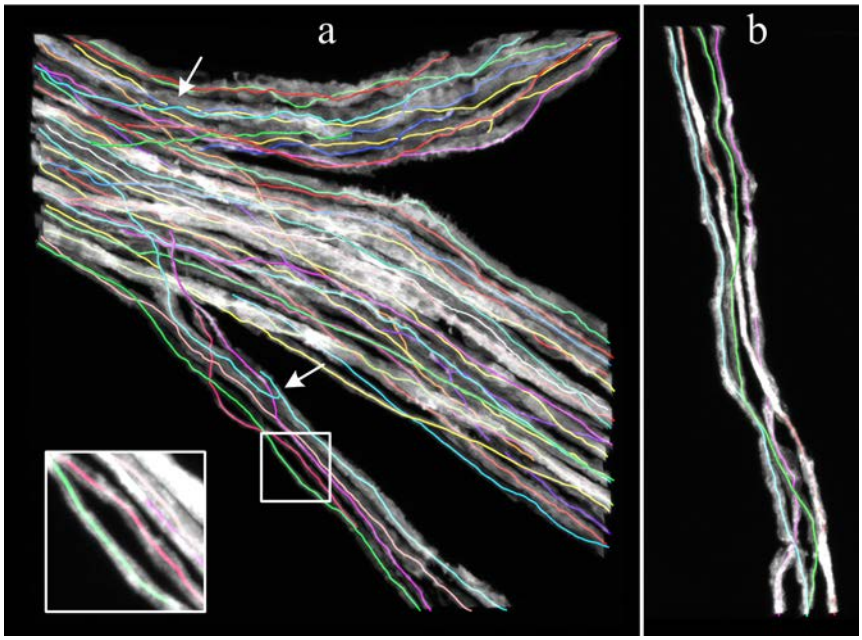
Supplementary Figure 12 The reconstruction of a middle-scale neuronal population. (**a**) From the reconstructed neuronal population, the neuronal trees are identified by pseudo-colors. The inset figure shows a half-coronal profile (top-right). The dataset in the yellow square was used for the reconstruction whose results are shown in (**a**). (**b**) From the reconstruction in (**a**), several typical neuronal trees were selected to provide a comparison of the reconstructions derived from NeuroGPS-Tree (red) and manually (green), respectively.

Supplementary Figure 13



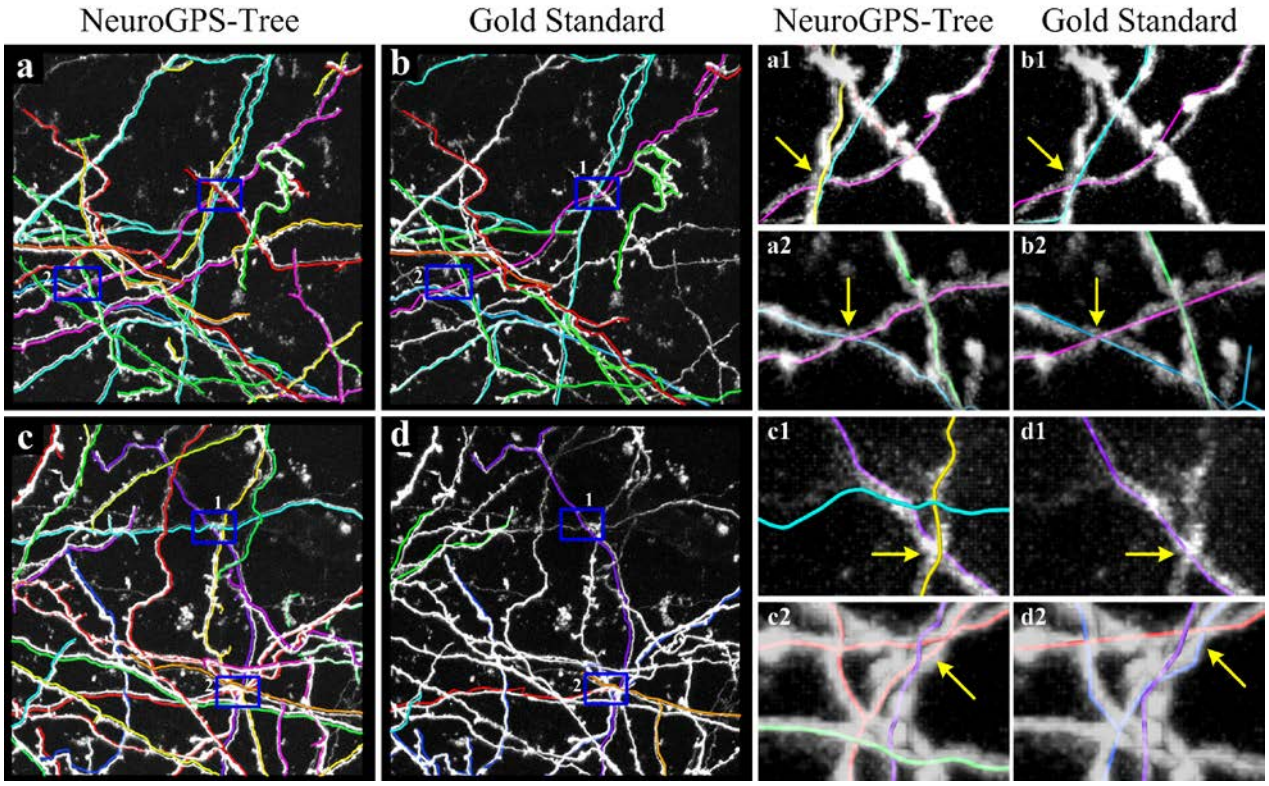
Supplementary Figure 13 The reconstruction from DIADEM datasets with NeuroGPS-Tree (I). (**a**) Olfactory Projection (OP) Fibers dataset (ref. 1) acquired with confocal microscopy was used to test NeuroGPS-Tree. The reconstruction with NeuronGPS-Tree is shown. Some short neurites could not be reconstructed, as labeled by the arrows. (**b**) Ground truth of the dataset. (**c**) The DIADEM score, precision and recall rate of the automated reconstructions of seven tested OP datasets. The DIADEM score was 0.75±0.10 for NeuroGPS-Tree, 0.71±0.16 for Open-Snake, and 0.82±0.09 for NeuronStudio.

Supplementary Figure 14



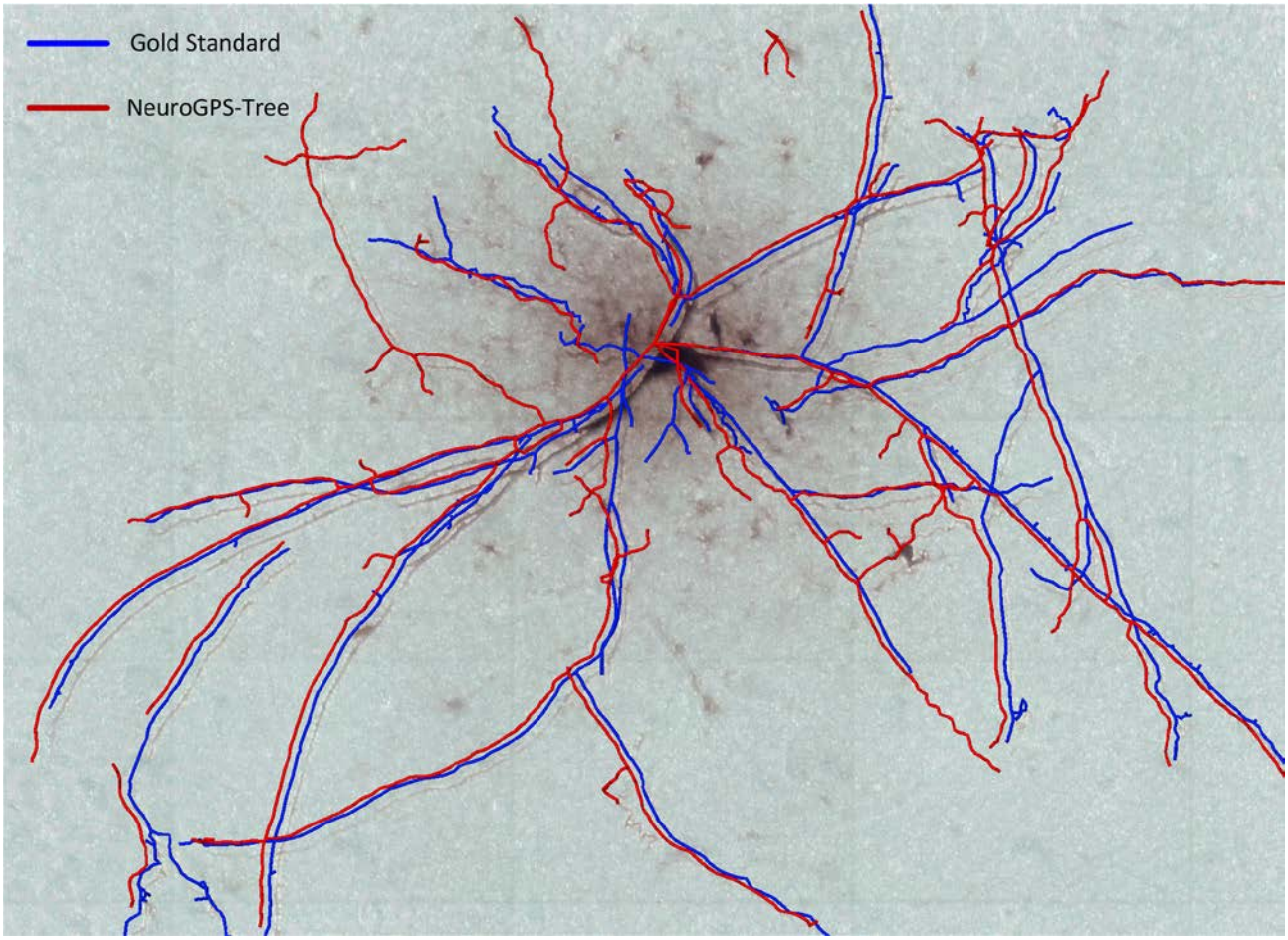
Supplementary Figure 14 The reconstruction from DIADEM datasets with NeuroGPS-Tree (II). Two neuromuscular Projection Fibers datasets (ref. 2) acquired using confocal microscopy were used to test NeuroGPS-Tree. (**a**)-(**b**) The reconstructions derived from NeuroGPS-Tree. The two incorrect reconstructions are labeled with arrows in (**a**). The axons in the rectangle seem to be unrecognizable. In fact, in the enlarged rectangle, by reducing the range of voxel colors, high contrast between foreground and background can be obtained.

