BIOINFORMATICS

Vol. 26 ISMB 2010, pages i38–i46
doi:10.1093/bioinformatics/btq212

Automatic reconstruction of 3D neuron structures using a graph-augmented deformable model

Hanchuan Peng*, Zongcai Ruan, Deniz Atasoy and Scott Sternson

Janelia Farm Research Campus, Howard Hughes Medical Institute, Ashburn, VA 20147, USA

ABSTRACT

Motivation: Digital reconstruction of 3D neuron structures is an important step toward reverse engineering the wiring and functions of a brain. However, despite a number of existing studies, this task is still challenging, especially when a 3D microscopic image has low single-to-noise ratio and discontinued segments of neurite patterns.

Results: We developed a graph-augmented deformable model (GD) to reconstruct (trace) the 3D structure of a neuron when it has a broken structure and/or fuzzy boundary. We formulated a variational problem using the geodesic shortest path, which is defined as a combination of Euclidean distance, exponent of inverse intensity of pixels along the path and closeness to local centers of image intensity distribution. We solved it in two steps. We first used a shortest path graph algorithm to guarantee that we find the global optimal solution of this step. Then we optimized a discrete deformable curve model to achieve visually more satisfactory reconstructions. Within our framework, it is also easy to define an optional prior curve that reflects the domain knowledge of a user. We investigated the performance of our method using a number of challenging 3D neuronal image datasets of different model organisms including fruit fly, *Caenorhabditis elegans*, and mouse. In our experiments, the GD method outperformed several comparison methods in reconstruction accuracy, consistency, robustness and speed. We further used GD in two real applications, namely cataloging neurite morphology of fruit fly to build a 3D 'standard' digital neurite atlas, and estimating the synaptic bouton density along the axons for a mouse brain.

Availability: The software is provided as part of the V3D-Neuron 1.0 package freely available at http://penglab.janelia.org/proj/v3d

Contact: pengh@janelia.hhmi.org

1 INTRODUCTION

A major engineering challenge recognized by the National Academy of Engineering is to reverse engineer a brain (Perry *et al.*, 2008; Roysam *et al.*, 2009). To achieve such a goal, we would need a number of enabling techniques, of which one of the fundamental computational techniques is to precisely digitize the 3D morphological structure of a neuron acquired through various microscopy methods, such as laser scanning microscopy. This process is often called neuron reconstruction or tracing.

There are a number of existing studies on neuron tracing. One may use structure elements (spheres, cylinders) as image matching templates to progressively fit and march along a neuron structure (Al-Kofahi, *et al.*, 2002, 2003). Wearne *et al.* (2005) proposed a ray-bursting algorithm to cast rays in the bright image regions to approximate a neuron structure. Zhang *et al.* (2007) assembled multiple detected center skeletons of a neuron as the complete 3D structure. Interactive tracing was also considered. The widely used Neurolucida software (MBF Bioscience) provides a method to reconstruct a neuron manually in 2D. Meijering *et al.* (2004) used the live-wire algorithm in 2D semiautomatic online tracing. In our V3D system (http://penglab.janelia.org/proj/v3d), we also provide an efficient 3D semiautomatic, interactive neuron tracing and editing method (Peng *et al.*, 2010). Fibrous tissue segmentation methods for other medical images and bioimages (e.g. electron microscopy images) are also relevant to neuron tracing. For instance, partial differential equations and mathematical morphology methods for blood vessel tracing (e.g. Zana *et al.*, 2001; Zhang *et al.*, 2009) can be used for neurons as well.

In this article, we focus on a hard case of 3D neuron reconstruction, where a 3D microscopic image had low single-to-noise ratio, and/or broken and fuzzy neurite patterns that are due to the intrinsic punctuated neurite structures (e.g. synaptic boutons) or imperfectness of sample preparation. Such datasets are common for the nervous systems of different animals. For instance, the punctuated and thus often broken neurites can be ubiquitously seen in the single-neuron images of *Drosophila melanogaster* (fruit fly) and *Caenorhabditis elegans* (Fig. 1a and b), and the multiaxon staining of a mouse brain (Fig. 1c). The high level of background noise in an image can also lead to broken and fuzzy neurites (e.g. Fig. 2A). In these challenging situations, the methods summarized above often fail. Our intuition to solve this problem is to combine both global and local cues. The global information will guide the finer-scale optimization using local information. We formulate a graph-augmented deformable (GD) model based on the geodesic shortest path, and produce satisfactory tracing by first optimizing a shortest path graph problem followed by refining a 3D deformable curve. Within the GD framework, it is also easy to incorporate an investigator's prior knowledge, e.g. the starting and ending locations for tracing, the shape prior of a 3D reconstruction, etc.

In a number of experiments using both synthetic data and real neuronal patterns of different model organisms, including fruit fly, *C.elegans* and mouse, our method outperforms several comparison methods in terms of accuracy, consistency, robustness and speed. We also applied GD to building a 3D neurite pattern atlas for the fruit fly brain and to estimating synaptic bouton density in a mouse brain.

2 METHODS

A basic module in most neuron-tracing algorithms is to trace an individual neurite tract/segment. We consider a similar strategy. Thus, in the following descriptions we focus on how to trace such a tract. We assemble multiple tracts by detecting and merging their shared portions (Peng *et al.*, 2010).

*To whom correspondence should be addressed.