Supplementary Figure 15



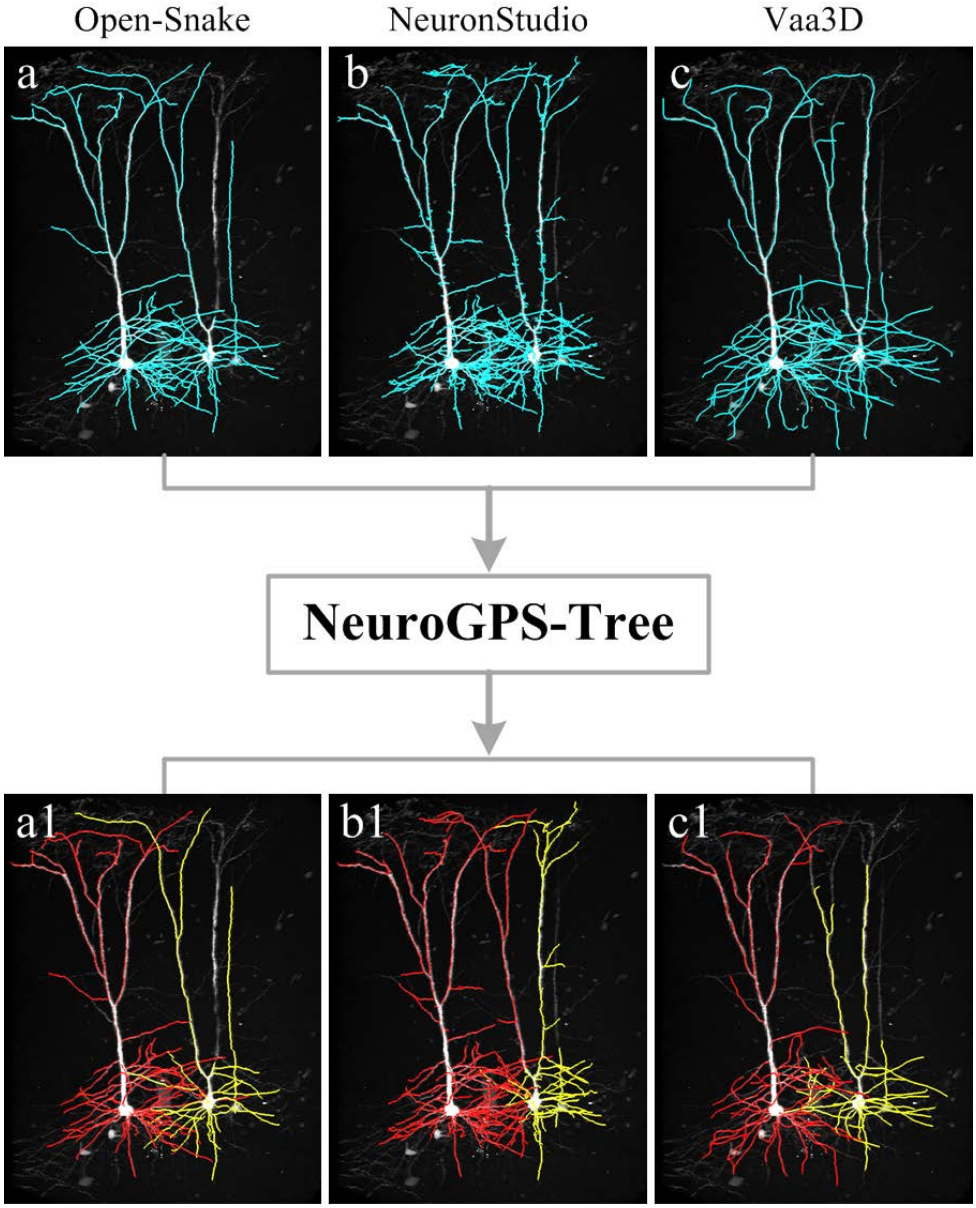
Supplementary Figure 15 The reconstruction from DIADEM datasets with NeuroGPS-Tree (III). Two neocortical Layer 1 Axons datasets (ref. 3) acquired using two-photon laser scanning microscopy. (**a**) The reconstruction derived from NeuroGPS-Tree. (**b**) Gold Standard. (**a1**)-(**b1**) & (**a2**)-(**b2**) The enlargements of the blue rectangles in (**a**)-(**b**), the arrows that indicate the crossings. For the reconstructions of the other dataset, (**c**)-(**d**), (**c1**)-(**d1**) and (**c2**)-(**d2**) are similar to (**a**)-(**b**), (**a1**)-(**b1**) and (**a2**)-(**b2**), respectively.

Supplementary Figure 16



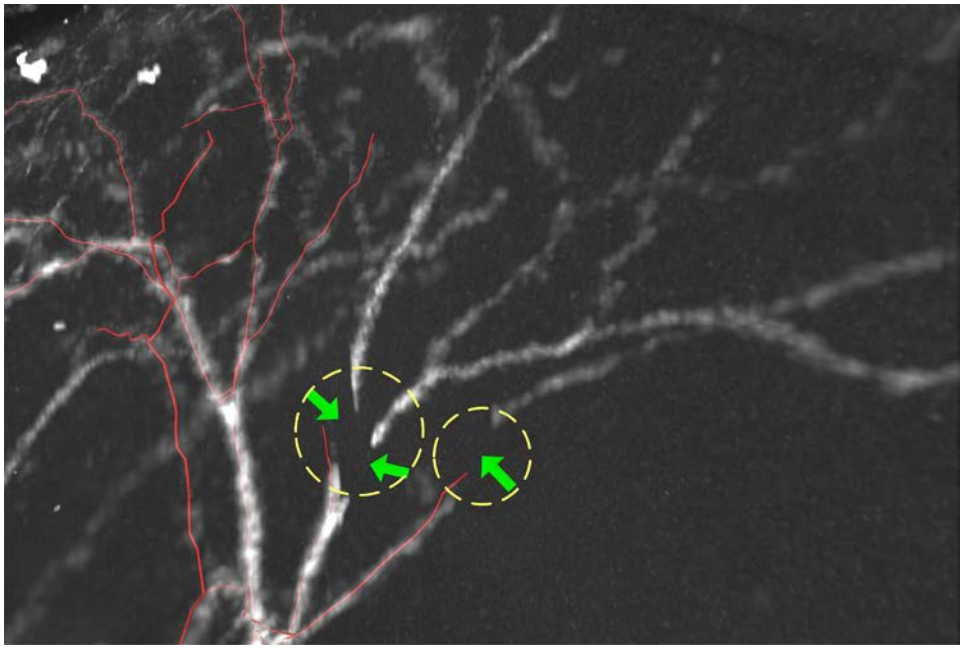
Supplementary Figure 16 The reconstruction from DIADEM datasets with NeuroGPS-Tree (IV). Hippocampal CA3 Interneuron dataset (ref. 4) acquired with transmitted light bright field microscopy used to test NeuroGPS. The reconstructed results derived manually (blue line) and from NeuroGPS-Tree (red line) are shown.

Supplementary Figure 17



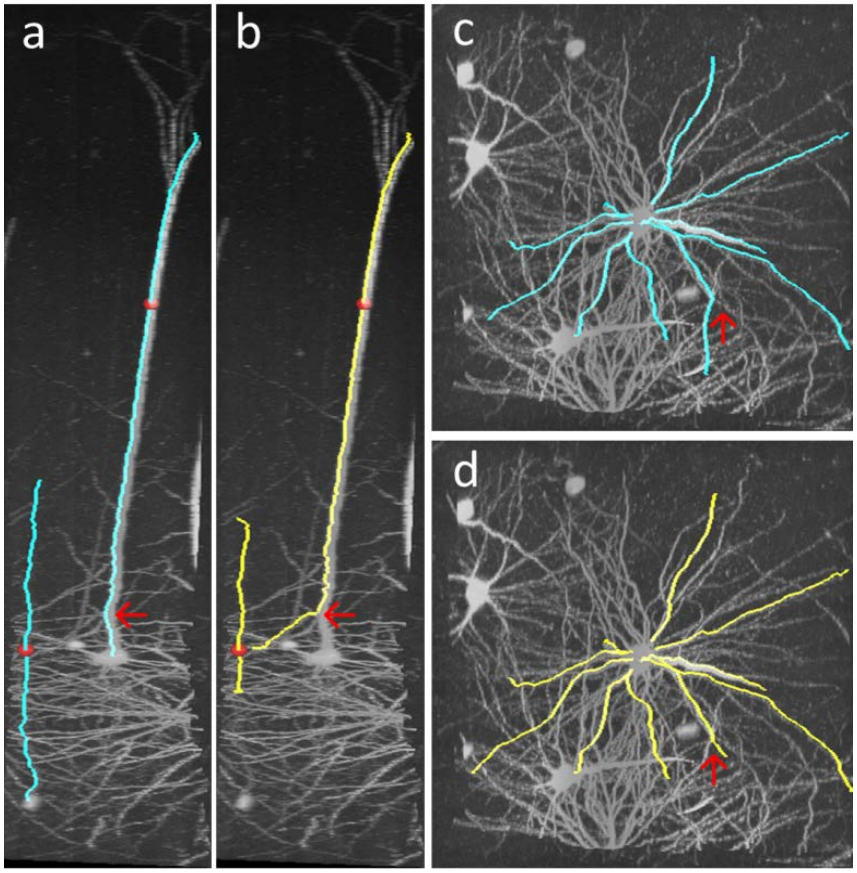
Supplementary Figure 17 The potential compatibility of NeuroGPS-Tree with other suitable tracing methods. Row 1: The reconstructions, derived from Open-Snake, NeuronStudio and Vaa3D respectively, could not provide the separated individual trees, but provided the skeletons of the neurites and the links between two traced neurites. Based on the reconstructed skeletons and links and their adjacent voxel signals in the images, NeuroGPS-Tree could assign the traced neurites to individual neurons, and the reconstructed individual trees that resembled the vast majority of traced skeletons were obtained, as shown in **Row 2**.

Supplementary Figure 18



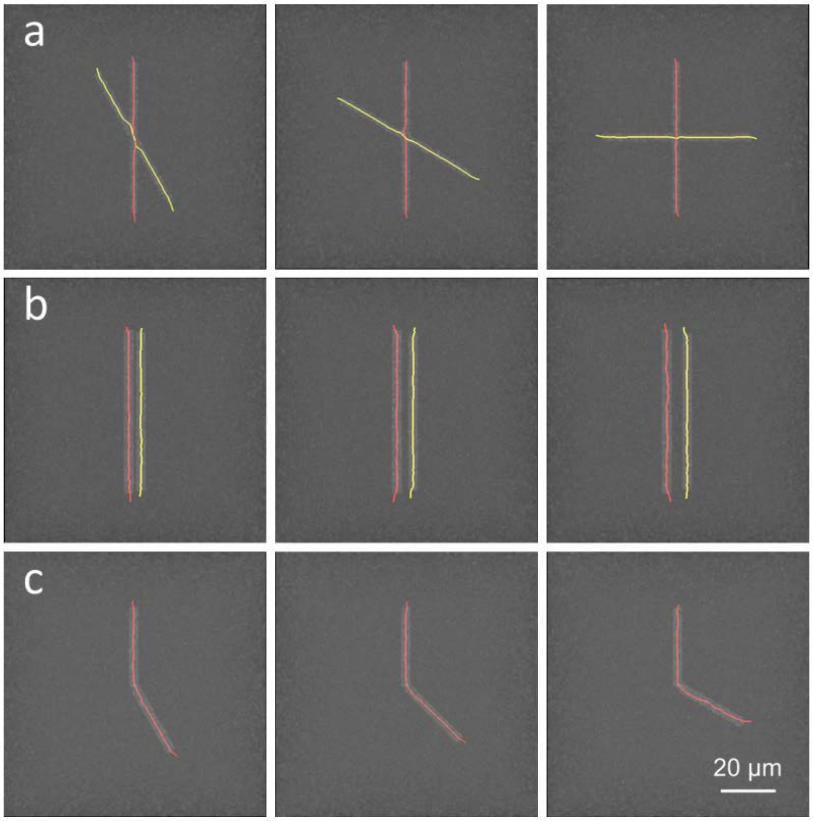
Supplementary Figure 18 The limitation of the tracing procedure in NeuroGPS-Tree. The tracing procedure failed to trace broken neurites where signals are missing for approximately 6 μm, as noted by the green arrows. The red curves represent the skeletons of neurites reconstructed with NeuroGPS-Tree.

Supplementary Figure 19



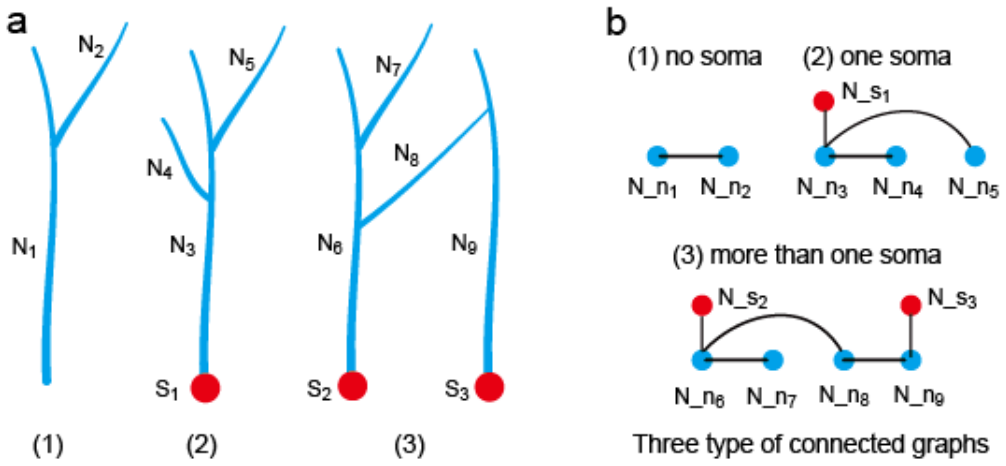
Supplementary Figure 19 Strategies in the tracing procedure for better performance. (**a**)-(**b**) Comparison of the traced neurites derived from a constrained principal curve (**a**) and principal curve (**b**). The red dots represent the seeds for tracing, and the signals indicated by the arrow caused the wrong tracing of the principal curve, which can be correctly analyzed by the constrained principal curve. (**c**)-(**d**) Comparison of the tracing with (**c**) and without (**d**) Rayburst sampling to address the signals whose direction changes sharply, as indicated by red arrows.

Supplementary Figure 20



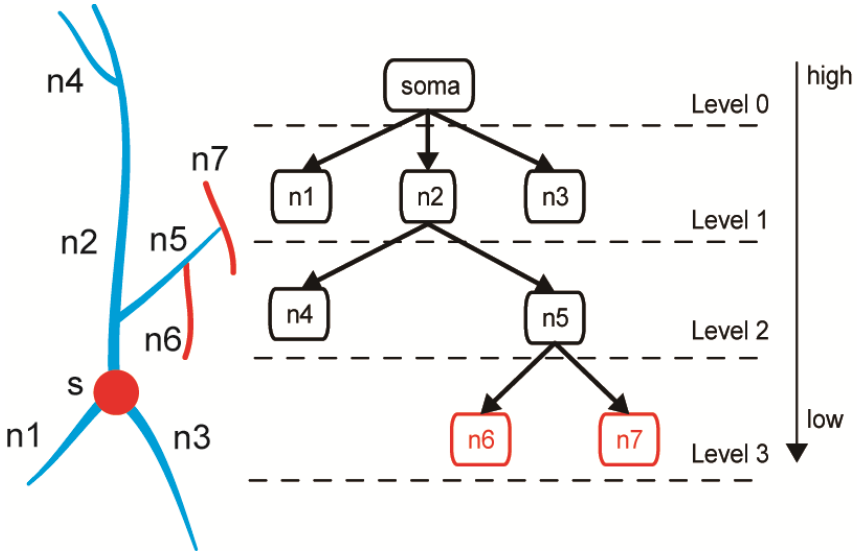
Supplementary Figure 20 Tracing neurites of three typical simulated patterns with NeuroGPS-Tree. The traced results for the patterns in which two neurites cross (**a**), two neurites are positioned in parallel (**b**), and the direction of a single neurite changes sharply (**c**). The simulated image stacks in (**a**)-(**c**) have the same size of 100 × 100 × 100 volume pixels, and their max-intensity projections are shown.