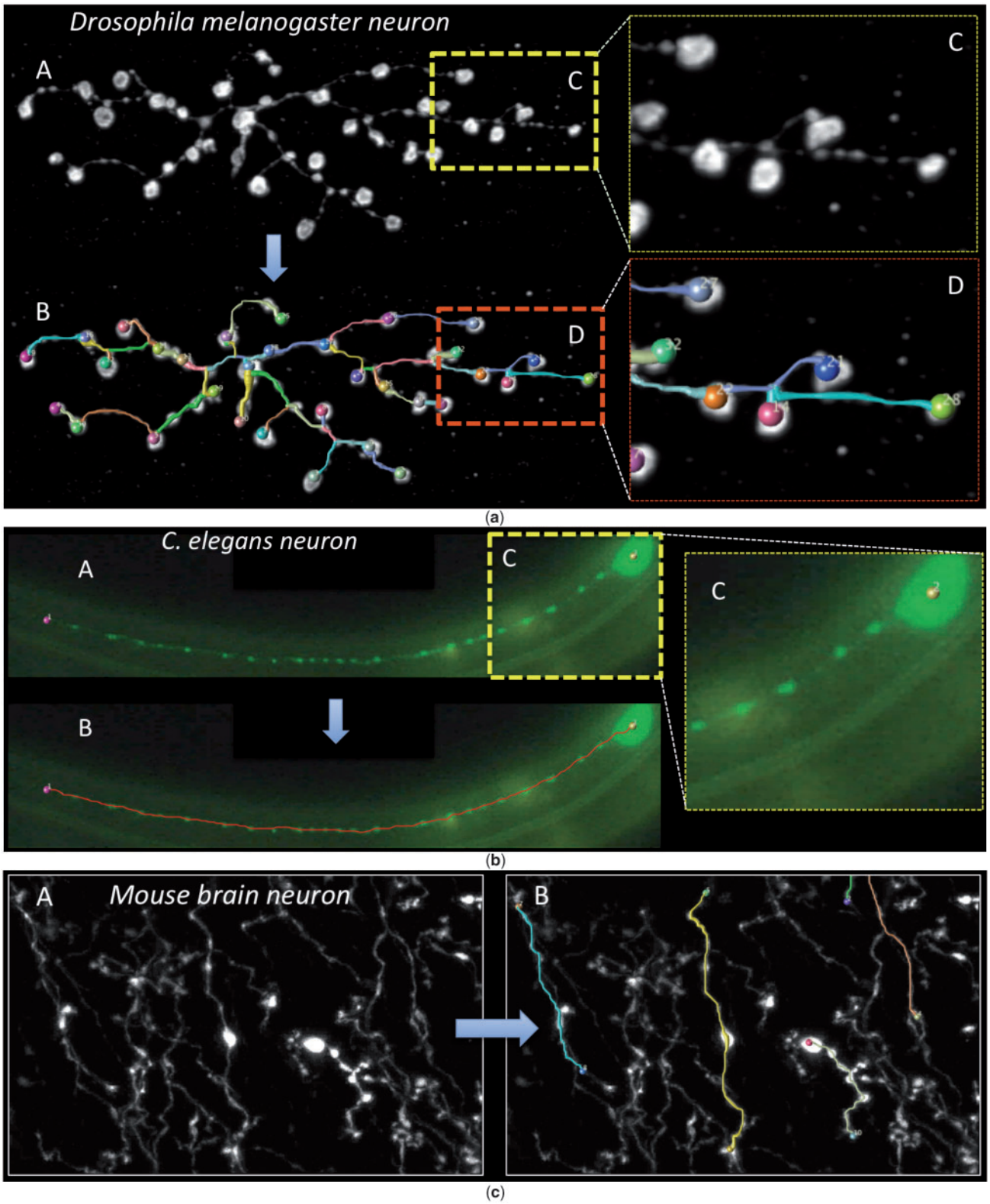


Fig. 1. Examples of highly punctuated neurites, that have broken and often fuzzy structures and low single-to-noise ratio and their 3D reconstructions. (**a**) A single lamina neuron of fruit fly (G. Rubin laboratory) along with the 3D reconstruction. In the input image (A and C) the fruit fly neuron has broken and punctuated neurites. We applied the GD method to trace the neuron morphology (B and D) segment by segment (color-coded neurite models). (**b**) Punctuated axon (MEC::YFP) of *C.elegans* (A) (Chelur *et al.*, 2002), along with the GD tracing (B). (**c**) Multiaxon staining (A) of a mouse brain region (S. Sternson laboratory), which displays punctuated structures including synaptic boutons, along with GD reconstructions of several user-specified neurites (B).

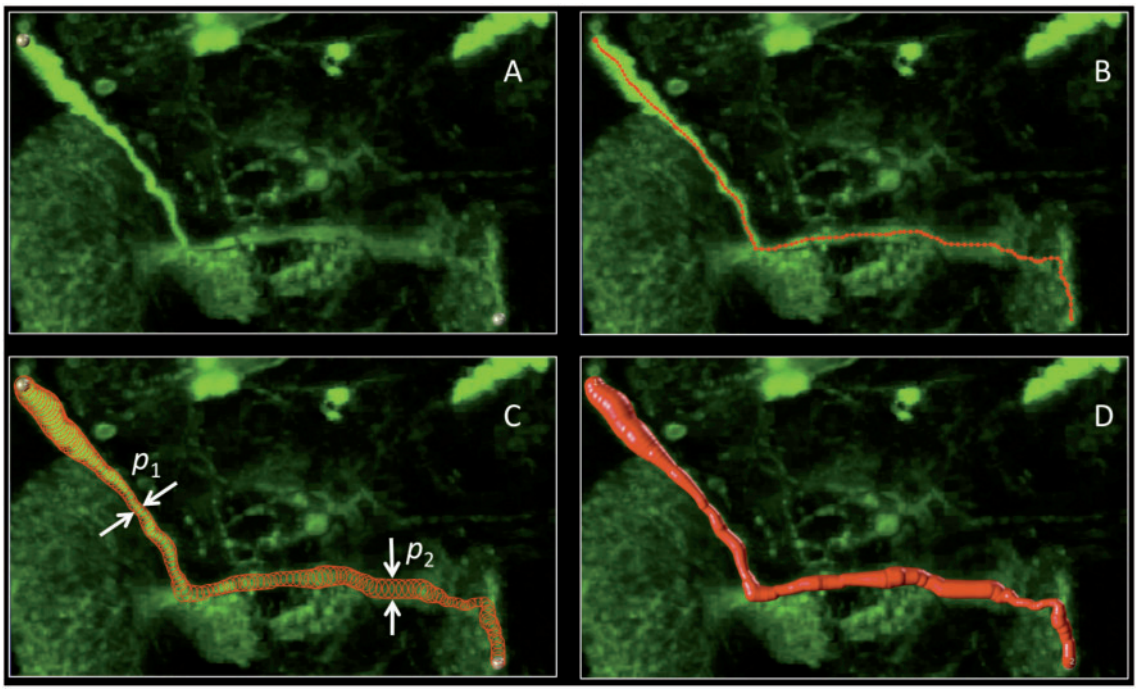


Fig. 2. (**A**) Reconstruction of a neurite tract in a 3D confocal image of fruit fly brain. The skeleton (**B**) and the estimated width (**C**) are assembled as a digital model (**D**). In our formulation, this reconstruction has a smooth skeleton and represents the maximum neurite tract information with the shortest length. Since the neuronal pattern in (A) is broad, the haze regions make it hard to trace the skeleton in (B and D). The GD method overcomes this problem by specifying a pair of endnodes [two spheres in (A)], or using other ways to incorporate a user's prior knowledge, as global cues to guide tracing.

2.1 Detect the skeleton of a broken and fuzzy neurite using GD

A neurite tract (Fig. 2A and D) can be modeled using a curved skeleton (Fig. 2B), or 'backbone', plus the estimated width along the skeleton (Fig. 2C). Similar examples can be seen in subfigure D of Figure 1a. If in the image domain such a neurite tract was sufficiently continuous and clear (i.e. has high contrast to the image background), many methods, e.g. a tube- or sphere-fitting model (e.g. Al-Kofahi *et al.*, 2002, 2003), anisotropic diffusion (Perona and Malik, 1990), would produce a reasonably good reconstruction. Unfortunately, in the case of broken and fuzzy neurites as in Figure 1, and the noisy image as in Figure 2A, it is hard to directly employ these approaches. Alternatively, here we intuitively define the 'optimal' skeleton as a smooth curve through as many local centers of high intensity image regions as possible, and at the same time to minimize the length of this curve as much as possible, similar to the reconstruction in Figures 1b and 2B. We also require that a reconstruction should also reflect a user's domain/prior knowledge (e.g. the bushy area in Fig. 2A is not part of the tract). We formulate a constrained variational problem to determine this skeleton.