Supplementary Figure 21



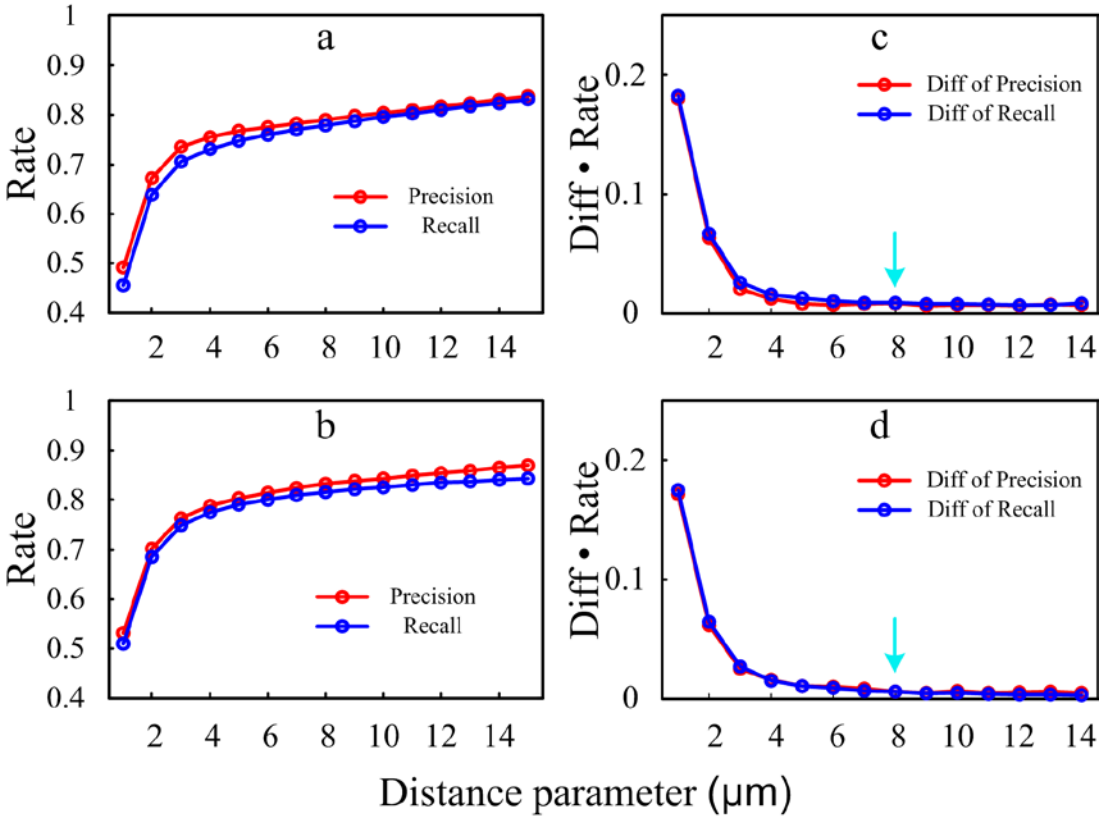
Supplementary Figure 21 The reconstructed networks and their corresponding connected graphs. (**a**) Three types of reconstructed networks all with the corresponding connected graphs. Both Type (2) and (3) contain the positioned somas (red dots), the traced neurites and the link between two traced neurites. (**b**) Three types of connected graph: (1) containing no soma node, (2) only containing one soma node, (3) containing more than one soma node. Neuronal population reconstruction is essential to decompose Type (3) connected graph into Type (2) connected graphs. Red dot represent soma node; blue, neurite node.

Supplementary Figure 22



Supplementary Figure 22 Detecting spurious links in a single neuron. (**Left**) A single neuronal tree in the cartoon, extracted from the reconstructed network, consists of the soma position, the skeleton of the traced neurites and the links between traced neurites and between the soma and the traced neurites. The redundant neurites are labeled with in red. (**Right**) A hierarchical tree structure obtained from a single reconstructed tree. Using this structure, the redundant neurites are identified and deleted. As shown, the end of neurite n5 in the level 2 node contacts the middle part of neurite n7 in the lower node (level 3). In this case, the neurite n7 is regarded as a redundant neurite.

Supplementary Figure 23



Supplementary Figure 23 The precision and recall rate as a function of the distance parameter. (**a**)-(**b**) The precision and recall rates of the automated reconstruction of two neurons in **Fig. 2** as a function of the distance parameter. This distance parameter is used to judge whether a point in the machine-reconstructed skeleton is equated to a point in the human-reconstructed skeleton. (**c**)-(**d**) The differences in the precision and recall rate curves for the two neurons.

Supplementary Note 1

1. Seeds selections and soma detection

For the given original image stacks *Imag_s*, min(*Imag_s*, *thre*) is convolved 20 times with the averaging template ($9 \times 9 \times 1$ pixels^3), and the convolved image stacks *Imag_b* is regarded as the background, while *thre* is a roughly estimated value that can distinguish foreground and background. min(*Imag_s, thre*) is the operation in which the gray-level values of *Imag_s* more than *thre* are replaced by *thre*. After computing *Imag_b*, we use our proposed methods to locate the soma (ref. 5) and label the volume pixels in the interior soma (ref. 6). For all unlabeled volume pixels, we select some as the seed points, denoted as o, according to the prefix conditions in which the signal value *Imag_s*(o) was the local peak value in the cube centered on the position o and with a size of $11 \times 11 \times 11$ volume pixels. *Imag_s*(o) was more than the sum of *Imag_b*(o) with six times the square root of *Imag_b*(o). Finally, we sorted the seed points according to their corresponding signal intensity in descending order.

2. Neurite tracing

In this section, we first provide the brief scheme of single neurites tracing, which involve in the complicated procedures of some steps. Second, we provide a detailed description of these complicated procedures. Finally, we described the procedure used to trace all neurites.

The scheme of the tracing of neurite

Starting from the initial point, i.e., the seed point o_c, we traced a single neurite as follows:

Step 1 Set $o^m = 1$, $m = 1$, $o^1 = o_c$

Step 2 Extracting the local signal: computing the prior direction of o^m, denoted by r_p, and using the linear window determined by o^m and r_p to extract the signal in the neighborhood of o^m.

Step 3 Correcting the position of o^m: using constrained principal curves to analyze the extracted signal. The adjusted position of o^m and the direction of the extracted signal are computed and denoted as o^m and r^*, respectively.

Step 4 Computing the preferred direction of o^m: selecting the linear window with the strongest signal from the linear windows in which the starting position is o^m and whose direction r ensures that the inner product of r and r^* is greater than 0.85. The direction of the selected linear window is the preferred direction of o^m, given by r^m.

Step 5 $o^m + 1.5r^m \rightarrow o^{m+1}$, $m + 1 \rightarrow m$, repeat ***Step*** 2–4 until the tracing termination conditions are satisfied.

Note 1. One portion of the traced positions of a neurite is generated for the initial setting (o^1, r_p). The other portion is generated for the setting (o^1, $-r_p$). These two traced positions are merged to the complete centerline of the neurite. Then, the above procedures are carried out for another seed point not in the traced neurites. By repeating the tracing procedure until all seed points are traversed, the centerlines of all neurite from image stack can be generated.

Note 2. The description of the process used to compute the prior direction (***Step*** 2) of the position o^1 is in **Section 2.1** and that of the other points is in **Section 2.2**. The description of the extraction of the local signal in ***Step*** **2** is in **Section 2.3**. **Section 2.4**, **2.1** and **2.5** provide the detailed descriptions of ***Step*** 3, 4 and 5.

2.1 Description of linear window method

For the given point o_c, its linear window can be described as:

$$S(\phi,\theta) = \bigcup_{t\in T} s(\phi,\theta,t) = \bigcup_{t}\{o \mid o \in \delta(o_c + tr_{\phi,\theta})\} \tag{1}$$

where $r_{\varphi,\,\theta}$ is the directional vector, representing ($sin\varphi cos\theta$, $sin\varphi sin\theta$, $cos\varphi$), $\delta(o)$ represents the neighborhood of the position o, T represents the selected subset on the integer domain. The $\delta(o)$ and T are used to control the size of linear window. The signal intensity of the linear window $I(\varphi, \theta)$ is calculated as the averaged values of $I(\varphi, \theta, t)$ for all t belonging to T. $I(\varphi, \theta, t)$ represents the average signal intensity of the neighborhood of the position $o_c + tr_{\varphi,\,\theta}$. To confirm the linear window with the maximum signal intensity, we discretized the two parameters φ and θ and obtain the grid points (φ_i, θ_j) = ($i\pi/n_1$, $j\pi/n_2$) for i = 1, 2, …, n_1; j = 1, 2, …, n_2. n_1 and n_2 are set to 12 and 9, respectively, for our analysis. For each grid point, we compute the signal intensity $I(\varphi_i, \theta_j)$ of linear window. From these values, we selected the maximum value expressed by $I(\varphi^*, \theta^*)$ whose corresponding direction $r(\varphi^*, \theta^*)$, equal to ($sin\varphi^* cos\theta^*$, $sin\varphi^* sin\theta^*$, $cos\varphi^*$), was also confirmed.
The initial prior direction of o^1 in ***Step*** **2** was the direction of the linear window with the maximum signal intensity. The parameters were as follows: $\delta(o)$ represents the position o and its 26 neighborhood. T contains the integer numbers ranging from –8 to 8. Here, the large size of linear window was set to obtain a stable prior direction of o^1. In the extraction of the local signals in ***Step*** **2** and ***Step*** **4**, the parameters of linear window were as follows. $\delta(o)$ represents the position o and its 7 neighborhood. T contains the integer numbers ranging from –3 to 3. This setting, with a small

linear window, remains valid due to the constraints of prior direction, while avoiding the interference of other signals.

2.2 Computing the prior direction

As described in **Section 2.1**, we use linear window method to obtain the prior direction of o^1. From the point o^m (1< m <5), the prior direction of each point was the average of the prior direction of o^1 and the direction computed from the extracted local signal of the point o^m (***Step* 3**). For the point o^m (m > 4), its prior direction was computed as:

$$r_p = \frac{1}{n_{scal} m_0} \sum_{j=m-m_0}^{m-1} \frac{\overrightarrow{o^j o^{j+1}}}{\| \overrightarrow{o^j o^{j+1}} \|_2} \tag{2}$$

where m_0 is the maximum value of m–10 and 1, n_{scal} is a normalized parameter that keeps r_p as a unit vector. We hope to obtain the robustness of the prior direction for each point in the tracing process, i.e., the prior directions of the neighboring points should change slowly. Obviously, for a small number of m, carrying out equation (2) cannot maintain robustness in computing the prior directions. Therefore, we modified the method of computing for the prior direction of several initial points.

2.3 Using the linear window method to extract the signal of the neighborhood

To extract the signal of the neighborhood of position o^m, the linear window method was used as described below. We discretized both the polar angle and azimuth angle, as described in **Section 2.1**, and the corresponding linear windows were produced by fixing the values of T. Among these generated linear windows corresponding to T, from integers ranging from –3 to 0, we select those whose directions denoted by r to satisfy the inner product of r and to produce r_p less than –0.95, with r_1 as the direction of the one of the selected linear windows with the maximum signal intensity. When T was set to integers ranging from 1 to 3, r_2 was obtained by the same procedure as r_1 with the inner product of r_2 and r_1 greater than 0.95. After obtaining the directions r_1 and r_2, we could define the neighborhood of o^m, expressed as:

$$S = \left\{ \bigcup_{t_1} \{ o \mid \left\| o - o^m - t_1 r_1 \right\| \le 2 \} \right\} \cup \left\{ \bigcup_{t_2} \{ o \mid \left\| o - o^m - t_2 r_2 \right\| \le 2 \} \right\} \tag{3}$$

with t_1 representing the integers ranging from –3 to 0 and t_2 representing the integers ranging from 1 to 3. From the set S, we derive two sets, expressed as:

$$S_m = \{ o \mid o \in S, Imag_s(o) \ge Imag_b(o) \} \text{ and } M = \{ Imag_s(o) - Imag_b(o) | o \in S_m \}$$

respectively. S_m and M were used to correct the position of o^m (***Step*** 2–3).

2.4 Constrained principal curve to correct the position

Based on principal curves (ref. 7), we proposed constrained principal curves to correct the position of o^m. We briefly describe principal curves model. According to the position $o^m$and its signal (S_m, M), the gradient of kernel density estimate derived from (S_m, M) is calculated as:

$$g = \frac{\sum_{o \in S_M} c(o) A d(o)}{\sum_{o \in S_M} c(o)}$$

where A is the diagonal matrix whose elements in the main diagonal are all set as 0.15 in our analysis, $d(o)$ equals $o–o^m$, $c(o)$ is weight parameter and is defined as:

$$c(o) = \begin{cases} M(o)\exp(-d(o)^T A d(o)/2) & if \ \| d(o) \| < 4 \\ 0 & otherwise \end{cases}$$

with $d^T(o)$ denoting the transpose of $d(o)$. The averaging deviation of o^m was calculated as:

$$d_{aver} = \left(\sum_{o \in S_M} c(o) A\right)^{-1} \left(\sum_{o \in S_M} c(o) A d(o)\right)$$

According to the above three equations, the matrix used to describe the anisotropic of the signal (S_m, M) was:

$$H_g = \frac{-1}{\sum_{o \in S_M} c(o)} \sum_{o \in S_m} c(o)(d(o)d(o)^T - A) + gg^T \tag{4}$$

By the matrix H_g, the principal curves model could be expressed as (ref. 7):

$$p_{c1} = \arg\max_{P^T P = 1} p^T H_g p \tag{5}$$

Considering that the principal direction of signal (S_m, M) should be similar to both its prior direction r_p and the direction of its corresponding linear window r_2, the optimization model (5) could be modified to:

$$p_{c1} = \arg\max_{\substack{p^T p = 1, p^T r_2 \geq 0.9 \\ p^T r_p \geq 0.9}} p^T H_g p \tag{6}$$

This modified model can be called constrained principal curve. Solving optimization problem (6), we obtained p_{c1} and used the algebraic method to compute p_{c2} and p_{c3} based on the fact that any two of these three directions are perpendicular to each other. Using p_{c2}, p_{c3} and d_{aver}, the position o^m can be corrected as:

$$o^m = o^m + \left(p_{c2}^T p_{c2} + p_{c3}^T p_{c3}\right) d_{aver} \tag{7}$$

where p_{c1} is the principal direction of o^m and is also regarded as the principal direction of signal (S_m,

M).

2.5 The termination conditions of tracing

Before correcting the position of o^m in the tracing process, we averaged the twenty biggest values in a set of M. If the mean value was less than the given threshold value, a maximum of 0.3 times background $Imag_b(o^m)$, 4.5 times the square root of $Imag_b(o^m)$ and trace threshold value, the tracing was terminated. Otherwise, the position of o^m was corrected using equation (7), and whether the tracing should be terminated was determinated by the following rulers:

a) The collision detection procedure was used to stop the tracing.
b) If the distance of the position o^m and o^{m-1} was less than 0.15 μm, the tracing was stopped.
c) The prior direction of position o^k, setting $k = \max(m-6, 2)$, denoted by r_p, yields an inner product with the direction of $\overrightarrow{o_k o_{m-1}}$, $\overrightarrow{o_k o_{m-2}}$, $\overrightarrow{o_{k-1} o_{m-1}}$, $\overrightarrow{o_{k-1} o_{m-2}}$ respectively. If the maximum of these inner product values was more than 0.5, the tracing was stopped.
d) If the signal values of o^{m-1} and o^{m-2} were both less than the sum of their corresponding background and two times the atandard deviation of their corresponding background, we detect whether the position o^m was the terminated position. If it was a terminate position, the tracing was stopped.

We describe the main procedures for collision detection in a). Before the beginning of the tracing, based on our previous work that showed that the shape of cell body can be precisely reconstructed (ref. 6), the interior region of each cell body can be identified by assigning a value that can distinguish other cell bodies and neurites. After terminating each tracing, we use the sequence number of the corresponding seed to label the area of this traced neurite. For each traced position, we select its neighborhood that contain 125 volume pixels and assign the sequence number to the volume pixels in the selected neighborhood. For the current position of o^m, we count the number of labeled volume pixels in its position and its cubic neighborhood with size of five pixel volumes. If the counting number was less than 75 volume pixels, the tracing continued. Otherwise, starting from the o^m, the tracing procedure was carried out ten times and the new positions were generated. The distribution of these positions could be divided into three cases: 1) located in the labeled volume pixels; 2) the position located the background and the tracing was not finished; 3) the position located in the unlabeled volume pixels with the signal values. If case 1) or 2) occurred, we stopped the tracing and built a connection between the traced neurites and the labeled neurite. In case 3), the tracing continued.