Denote the skeleton curve as C, we minimize the following geodesic length $L(C)$ along it,

$$L(C)=\int_C g(p,I[\Theta(p,r)])\|dp\| \tag{1}$$

where $g(.)$ is a geodesic metric function (Section 2.2), $p=(x,y,z)$ represents a 3D location on the curve, $I[\Theta(p, r)]$ is the distribution of image voxel intensity in the local region $\Theta(p, r)$ around the position p with radius r, $||dp||$ is the differential Euclidean length along the curve, i.e. $\|dp\|=\sqrt{(dx)^2+(dy)^2+(dz)^2}$.

We solve this variational problem using a discrete deformable model, which converts it to a multivariate optimization problem. We use a cubic-spline curve with K control point $\{C_k,\ k=1,2,\ldots,K\}$ to represent the continuous curve C. Since each control point corresponds to a 3D location, there are totally 3K variables to be optimized for a 3D image. Evidently, it is a non-linear optimization because of the non-linear distribution of image intensity.

It is well known that non-linear multivariate optimization is very sensitive to the initialization, and cannot guarantee to converge to the global optima in finite time using existing local search techniques. However, we note that for a graph with finite nodes, there exist shortest path algorithms to find the global solution for the entire graph. This motivates us to use a graph algorithm to do global coarse searching to find a good solution, which is then used as the input of a finer-scale local search using the deformable curve. We call this approach GD model, which has two steps:

- **Graph-step** (Section 2.3): create an undirected graph G on the image (graph vertexes are image voxels) with edge weight calculated according to the geodesic metric function $g(.)$, and then find the shortest path P with respect to a pair of 'defined' endnodes a and b of the skeleton curve. (See Section 2.6 for ways to 'define' the endnodes a and b.)
- **Deforming-step** (Section 2.4): take the result of the first step, P, as the initial control point estimation of the deformable model $\{C_k,\ k=1,2,\ldots,K\}$, and then use local optimization techniques to refine and smooth the skeleton curve C.

2.2 Geodesic metric function

The geodesic metric function $g(.)$ is the key to the GD algorithm. It contains three factors to define the skeleton curve.

$$g(p,I[\Theta(p,r)])=g_E g_I g_C \tag{2}$$

where p, $\Theta(p,r)$, $I[\Theta(p,r)]$ are the same in Equation (1), g_E, g_I and g_C are three metrics defined for the Euclidean length, image voxel intensity and the closeness to the local centers of image intensity distribution, respectively.

2.2.1 Metric of Euclidean length Since the geodesic length of a curve should reduce to the ordinary curve length when other constraints were not considered, we define the Euclidean length metric:

$$g_E = 1 \tag{3}$$

When g is set as g_E, we get the curve length $\int_C \sqrt{dx^2+dy^2+dz^2}$.

2.2.2 Metric of image voxel intensity Let us assume bright (but not dark) image voxels represent the neurite signal. This metric constrains that the skeleton curve should pass through image voxels with high intensity. We define

$$g_I = \exp\left(\lambda_I \left(1 - I(p)/I_{\max}\right)^2\right) \tag{4}$$

where $I(p)$ is the intensity value at location p and $I_{\max}$ is the maximum intensity of the entire image I, respectively. $g_I(.)$ can also be made adaptive to the local image content by replacing I_{max} using the maximum intensity of a local area, i.e. $I_{\max}[\Theta(p,r)]$. We use the exponent of squared inverse intensity to emphasize the voxel intensity that represents signal. λ_I is a positive coefficient to control the contribution of this term. We choose $\lambda_I = 10$ (other big values like 20 lead to similar results).

2.2.3 Metric of skeleton's closeness to local centers of image intensity distribution In order to make the skeleton better represent the image intensity distribution, intuitively we would like to constrain the skeleton curve to pass through the local centers of mass of an image. Thus, we define

$$g_C(p, I[\Theta(p,r)]) = \exp\left(\lambda_C \frac{\iiint_{\Theta(p,r)} \|p-q\|^2 I(q)dq}{\iiint_{\Theta(p,r)} I(q)dq}\right) \tag{5}$$

where we aggregate the voxel intensity weighted distances between p and p's neighboring voxels [within $\Theta(p, r)$], normalized by the total mass of this neighboring area. λ_C is a positive coefficient to control the contribution of this term (we use $\lambda_C = 1$). Of note, g_C can also be understood as the external 'image' energy in a deformable curve model.

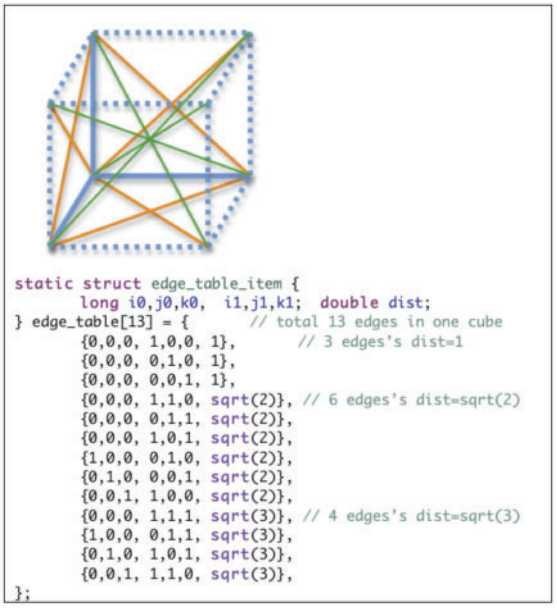
2.3 Graph-step: shortest path in graph

We create a graph $G=(V, E)$ for image voxels, where V is the set of vertexes and E the set of edges among vertexes. In V, each vertex stands for an image voxel. A pair of vertexes, which correspond to image voxels at locations $v_0 = (x_0, y_0, z_0)$ and $v_1 = (x_1, y_1, z_1)$, have an undirected edge between them only when the two vertexes correspond to immediate spatial neighbors, i.e. their spatial coordinates simultaneously satisfy $|x_0 - x_1| \le 1$, $|y_0 - y_1| \le 1$ and $|z_0 - z_1| \le 1$ (and of course at two different locations). The edge weight is determined using our geodesic metric function. In the graph step, since we have not yet produced an estimation of the optimal neighborhood $\Theta(p,r)$ of each skeleton point p, we set $r = 0$. Thus from the following Equations (2)–(5), we define the edge weight between vertexes v_0 and v_1 as

$$e(v_0, v_1) = \|v_0 - v_1\| \cdot \left(\frac{g_I(v_0) + g_I(v_1)}{2}\right) \tag{6}$$

In our implementation, we use an edge lookup table in Figure 3 to speed up the creating of the graph. This lookup table, shown in C language, is defined as a cube of eight neighboring nodes. With it we need to do only one pass of $g_I(.)$ computation and directly set up the graph.

Obviously, our geodesic metric function outputs only positive value. Thus, we use the Dijkstra algorithm (Dijkstra, 1959) to find the shortest path in G. Its time complexity is $O(|E|+|V|\log(|V|))$ using a Fibonacci heap. For a 3D image with totally N voxels, $|E| = 13|V|$ and $|V| = \kappa N$, where $\kappa (0 < \kappa \le 1)$ is a subsampling factor controlling if only a portion of image voxels are used in the computation. Thus, the time complexity of the graph-step is $O(N\log N)$. It can also be significantly accelerated by choosing a small κ via downsampling a large image, or using only image voxels that are bright enough (e.g. voxel intensity larger than the mean intensity of the entire image). Removing the diagonal edges in Figure 3, i.e. those with sqrt(2) (orange colored edges) and sqrt(3) (green colored edges), can also speed up the computation significantly, at the cost of a less optimal initialization for the subsequent deforming-step.



```
static struct edge_table_item {
       long i0,j0,k0,  i1,j1,k1;  double dist;
} edge_table[13] = {       // total 13 edges in one cube
       {0,0,0, 1,0,0, 1},       // 3 edges's dist=1
       {0,0,0, 0,1,0, 1},
       {0,0,0, 0,0,1, 1},
       {0,0,0, 1,1,0, sqrt(2)}, // 6 edges's dist=sqrt(2)
       {0,0,0, 0,1,1, sqrt(2)},
       {0,0,0, 1,0,1, sqrt(2)},
       {1,0,0, 0,1,0, sqrt(2)},
       {0,1,0, 0,0,1, sqrt(2)},
       {0,0,1, 1,0,0, sqrt(2)},
       {0,0,0, 1,1,1, sqrt(3)}, // 4 edges's dist=sqrt(3)
       {1,0,0, 0,1,1, sqrt(3)},
       {0,1,0, 1,0,1, sqrt(3)},
       {0,0,1, 1,1,0, sqrt(3)},
};
```

Fig. 3. The edge lookup table for creating the shortest path finding graph.

2.4 Deforming-step: energy minimization for local optimization

We use the shortest path P obtained in the graph-step to initialize the control point locations $\{C_k,\ k = 1, \ldots, K\}$ of the deformable curve C, which is further refined and smoothed using a local search. Using the geodesic metric definitions in Equations (1)–(5), we define three energy terms $E_E(k)$, $E_I(k)$ and $E_C(k)$,

$$E_E(k) = \|C_k - C_{k+1}\| (k = 1, \ldots, K-1) \tag{7}$$

$$E_I(k) = \lambda_I \left(1 - \frac{I(C_k)}{\max I[\Theta(C_k, r)]}\right)^2 \tag{8}$$

$$E_C(k) = \frac{\lambda_C \sum_{q\in\Theta(C_k,r)} \|C_k - q\|^2 I(q)}{\sum_{q\in\Theta(C_k,r)} I(q)} \tag{9}$$

The overall energy function for the discrete deformable curve has the following form

$$E_{GD} = \sum_{k=1}^{K-1} \exp(E_I(k) + E_C(k)) \|C_k - C_{k+1}\| \tag{10}$$

We note that the deforming-step is a local search around the optimal skeleton, thus we can reasonably assume $E_I(k)+E_C(k)$ is close to zero. Thus, we further consider the Taylor expansion of this term at 0 and get

$$\exp(E_I(k) + E_C(k)) \approx 1 + E_I(k) + E_C(k) \tag{11}$$

To achieve further convenience in the local search, we minimize an upper bound of Equation (10), where we replace the exponential term using Equation (11) and also substitute the total curve length (which is larger than 1) with the sum of squared piece-wise distances of control points.

$$E_{GD} = \sum_{k=1}^{K-1} \|C_k - C_{k+1}\|^2 + \max_k \|C_k - C_{k+1}\| \sum_{k=1}^{K} (E_I(k) + E_C(k)) \tag{12}$$

In this manner, we indeed can rewrite the energy function as Equation (13), which is similar to the backbone detection without boundary (BDB) algorithm we developed for detecting the skeleton of a curved *C.elegans* body (Peng *et al.*, 2008),

$$E_{GD} = \alpha E_{\text{image}} + \beta E_{\text{length}} + \gamma E_{\text{smoothness}} \tag{13}$$

$$E_{\text{image}} = \sum_{k=1}^{K} (E_I(k) + E_C(k)) \tag{14}$$

$$E_{\text{length}} = \sum_{k=1}^{K-1} \|C_k - C_{k+1}\|^2 \tag{15}$$

$$E_{\text{smoothness}} = \sum_{k=2}^{K-1} \left\| C_k - \frac{C_{k-1} + C_{k+1}}{2} \right\|^2 \tag{16}$$

where α, β, γ are three positive weighting factor (e.g. $\alpha = 1$, $\beta = \gamma = 0.2$), $E_{\text{smoothness}}$ is an additional constraint term to make the control points to be evenly spaced on the curve. Equation (13) can be minimized via iterative updating of control points based on the gradient descent (Peng *et al.*, 2008). Of note, after the graph-step, $I(C_k)$ in Equation (8) almost reaches its local maximum, in another word $E_I(k) \approx 0$. Thus in the deforming-step we can ignore it to make the computation even faster.

2.5 Dynamically estimate local region radius

While we optimize the skeleton curve of a neurite tract (Fig. 2B), we also dynamically estimate the width around each of its control points (Fig. 2C). We use a simple method. We define a radius-adjustable spherical region centered at a control point, and therefore adjust the radius from small to large until 0.1% of the image voxels within this sphere are darker than the average voxel intensity of the entire image. In most cases, the choice of 0.1% makes estimated boundary of a neurite have clear contrast to the image background. Optionally, we also allow a mean-shift type adjustment of the 3D location of a control point, based on the optimal radius detected. In such a case, we discard the tangent-direction movement of a control point, and thus constrain the movement of a control point only in the orthogonal plane. In this way, both the width of the neurite tract and the location of control points can be well detected.

2.6 Consider prior knowledge and global cues

At least two types of prior knowledge, i.e. the prior locations of the two endnodes and the prior skeleton curve, can be conveniently incorporated in the GD model. They serve as the global cues to guide the tracing. In our framework, the locations of two endnodes of a neurite tract specify where the shortest path starts and ends. This is similar to the 'seeding' step in earlier methods (e.g. Wearne *et al.*, 2005) so that the tracing program knows where to start the search. In addition, given a prior curve C_{prior}, we extend the energy function of the deformable model as

$$E = E_{GD} + \eta E_{\text{prior}} \tag{17}$$

$$E_{\text{prior}} = \sum_{k=1}^{K} \|d(C_k, C_{\text{prior}})\|^2 \tag{18}$$

where $d(.)$ is the shortest Euclidean distance between the control point C_k and the prior curve C_{prior}, and η is a positive coefficient. Equation (17) can be easily minimized via gradient descent.