Because of the constraint in the tracing direction using a constrained principal direction, the tracing activity sometimes stopped when the direction of the neurite changed sharply. To avoid this case and keep the whole tracing as complete as possible, we introduce the terminal condition d). In d), if the signal intensity of the position o^{m-2} and o^{m-1} was low, which may be caused by the sharp direction of neurite, we carried out the following procedure: starting from o^{m-2}, the tracing activity was carried out five times using the Rayburst method (ref. 8); if we obtained this complete tracing, the current traced position was the starting point, and the tracing continued using our method. Otherwise, the position o^{m} was detected as the terminal position. Generally, when using the Rayburst sampling method for tracing, two types of rays that began from the current position, i.e., forward and backward, could be divided. The average positions of volume pixels passed through by the forward rays is the next position of the current position. Repeating this operation, we can obtain a series of orderly positions consisting of the centerline of neurite. Essentially, Rayburst sampling method adopts regional increases of the tracing and thus effectively addresses a sharp change in the direction of neurite. However, this method cannot overcome the problem that some positions of neurite are polluted by noise. In our analysis, the Rayburst sampling method was only used in a few cases.

2.6 Thick neurite tracing and other tracing methods

In our processing, tracing of thick neurites occurred prior to other neurites. Thick neurites have more connections with other neurites than thin ones; therefore, precise tracing of the thick neurites may be more important than that of other neurites. Generally, more precision in the initial tracing can be achieved due to fewer collisions. Therefore, we trace the thick neurites in the initial stage. In the tracing of thick neurites, the initial position in the detected region should be confirmed. First, we use the region increasing method to detect the region, which is the outward soma and connected to the soma surface. The soma surface could be reconstructed by our previous method (ref. 6). Second, from the detected region, we select the volume pixels. The distance of the positions of the selected volume pixels and the soma surface was greater than two times of the size of volume pixel. These selected volume pixels form the multiple connected regions due to the lose of the connected bridge and the soma surface. Finally, from the multiple connected regions, we select the region with the maximum number of volume pixels and compute the center position of the selected region, which is regarded as the starting position. After finding the starting position, the tracing of the thick neurites was carried out using the Rayburst sampling method (twelve traces). Then, we regarded

the traced current position as the initial position, computed the prior direction according to the traced positions, and continued tracing using the procedure described in the initial part of **Section 2**. Compared to the constrained principal curve method, Raybust sampling has no procedure for local signal extraction and is more robust to deal with sharp changes in the size of neurite near the soma. For the tracing of other neurites, we used the tracing procedure described in the initial part of **Section 2**.

Supplementary Note 2

1. The core of the reconstruction of the neuronal population is a special graph partitioning problem

The reconstruction of the neuronal population generally involves two steps: (a) localizing the soma, tracing the neurites and building the link between the two neurites or the link between neurite and soma; and (b) assigning the traced neurites to different individual neurons. By using the L_1 minimization model (ref. 5) and the constrained principal curve method (**Supplementary Note 1**), the positions of somas, the skeletons of neurites, and the corresponding links could be obtained. Here, the link refers to one end of a neurite contacting another neurite, or contacting a soma. The soma position and neurite skeleton could be regarded as the nodes of a graph, and the corresponding links could be used to build the edges of a graph. Based on this, the graph could be reconstructed, as follows.

Let S_1, S_2, …, S_m and N_1, N_2, …, N_n represent the positioned somas and the traced neurites, respectively. Correspondingly, let N_s_1, N_s_2, …, N_s_m and N_n_1, N_n_2, …, N_n_n represent the soma nodes and neurite nodes in the graph. If one end of neurite N_i contacts soma S_k, the edge of the graph with ends N_n_i and N_s_k would be built and denoted by (N_n_i, N_s_k); if one end of neurite N_i contacts with neurite N_j or one end of neurite N_j contacts neurite N_i, the edge of the graph with ends N_n_i and N_n_j would be built and denoted by (N_n_i, N_n_j). In this way, Graph $G = (V(G), E(G))$ can be reconstructed. $V(G)$ is a set of nodes and $E(G)$ is a set of edges.

Graph G generally contains many connected graphs, and these connected graphs can be classified into three types: (1) containing no soma node, (2) containing only one soma node, and (3) containing more than one soma node (**Supplementary Fig. 21**). For Type (1) connected graph, neurite nodes had no connection with the soma node. Correspondingly, the traced neurites cannot be assigned to individual neurons. For Type (2) connected graph, all the traced neurites

corresponding to neurite nodes in the graph were assigned to a single neuron corresponding to the soma node. For Type (3) connected graph, any one of the neurite nodes were directly or indirectly link with two or more soma nodes, and this indicates that one neurite can be assigned to two or more neurons, in contradiction to the basic fact that one neurite only belongs to one neuron. Therefore, the core of the assignment of a neurite in the reconstruction of a neuronal population ris a special graph partitioning problem in that a Type (3) connected graph is required to be partitioned into a Type (2) connected graphs.

2. The principle underlying the methodof neuronal population

Property 1 For Type (3) connected graph, there exists at least one neurite node whose corresponding neurite has two links, i.e., both two ends of the neurite contact with another neurite, or one end contacts the soma and the other contacts another neurite.

Proof: According to the definition of Type (3) connected graph, any two soma nodes (N_s_i and N_s_j) are connected by a path, denoted by N_s_i–N_n_{i1}–N_n_{i2}–…–N_n_{ik}–N_s_j. Suppose that all neurites, corresponding to neurites nodes in Type (3) connected graph, have one link. Based on the building of the edge, for edge (N_s_i, N_n_{i1}), N_s_i is the soma node, and neurite N_{i1} thus has one link; that is, its end contacts soma S_i. For edge (N_n_{i1}, N_n_{i2}), there exists two cases, the end of neurite N_{i1} contacts neurite N_{i2}, or the end of neurite N_{i2} contacts neurite N_{i1}. If the former is true, neurite N_{i1} has two links, namely, one end contacts the soma S_i and the other contacts neurite N_{i2}, contradicting the supposition. Therefore, the latter is true, and neurite N_{i2} has a link. For edge (N_n_{i2}, N_n_{i3}), the analysis is similar to the edge (N_n_{i1}, N_n_{i2}) indicating that N_{i3} has a link (its end contacts N_{i2}). The above analysis was used for edge (N_n_{i3}, N_n_{i4}),…, (N_n_{ik-1}, N_n_{ik}), and was concluded that N_{i4},…, N_{ik-1}, N_{ik} have one link. For edge (N_n_{ik}, N_s_j), N_s_j is the soma node, and therefore, one end of neurite N_{ik} contacts the soma. Neurite N_{ik} has two links, namely, one end contacts soma S_j and the other contacts neurite N_{ik-1}, contradicting the supposition.

Property 2 If all neurites have one link, there are no Type (3) connected graphs in the reconstructed graph $G = (V(G), E(G))$.

Proof: If $G = (V(G), E(G))$ includes Type (3) connected graph, according to **property 1**, there exists at least one neurite node whose corresponding neurite has two links, contradicting the conclusion that all neurites have one link.

Property 3 Type (2) connected graph, reconstructed with only the neurites that have one link (only one end of a neurite contacts another neurite or soma) are tree graphs.

Proof: Suppose that a Type (2) connected graph is not a tree graph. A cycle N_n_{i1}–N_n_{i2}–…, –N_n_{ik}–N_n_{i1} exists in the connected graph. There are two cases: 1) the cycle contains the soma node; 2) the cycle doesn't contain the some node. For Case 1), the conclusion can be easily derived by using the analysis in proof of **property 1**. In the following, we aim at Case 2).

Let N_n_{im} be the node in the cycle that is nearest to the soma node, namely, the length of the shortest path between N_n_{im} and the soma node is less than that between another node in the cycle and the soma node. Let N_{s}–N_n_{it1}–N_n_{it2},…,–N_n_{itk}–N_n_{im} be the shortest path. If no one end of neurite N_{im} contacts neurite N_{itk}, there exists at least one neurite node in the shortest path whose corresponding neurite has two links, contradicting the conclusion that all neurites have one link. This can be proofed by using an analysis similar to that in the proof of **Property 1**. Therefore, neurite N_{im} has a link; that is, its end contacts neurite N_{itk}.