We have recently developed a real-time 3D visualization-assisted image analysis system, called V3D, for very large bioimages (Peng *et al.*, 2010). One advantage of this system is to allow highly ergonomic interaction of 3D image *content* using an ordinary computer mouse, but not much more expensive stereo view or even virtual reality hardware. With V3D it is easy to supply the prior knowledge. A user can directly pinpoint any 3D image location in the 3D space, with only one or two mouse clicks, and thus specify the locations of endnodes of a neurite tract. The user can also draw directly in the 3D space using the mouse, with just one stroke, to produce a 3D prior curve that follows a neurite tract.

The prior information can also be produced automatically. For instance, V3D provides an Auto-Marker function to seed the GD tracing. We can also use the graph-step result as the prior curve, which will balance with the energy of the deformable model.

3 EXPERIMENTS AND DISCUSSIONS

We evaluated GD using both synthetic and real images, and compared it to other tracing tools including manual tracing

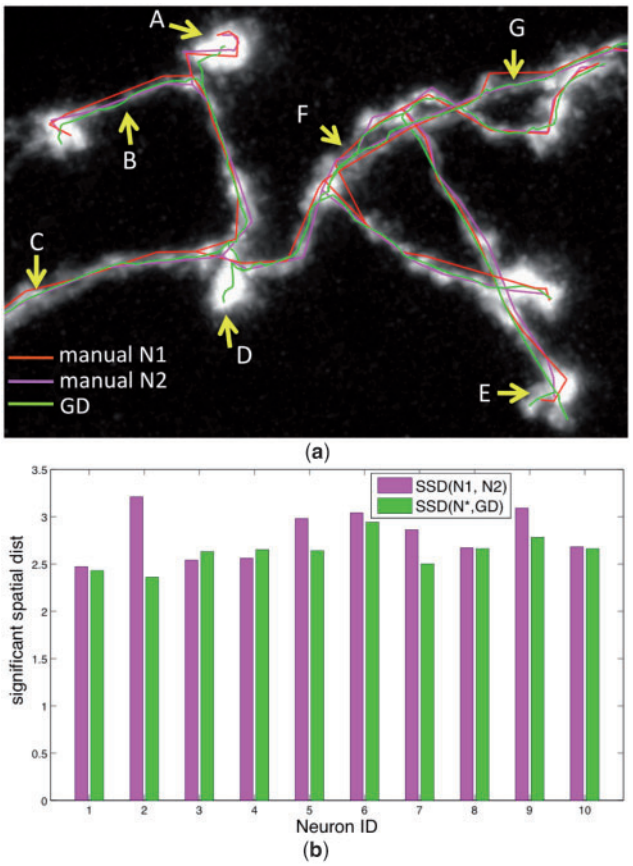


Fig. 4. Comparison of automatic GD reconstructions and manual reconstructions. (**a**) Visualizing the GD reconstruction and two independent manual reconstructions (N1, N2) using Neurolucida. Location pointers A–G indicate where GD significantly outperforms manual tracing. For clearer visualization, only the skeletons of traced neurites are displayed on top of the raw image. (**b**) SSD scores (in voxels) for the manual reconstructions compared to those between GD results and the respective best-matching manual reconstructions (i.e. SSD (N*, GD) = min{SSD (N1, GD), SSD (N2, GD)}.

(Neurolucida) and automatic tracing (NeuronStudio). We also applied GD to nervous systems of different animals, including fruit fly, *C.elegans* and mouse. Finally, we applied GD to cataloging different morphology in a fruit fly brain, building a 3D digital atlas of neurites in the fly brain, and estimating the distribution of synaptic boutons along axons in a mouse brain; these applications represent challenging tasks that have not been previously quantitatively studied, and thus also demonstrate the strength of GD.

3.1 Accuracy and speed of GD

We used 3D confocal images of 10 single neurons in a fruit fly brain (courtesy of T. Lee laboratory). An image has a size $512 \times 512 \times Z$ voxels (Z is about 70). A neuron in this dataset has 12–21 tracts (branches) in its full reconstruction. In total, there are 160 tracts. The punctuated structures in these neurons lead to uneven brightness and fuzzy structures in the respective images (Fig. 4a). In addition, morphologically these neurons have a lot of sharp turns and dense arborization (Fig. 4a), which make them hard to reconstruct even

manually. Indeed with the Neurolucida software, we spent 2 weeks in manually tracing their structures in two independent trials. The manual reconstructions were used as the 'ground truth' to evaluate the accuracy of GD reconstructions. When we used GD tracing, once the seeding step was done, each neuron was reconstructed within 20–30 seconds on a MacBook Pro laptop with 4 GB of memory.

We used the two independent trials of manual reconstructions as the baseline control to evaluate the accuracy of the automatic GD reconstructions. As shown in Figure 4a (locations A, B, C, E, F and G), GD produced a more accurate and also smoother reconstruction than the manual tracing. GD also correctly detected a missing branch that the human tracer missed twice (location D in Fig. 4a). Why is the automatic method better? It is because GD considers the 3D information comprehensively while it was quite difficult for a human to do so using Neurolucida.

We also quantified the reconstruction accuracy. Because any two reconstructions (denoted as R_1 and R_2) of the same neuron may have some difference in their structures and locations, we computed the 'distance' between R_1 and R_2 by averaging the reciprocal minimal spatial distances of their reconstruction nodes (Peng *et al.*, 2010). A larger distance means the greater discrepancy between R_1 and R_2. In addition, we noted that when such a minimal distance of individual reconstruction nodes was less than 2 voxels, visually it was hard to tell which reconstruction fit the image signal better. Thus, we averaged the reciprocal distances that were no less than 2 pixels. We called this score the significant spatial distance (SSD) between two reconstructions. Figure 4b shows that for 8 out of 10 neurons, the SSD of a pair of manual reconstructions is bigger than that of the automatic GD reconstruction and its best matching of the two manual reconstructions. The SSD scores of GD and manual reconstructions in the two 'failure' cases (neurons 3 and 4) are indeed comparable. Therefore, we have provided evidence that GD will produce an automatic tracing that is close to a good manual reconstruction, thus GD is very accurate; in addition, GD tracing is more stable than the manual reconstruction as the variation is smaller. Of note, changing the threshold for SSD computation, i.e. 2 voxel, did not change the conclusion (results omitted).

3.2 Robustness and consistency of GD

While Figure 4 already shows that GD can well trace punctuated neurites in real images, for a complete test of GD's ability in tracing a broken and fuzzy/noisy neurite, we produced synthetic test images that contained 'contaminated' structures. For an input image $I(p)$, where p is a pixel (or voxel), we generated a contaminated image $J(p)$ by multiplying $I(p)$ with a synthetic broken image mask $B_n(p;\sigma)$, followed by adding to it the white noise $\delta N(0,1)$ drawn from a normal distribution with mean 0 and SD 1:

$$J(p;\sigma,\delta)=I(p)B_n(p;\sigma)+\delta N(0,1) \tag{19}$$

$$B_n(p;\sigma)=\min_{i=1}^{n}\left\{1-\exp\left(-\frac{\|p-q_i\|^2}{2(\sigma\max(S_W,S_H,S_D))^2}\right)\right\} \tag{20}$$

where σ and δ are two factors controlling how broken and fuzzy the synthetic image is, n is the number of 'breaking' kernels used to produce such an image ($n=100$ in this article), q_i is the center of breaking kernel i, S_W, S_H, S_D are the width, height and depth (i.e. number of z-slices) of the image $I(p)$. With a greater σ we obtained a more severely broken image. Bigger δ leads to a fuzzier image. With this synthetic image model, we produced many comprehensive test cases (e.g. 2nd column of Fig. 5) for evaluating GD.

We used a 3D confocal image stack of a single fruit fly neuron (size = $512 \times 512 \times 60$ voxels) as the input. Figure 5 shows comparison results of GD and NeuronStudio (Wearne *et al.*, 2005) on test images of different broken and fuzzy levels. We can see that some test images are very challenging even for human (e.g. the last two test images of Fig. 5). In most tests, NeuronStudio only detected the continuous pieces and failed to produce complete neurite tracts (e.g. $\sigma=0.05$ and $\delta=0.15$). Differently, GD produced complete reconstructions for all test images. This demonstrates the robustness of GD.

We also quantified the consistency of GD tracing using data with different noise levels. We computed the spatial distance between the reconstruction of a contaminated image and the reconstruction of the original image without noise. As this is not a visual comparison, we set the SSD threshold as 0 (changing the threshold to 2 voxel as in Section 3.1 did not change the conclusion). As shown in Table 1, for different broken and fuzzy levels, GD produced a consistently smaller distance score than NeuronStudio in most cases, especially when the noise level was high. Indeed the average and SDs of these scores are 0.874 ± 0.324 for GD and 1.354 ± 0.673 for NeuronStudio, which are significantly different. Of note, since with a few seeds NeuronStudio could not produce a full reconstruction, we repeatedly added many seeds to make it be able to produce a comparable result to the GD reconstruction. Thus, the comparison in Table 1 very much favors NeuronStudio; yet, GD produced better reconstructions.

3.3 Applications in fruit fly: catalog neurite patterns, and build a 3D digital atlas of neurite patterns for the fruit fly brain

GD is a general method for tracing neurite tracts and any similar fibrous structures in an image, such as microtubule fibers. For neuron tracing, we have applied GD to a number of microscopic images of different model animals. Examples for fruit fly, *C.elegans* and mouse can be seen in Figure 1.

In a nervous system of a particular animal, usually there are multiple different types of neurons. In addition, with the development and aging of an animal, the morphology of the neurons in the same brain area also undergoes some level of change. With the GD tool, we are now able to catalog the morphology of different types of neurons, assess their morphological variations and study the distribution and connection of neurons more easily. A number of related biology problems could be quantitatively tackled, such as the stereotypy of whole-brain scale neurite distribution (Peng *et al.*, 2010) and the symmetry of a brain's wiring.

As a proof of principle study, we used GD to reconstruct the space-filling neurite patterns from 3D confocal images of 200 fruit fly GAL4 lines (collaborations with J. Simpson and G. Rubin laboratories) that have relatively distinctive expression patterns in the central nervous system. Each image was 3D aligned using our BrainAlinger pipeline (unpublished data of Peng laboratory), and thus has a size of 400 MB. The GAL4 pattern in an image, however, has a lot of background noise in the image data (e.g. Fig. 2A). This makes it hard to trace the neurite tracts. As far as we tested various tracing tools, GD was the only one that was able to produce reasonable reconstructions quickly (e.g. Fig. 2D). We

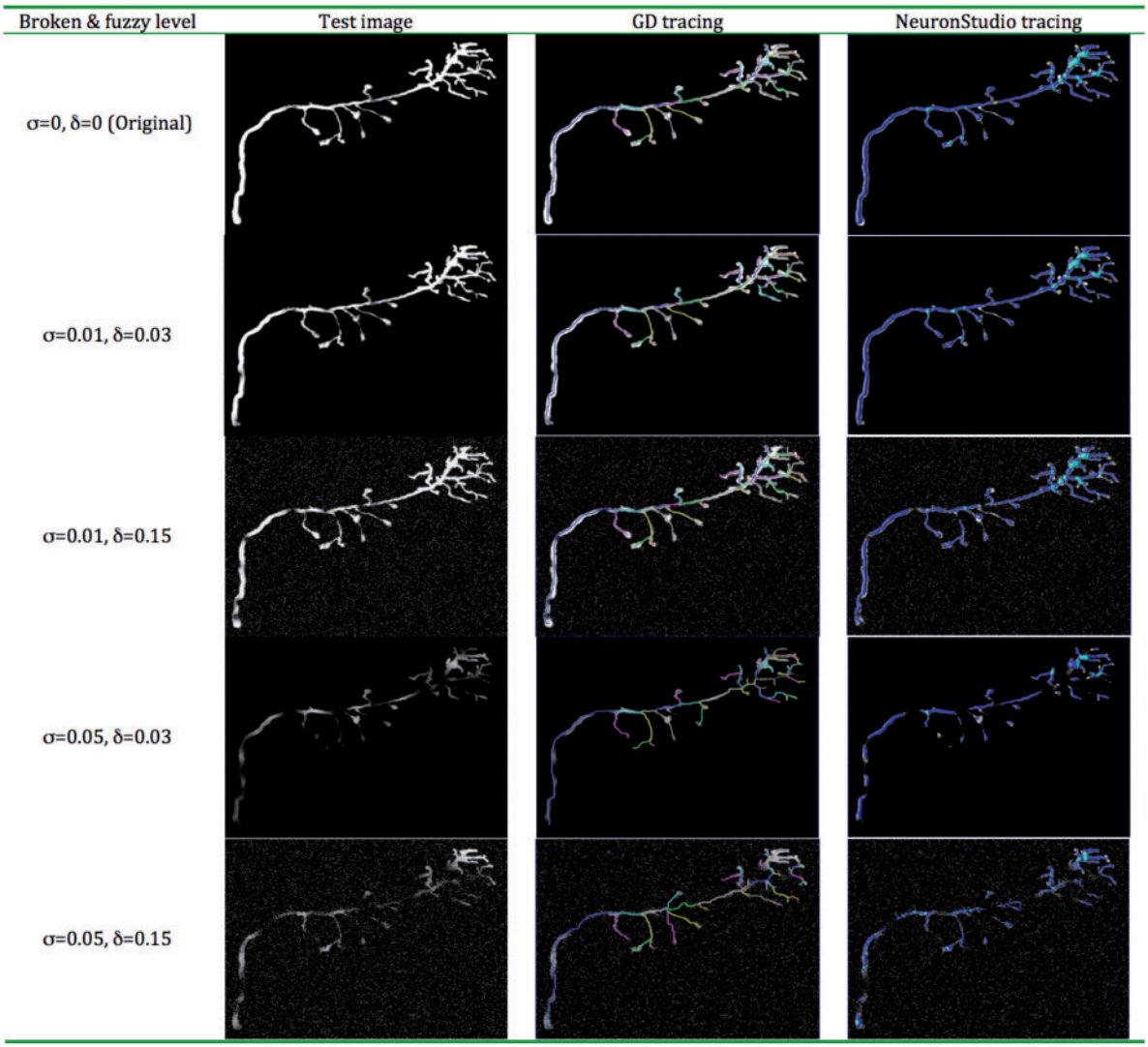


Fig. 5. Comparison of GD and NeuronStudio using synthetic images of different broken and fuzzy levels. For NeuronStudio, we used its default setting. Since a NeuronStudio tracing in most cases did not return a complete structure, we repeated multiple times at different seed locations to produce an as complete as possible reconstruction. For GD, we used the same set of seed locations in all tests. GD segments were color coded for better visualization.