If N_n_{im} does not equal N_n_{i1}, the cycle decomposes into two paths. One is N_n_{im}–N_n_{im-1},…, N_n_{i2}, N_n_{i1}, and the other is N_n_{im}, N_n_{im+1}, …, N_n_{ik}, N_n_{i1}. The conclusion that the end of neurite N_{i1} contacts N_{i2} can be drawn. If not, a node exists in the former path whose corresponding neurite has two links, contradicting the conclusion that all neurites have one link, according to the analysis in the proof of **property 1**, and that N_n_{itk} is not in the cycle (If N_n_{itk} is in the cycle, it contradicts that N_n_{im} is the node that is nearest to the soma node). Similarly, the end of neurite N_{i1} contacts neurite N_{ik}. This indicates that neurite N_{i1} has two links, contradicting the statement that all neurites have one link.

If $N_n_{im} = N_n_{i1}$, the cycle can be divided into the paths N_n_{i1}–N_n_{i2} and N_n_{i1}– N_n_{ik}– N_n_{ik-1}–, …, – N_n_{i2}. For these two paths, the above analysis was used, and we found that neurite N_{i2} has two links, contradicting the statement that all neurites have one link.

Remark 1. According to **Property 1–2**, we conclude the candidate spurious links constitute the link between one end of a neurite with two links and another neurite. Here, the spurious links refer to the links that lead to any two soma nodes connected by a path. By deleting those spurious links, Type (3) connected graph can be decomposed into individual Type (2) connected graphs (**Property 2**). This helps identify the candidates for those spurious links. This also can explain why

all of the spurious link patterns in left-hand panel of **Fig. 1c** involve a neurite that has two links, and why a '–bridge–' is defined as a neurite whose two ends both have links (center panel of **Fig. 1c**).

Remark 2. The condition that all traced neurites have one link is easily satisfied in the manual reconstruction of a neuron, and this reconstruction therefore has a tree topology according to **Property 3**. Using this topology, the starting and termination ends of a neurite in the reconstructed neuron can be uniquely determined. This indicates that the statistical information from the starting and termination ends cannot be confused.

Remark 3. By deleting the spurious links that lead to any two soma nodes connected by a path, the graph partitioning problem for neuronal population reconstruction can be solved (**Property 2**). This illustrate the feasibility of the procedure in left-hand and center panels of **Fig. 1c**, which aimsing to identify neurites for which both ends have links and then deletes one of the two links.

Supplementary Discussions

1. The features of NeuroGPS-Tree

NeuroGPS-Tree utilizes a strategy quite different from those of previous tools (refs. 7–14). As mentioned, previous methods mainly use small-scale information (i.e., local signals surrounding the identification site; Scale 1 in **Fig. 1b**) to identify links. In addition to using these small-scale signals, NeuroGPS-Tree also uses larger-scale information, such as the related neuronal process (Scale 2**)**, the identified neuronal tree (Scale 3), and the whole reconstructed network (Scale 4 in **Fig. 1b**). Furthermore, this multi-scale information is combined with basic statistical information extracted from neuronal morphology (**Fig. 1e**) to progressively identify spurious links. This strategy partially mimics the manual identification process utilized by an experienced technician. In fact, when an experienced technician identifies spurious links, signals at different scales are extracted, integrated and progressively used. Intuitively, some reconstruction processes performed by humans should be integrated into the framework of automation as effectively as possible for better reconstruction (refs. 15 and 16). NeuroGPS-Tree may demonstrate the advantages of operating similarly to the brain.

Another feature of NeuroGPS-Tree is that extracted statistical information is repeatedly used. This feature is based on two considerations. First, direction can be confirmed for only a portion of the neuronal processes in the reconstructed network; thus, it is impossible to identify all spurious links

using a single procedure, as shown in **Fig. 1c**. Second, identifying spurious links in batches and from different directions using a few relatively simple operations may be more feasible and effective than attempts to detect all of the spurious links simultaneously using a complex procedure, especially for cases when the diversity of spurious links makes designing a complex and effective identification procedure extremely difficult.

To mimic the manual strategy, we also need to use the same morphological information used by humans in identification. We extracted this information by performing a statistical analysis over the same molecular type of neuron but with well-resolved morphology. This statistical information, including the size and signal intensity of the neurite ends and the angle between two directly connected neurites (**Fig. 1e**), is general and simple and does not involve specific or complicated patterns. Thus, the conclusions derived from this statistical information should be applicable to diverse types of neurons. Accordingly, NeuroGPS-Tree may be generalized to the reconstruction of a neuronal population containing a diverse array of neuronal types. It is worth noting that if the statistical information from other neuronal populations is different from that presented in **Fig. 1d, e**, directly applying NeuroGPS-Tree may achieve a low-accuracy reconstruction. To address this issue, other sets of statistical information useful for identifying spurious links should be extracted to replace the data used here. In this manner, the strategy of NeuroGPS-tree remains applicable.

2. The wide use range of NeuroGPS-Tree

The applicability of NeuroGPS-Tree to image stacks from other data sources and/or acquired using different imaging modalities is an important issue. In addition to fMOST data, NeuroGPS-Tree has been applied to DIADEM datasets (refs. 1–4), including a very dense one (the neuromuscular junction dataset), and was found to be effective. For the regular datasets (ref. 1), the average recall and precision were above 90%, and the DIADEM Score was approximately 0.75±0.10 (**Supplementary Fig. 13**). For the neuromuscular dataset (ref. 2), the proposed method could effectively trace the densely packed fibers (**Supplementary Fig. 14**).The reconstructed results from Neocortical Layer 1 Axons (ref. 3) and Hippocampal CA3 Interneuron datasets (ref. 4) show that NeuroGPS-Tree could distinguish between crossings and bifurcations (**Supplementary Fig. 15**), and could detect the weak signal (**Supplementary Fig. 16**). These tracing results reached a similar identification level as reported in previous studies (Refs. 7–14). Because the tested DIADEM datasets were acquired using confocal (Olfactory Projection Fibers (ref. 1) & neuromuscular junction datasets (ref. 2)), two-photon (neocortical layer datasets (ref. 3)) and

wide-field (Hippocampal CA3 Interneuron dataset (ref. 4)) microscopy, which represent all major optical microscopic imaging approaches, we propose that NeuroGPS-Tree can be applied widely. In addition, NeuroGPS-Tree can, in principle, reconstruct neuronal populations from multi-color labeling datasets, such as the Brainbow dataset (ref. 17). After obtaining a reconstruction for each channel using this method, a better reconstruction can be achieved by merging them as different channels may label different parts of the neuron.

Processing neuronal population data requires not only the tracing of neuronal processes but also assigning the traced neuronal processes to individual neurons. Obviously, tracing accuracy is one of the factors that determine the accuracy of the population reconstruction. Many tracing methods have been proposed, and each possesses an advantage for specific applications. For example, Vaa3D traces neuronal processes with diverse morphology (ref. 11); Open-Snake performs well in tracing close-lying axons (ref. 9); model-based tracing can distinguish weak signal from strong background (ref. 13); and principal curves (ref. 7) traces axons in the Brainbow dataset (ref. 17). NeuroGPS-Tree assigns neuronal processes to the corresponding neuron and reconstructs a neuronal population. As a trial, NeuroGPS-Tree is used to reconstruct neurons from traced results from Open-Snake, NeuronStudio, and Vaa3D (**Supplementary Fig. 17**). Although preliminary, this indicates the potential compatibility of NeuroGPS-Tree with other suitable tracing methods, rather than only its own, for neuronal population reconstruction.

3. The limitation of NeuroGPS-Tree

Compared with manual results, we obtained a reconstruction accuracy of approximately 80% (recall rate of 80% and precision rate of 80%). For manual operation, we obtained a faithful baseline representation for method evaluation, even for dense tracing. However, some neurites were missed during automated tracing. For example, our tracing method (**Supplementary Note 1**) failed to trace broken neuronal processes in which a signal was missing for approximately 6 μm (**Supplementary Fig. 18**), which reduced the recall rate of the population reconstruction. In addition, the precision rate of the reconstruction might also be decreased by the incorrect identification of bifurcations or crossings or by signal interference during the tracing process. Generally, this occurs in the regions near the soma or apical dendrite, as shown in **Fig. 2e**; i.e., the differences between Sholl measures from the automatic and manual reconstruction.

4. Analysis of evaluation indexes of the automated algorithm

For a reconstruction, the DIADEM scores (ref. 18) sometimes fall considerably below the recall and precision rates. These parameters quantify the comparison between manual and automated reconstructions. The recall and precision rates are used to measure the total difference in morphology between the automated and manual reconstructions. The DIADEM score mainly focuses on the topological structure of the automated reconstruction. As a result, partial reconstruction is acceptable by measurements such as precision rate and recall rate but is not acceptable by the DIADEM score. For example, when the ratio of the reconstructed fiber length over the manual traced value is less than a given threshold or when the termination point of the fiber identified by automatic reconstruction does not match that obtained manually, the reconstruction might be accepted by the former measure but not the latter (**Supplementary Fig. 11**). In addition, for short fibers, this standard in DIADEM score measurement is stricter and requires that the automated reconstruction is well matched with the manual reconstruction. The DIADEM score could be boosted by optimizing the automated and manual reconstruction skeletons based on neuronal morphology. The optimal skeleton will be achieved when the bifurcation nodes are precisely located and the points on the skeleton are located in the center-line of a neurite. It was also noted that the Sholl measurement (ref. 19) was first used here to compare an automated and a manual reconstruction and was then used as a parameter for the high-throughput calculation of morphological quantities after digitalization. A comparison with the same neuron is reasonable to some extent but is not acceptable across different neurons. In some cases, the results can be very misleading; for example, the same Sholl measurements may be derived from different neuronal trees.

Supplementary Software

System requirements:

The software was tested on Windows 7 × 64 and Windows 8 × 64 with a Microsoft Visual C++ 2008 Redistributable Package installed.

A video card that supports OpenGL is needed.

Installation:

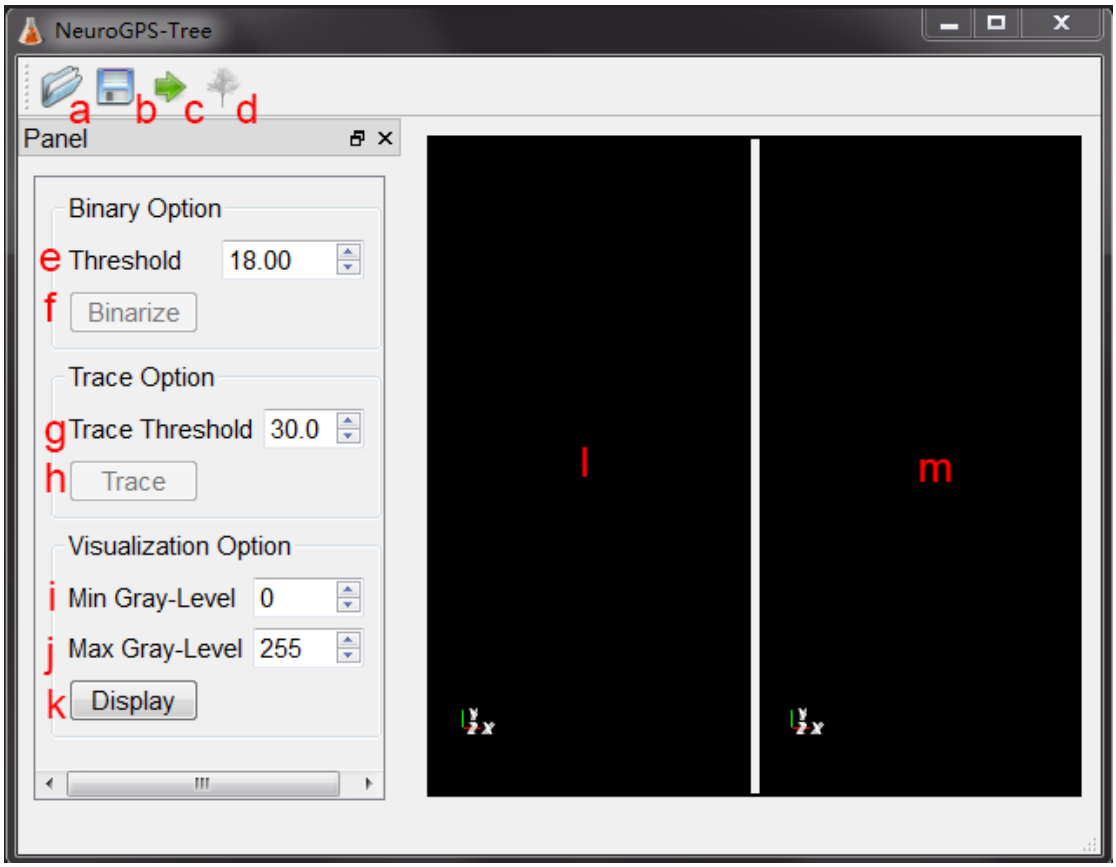
Copy NeuroGPS-Tree.zip to the hard drive.

Unpack NeuroGPS-Tree.zip (with WinZip, WinRAR, and free available software) into the

destination folder.

Start NeuroGPS-Tree.exe (Microsoft Visual C++ 2008 Redistributable Package must be installed on your computer)

User Interface:



a “Open Image” button; **b** “Save” button; **c** Batch reconstruction button; **d** “Open Neurite” button **e** Binary Threshold set; **f** “Binarize” button; **g** Trace Threshold button; **h** “Trace” button; **i** Min gray-level value of the colormap set for visualization; **j** Max gray-level value of the colormap set for visualization; **l** Left Visualization Window: render original image; **m** Right Visualization Window: render binary image.

Note 1: Please refer to the ***NeuroGPS-Tree User Guide*** for a detailed description of this software.

Note 2: NeuroGPS-Tree software requires the position of the somas to be known. Voxel size is set manually when opening image stacks. The NeuroGPS software has been applied in multi-core CPU to accelerate binarization and part of tracing procedures.

Note 3: As described in Supplementary **Note 2**, our tracing procedure involves some key parameters, including the binarization threshold, the size of the neighborhood of the current traced point, the size of the labeling area occupied by the traced neurite, collision detection parameters, and trace threshold etc. In NeuroGPS-Tree software, fixing these parameters’ settings can make it easy to operate but reduce its range of validity. Conversely, placing these parameters on the

software interface and allowing for variation of individual experiments will extend the validity range of the software but vastly increase the difficulty in using it. Considering the tradeoff between the validity range and the ease of using the software, two parameters, binarization threshold and tracing threshold, can be set on the software interface, and the other parameters are automatically modified based on voxel size. At present, our software can trace neurites with a radius less than 3 μm, which covers most of the neurons of the mice. Continuous tracing of the neurites requires that the voxel size be compatible with the thickness of the neurites. A voxel size that is too large fails to provide the neurite shape, while a voxel size that is too small results in a large data volume for large brain areas, making it difficult for computing. Here, we recommend a proper voxel size, ranging from $0.5 \times 0.5 \times 0.5$ μm^3 to $1 \times 1 \times 2$ μm^3. Datasets with more fine sampling with a voxel size smaller than $0.5 \times 0.5 \times 0.5$ μm^3 can also be processed using our software but will automatically be merged to a larger voxel size within this range. For datasets with a voxel size or neurite thickness beyond the above range, internally setting the tracing parameters may provide better reconstruction. For some DIADEM datasets (**Supplementary Figs. 13–16**), some tracing parameters inside the software are manually set and image preprocessing is also adopted.

Note 4: At present, NeuroGPS-Tree can provide the preliminary function for the visualization of the dataset and reconstructed results. For better visualization, third-party professional software is required.

References

1. Jefferis, G., et al. Cell **128** (6), 1187–203 (2007).
2. Lu, J., et al. PLoS Biol. **7** (2), e32 (2009).
3. Paola1, V., et al. Neuron **49** (6), 861–75 (2006).
4. Calixto, E., et al. J. Physiol. **586** (Pt 11), 2695–712 (2008).
5. Quan, T., et al. Sci. Rep. **3**, 1414 (2013).
6. Quan, T., et al. Sci. Rep. **4**, 4970 (2014).
7. Bas, E. & Erdogmus, D. Neuroinformatics **9** (2–3), 181–191 (2011).
8. Ming, X. et al. PloS one **8**, e84557 (2013).
9. Wang, Y., Narayanaswamy, A., Tsai, C.-L. & Roysam, B. Neuroinformatics **9** (2–3), 193–217 (2011).
10. Wearne, S. L., et al. Neuroscience **136**, 661–680 (2005).
11. Peng, H., Ruan, Z., Long, F., Simpson, J. H. & Myers, E. W. Nat. Biotechnol **28** (4), 348–353 (2010).

12. Turetken, E., Gonzalez, G., Blum, C. & Fua, P. Neuroinformatics **9** (2–3), 279–302 (2011).

13. Zhao, T., et al. Neuroinformatics **9** (2–3), 247–261 (2011).

14. Chothani, P., Mehta, V. & Stepanyants, A. Neuroinformatics **9** (2–3), 263–278 (2011).

15. Meijering, E. Cytometry Part A **77** (7), 693–704 (2010).

16. Donohue, D. E. & Ascoli, G. A. Brain Res. Rev. **67** (1), 94–102 (2011).

17. Livet, J., et al. Nature **450**, 56–62 (2007).

18. Gillette T. & Brown, K. & Ascoli, G. Neuroinformatics **9**, 233–245 (2011).

19. Sholl, D. J. Anat. **87** (Pt 4), 387–406 (1953).