thus cataloged these GAL4 patterns as demonstrated in examples of Figure 6A–E. Since the images containing these patterns were already registered in 3D, we conveniently assembled these patterns in a 'standard' coordinate system, and produced the first digital atlas of these patterns (Fig. 6F). We are currently analyzing the wiring diagram of GAL4 patterns derived from this atlas, based on which we hope to gain new insights into the structure of a fruit fly brain.

3.4 Application in mouse: estimate the punctuation rate of a brain area

Figure 1c shows the punctuated neurites in a mouse brain. Biologically, punctuations along an axon often correspond to boutons or synaptic terminals. It is of biological significance to detect these punctuated sites (P-sites) and calculate their density (punctuation rate). To faithfully capture the density of P-sites along an axon tract, we need to first reconstruct such a tract. As the second real application of GD, we used this method to trace 1006 axon tracts in 42 confocal images of a mouse brain (S. Sternson laboratory; see Fig. 1c for an example image). We also visually inspected these tracts, shown in Figure 7A and B, and found that they were meaningful.

We then profiled the regional intensity along each tract, and detected P-sites as the centers of local intensity maxima (Fig. 7B). We found this method was more robust than directly 3D segmentation of P-sites using watershed or adaptive template matching (results omitted due to the page limitation). This result is reasonable because the search space was constrained better when we detected P-sites only along neurites. This also makes it easier to identify the dark P-sites (yellow arrows in Fig. 7B).

The biologically interesting distribution of P-sites is shown in Fig. 7C. For the brain area in our images, we found the density

Table 1. Distances between GD/NS (NeuronStudio) reconstructions from noisy data and from original noise-less image

$\sigma \setminus \delta$		0.03	0.06	0.09	0.12	0.15
0.01	GD	0.284	0.405	0.526	0.505	0.378
	NS	0.257	0.476	0.559	0.592	0.509
0.02	GD	0.566	0.627	0.652	0.990	0.659
	NS	0.758	0.587	0.907	1.191	1.302
0.03	GD	0.785	0.992	1.003	0.922	1.003
	NS	1.206	1.143	0.847	1.670	1.785
0.04	GD	0.922	1.239	1.048	1.015	1.513
	NS	2.089	1.722	1.590	1.965	1.768
0.05	GD	1.208	0.989	1.234	1.206	1.282
	NS	1.784	1.928	2.235	2.673	2.273

The NS results were refined using the same method in Figure 5.

is about 0.22 P-sites per micron along an axon. In other words, the data suggested that along each axon in this mouse brain area, on average with every 5-micron increment there should be a putative bouton to accomplish certain synaptic information transformation.

3.5 Limitations of GD and solutions

While GD tracing is successful in tracing a number of neurite tracts, for other noisy patterns that have not been discussed here, GD may still have difficulties. For example, in an image containing two very close parallel tracts of which one is brighter than the other one, GD may be biased to find the brighter one. There are at least three possible solutions for this problem. First, giving a bigger weight to the smoothness term in our deformable model could produce a reconstruction with less sharp turns and thus may help detect the dimmer tract. Second, one can design a special repelling force in the GD framework to pull the two parallel tracts apart. Third, we can always use the V3D system (Peng *et al.*, 2010) to provide more prior knowledge such as the approximate shape information in addition to the two ends' locations of a tract, as well as to do post-editing of the traced structure.

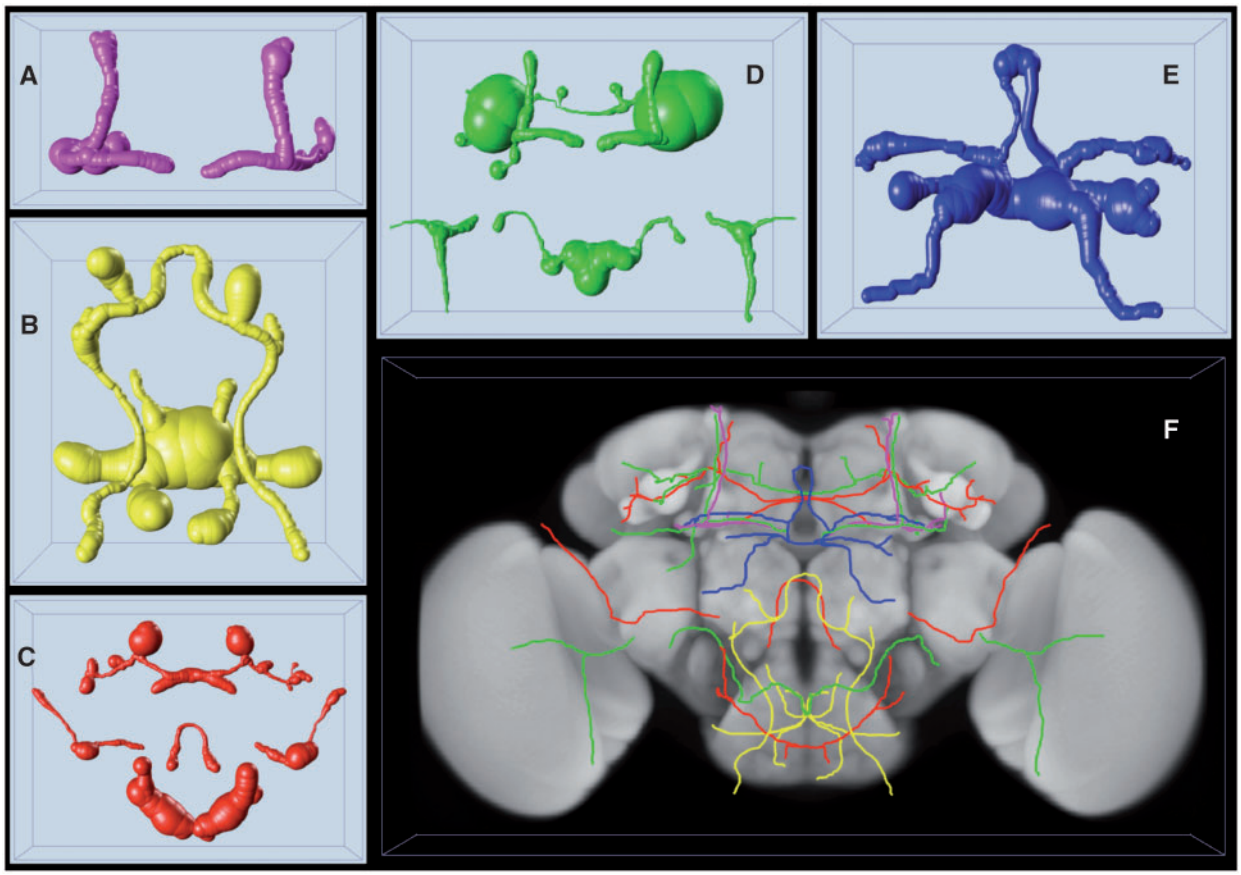


Fig. 6. Reconstruction and cataloging neuronal GAL4 patterns (**A**–**E**) of a fruit fly brain, and the 3D digital atlas of neurite patterns (**F**). In A–E, the space-filling models of the GAL4 patterns are displayed, where the width correspond to the thickness of a neurite tract or the estimated spanning range of the arborization. In F, for clearer visualization, only skeletons of the five GAL4 patterns are shown.

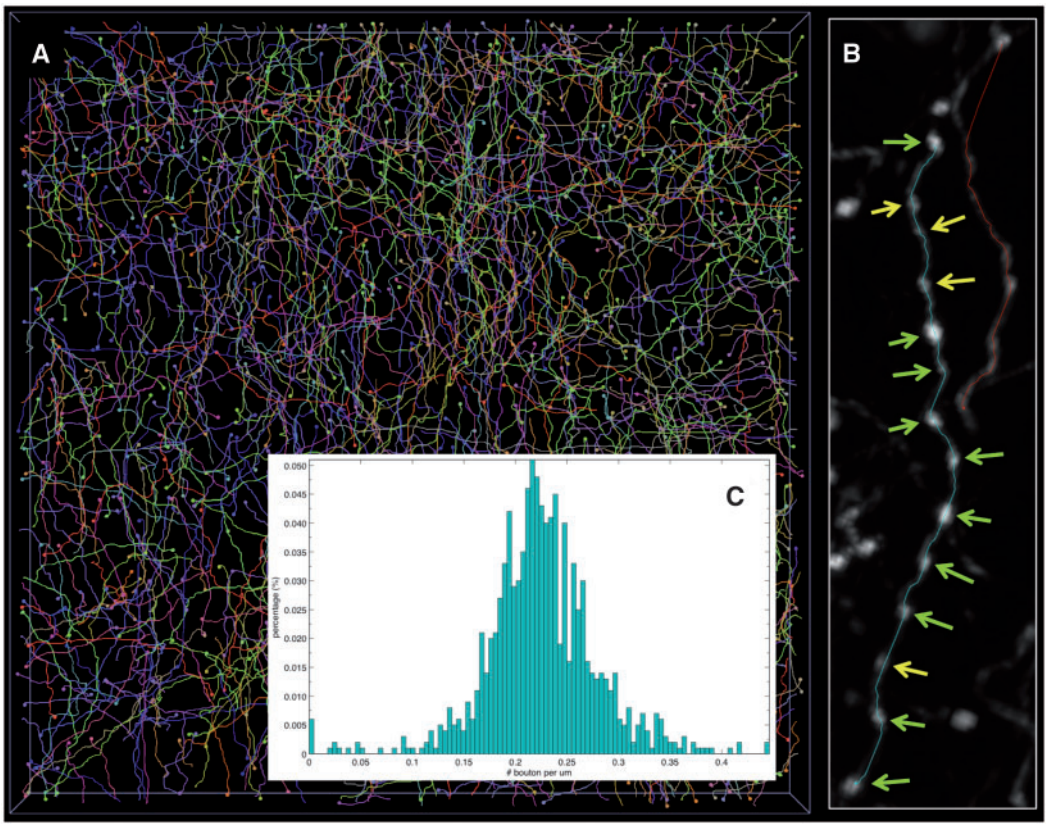


Fig. 7. Reconstruction of axon tracts in a mouse brain and estimation of putative bouton (P-site) density along axons. (**A**) 3D view of a database of 1006 automatically traced and manually verified axons tracts. The small dot at the tip of a tract indicates where the tract starts. (**B**) Two axon tracts (cyan and red) overlaid on the raw image. Green: bright P-sites. Yellow: dark P-sites that would be easy to miss if the neurite tract was not considered. (**C**) Histogram of P-site density of all axon tracts.

ACKNOWLEDGEMENTS

We thank Tzumin Lee laboratory, Gerry Rubin laboratory and Julie Simpson laboratory for some of the test data, Lei Qu and Benny Lam for help in neurite reconstruction, Fuhui Long, Gene Myers, Ting Zhao and Jun Xie for technical discussions, Margaret Jefferies for proofreading of the manuscript.

Funding: Howard Hughes Medical Institute.

Conflict of Interest: none declared.

REFERENCES

Al-Kofahi,K *et al.* (2002) Rapid automated three-dimensional tracing of neurons from confocal image stacks. *IEEE Trans. Inform. Technol. Biomedicine*, **6**, 171–187.

Al-Kofahi,K *et al.* (2003) Median-based robust algorithms for tracing neurons from noisy confocal microscope images. *IEEE Trans. Inform. Technol. Biomedicine*, **7**, 302–317.

Chelur,D.S. *et al.* (2002) The mechanosensory protein MEC-6 is a subunit of the C. elegans touch-cell degenerin channel. *Nature*, **420**, 669–673.

Dijkstra,E.W. (1959) A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**, 269–271.

Meijering,E. *et al.* (2004) Design and validation of a tool for neurite tracing and analysis in fluorescence microscopy images. *Cytometry,* **58A**, 167–176.

Peng,H. *et al.* (2008) Straightening Caenorhabditis elegans images. *Bioinformatics,* **24,** 234–242.

Peng,H. *et al.* (2010) V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. *Nat. Biotechnol.*, **28**, 348–353.

Perona,P. and Malik,J. (1990) Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Anal. Mach. Intell.*, **12,** 629–639.

Perry,W. *et al.* (2008) *Grand Challenges for Engineering*. National Academy of Engineering, Washington, DC.

Roysam,B. *et al.* (2009) The central role of neuroinformatics in the national academy of engineering's grandest challenge: reverse engineer the brain. *Neuroinformatics*, **7**, 1–5.

Wearne,S.L. *et al.* (2005) New Techniques for imaging, digitization and analysis of three-dimensional neural morphology on multiple scales. *Neuroscience*, **136**, 661–680.

Zana,F. and Klein,J.-C. (2001) Segmentation of vessel-like patterns using mathematical morphology and curvature evaluation. *IEEE Trans. Med. Imaging*, **11**, 1111–1119.

Zhang,Y. *et al.* (2007) Automated neurite extraction using dynamic programming for high-throughput screening of neuron-based assays. *NeuroImage*, **35,** 1502–1515.

Zhang,Y. *et al.* (2009) Detection of retinal blood vessels based on nonlinear projections source. *J. Signal Process. Syst.*, **55**, 103–112.