浙江大学

本科生毕业设计

开题报告

题目　在线交互式神经元重建的服务器系统

姓名　曲衡

学号　3130000569

指导教师　郑能干

年级与专业　2013 级 计算机科学与技术

所在学院　计算机科学与技术学院

一、题目：在线交互式神经元重建的服务器系统

二、指导教师对开题报告、外文翻译和中期报告的具体要求：

1. 开题报告要求：理解交互式果蝇神经元 3D 重建的技术需求，分析在线半自动神经元网络重建的服务器系统的技术难点，提出技术方案，分析可行性，制定研发计划。

2. 外文翻译要求：理解论文介绍的主要理论、方法和系统技术，用中文清晰准确表达，格式符合学校规定。

3. 中期报告要求： 理解 DVID 存储系统的基本架构，设计面向 DVID 的分布式数据服务，支持神经元网络的多分辨率 3D 可视化展示及多人编辑操作的后台系统设计，完成大规模神经元网络的重建工作。

指导教师（签名）_____

年　　月　　日

毕业设计开题报告、外文翻译的考核

导师对开题报告、外文翻译的评语及成绩评定：

成绩比例	开题报告占（20%）	外文翻译占（10%）
分值		

导师签名 _____

年　　月　　日

学院盲审专家对开题报告、外文翻译的评语及成绩评定：

成绩比例	开题报告占（20%）	外文翻译占（10%）
分值		

开题报告审核负责人 (签名/签章) _____

毕业设计中期报告考核

导师对中期报告的评语及成绩评定：

成绩比例	**中期报告占（10%）**
分值	

导师签名 _____

年　　月　　日

目 录

在线交互式神经元重建的服务器系统开题报告

1. 项目背景

将原始神经元图像信息进行神经元追踪和数字重建，有助于神经科学家直观地观察神经元结构，理解大脑运作的原理，甚至于探索智能的起源。因此，对大脑神经元相互连接形成的复杂网络进行数字重建一直是大脑神经科学家的目标之一。最近在神经元拓扑结构的自动化数字重建算法上取得了一些进展，如 Cannon RC 等人在海马神经重建上做出的工作[1]，Feng L 等人使用 mGRASP 在鼠类大脑上进行的重建工作[2] 均取得了出色的进展。

由于神经元拓扑结构的复杂性，在一些自动化重建结果的细节上仍然需要研究人员对数字重建的结果进行人工纠正和修改，以确保数字重建工作的准确性。另外研究人员需要对数字重建结果进行编辑，比如添加或删除一些网络分支等。为了便于研究人员编辑数字重建的结果，根据“所见即所得”的原则，设计出了 SWC 格式[3]。SWC 框架有以下特征: 清楚的 SWC 结构以及原始数据参考的可视化，明确定义的可操作单元，以及将用户输入和编辑操作直观地对应起来。在 SWC 格式的基础上，研究人员可以方便、直观地纠正自动化数字重建结果的错误，添加新的分支或删除已有分支。

尽管在 SWC 格式的基础上已经开发出了大量可以产生 SWC 格式文件的神经元结构重建软件，例如专注于半自动重建的 FARSIGHT [4]、半手动半自动重建的 Neuromantic [5]，以及一些其他主流数字重建工具 Vaa3D [6]、Neurostudio [7] 等等。但是这些软件并没有充分利用 SWC 格式的优点，并且为用户提供高效、精确重建的界面，在 [8] 中对这些软件的优缺点进行了详细的论述。为了充分发挥 SWC 格式高效，精确的优点，提供一个学习成本低，操作直观，并能提供相应的数据可视化功能，赵挺老师基于 SWC 格式，设计了名为 neuTube 的新软件[8]，同时具备 2D 和 3D 的可视化和直观的编辑、绘制功能。

由于 neuTube 是运行在单机的软件，无法满足多用户协同编辑修改的需求，也不利于数字重建结果的交流，无法共享完成数字重建的神经元结构。随着计算机性能和网速的提升，使得在线实时编辑神经元网络结构成为了可能。在此背景下，我们希望设计并实现一个在线多用户的神经元网络结构编辑平台，利用互联网便于数据共

享的特点，帮助神经学研究人员便捷地进行异地，多用户协同编辑神经元网络结构，并能分享完成数字重建的神经元结构，共同探索神经元结构下的奥秘。

2. 目标和任务

2..1 目标

我们的目标是设计并实现在线多用户的神经元网络结构编辑平台，利用互联网数据共享的特点，帮助神经学研究人员可以进行异地，多用户协同编辑神经网络结构，并能分享人工纠正、修改得到的神经元结构。充分发挥 SWC 文件格式准确，清楚，高效的优势，提供一个便于交互，同时具备 2D 和 3D 的可视化和直观的编辑、绘制功能的在线多用户神经网络结构编辑平台，并能处理多用户同时编辑时可能产生的冲突。

在这个平台上，用户可以上传神经回路的原始图像信息并储存在云端的服务器之中，利用 neuTube 或自己实现神经元网络结构的自动化数字重建工作。使用 SWC 格式储存数字重建工作的结果，作为后续修改、编辑的基础。平台包含数据可视化的功能，将数字重建结果以 2D 和 3D 的形式展现给用户。用户可以对数字重建结果进行人工编辑、修改，纠正自动数字重建结果中不精确的细节。用户也可以在已经完成的数字重建结果的基础上，删除已有分支或添加新的分支。

由于允许多用户同时编辑同一份 SWC 文件，这可能会造成文件内容的冲突。我们需要提供一套文件冲突解决方案，在文件内容产生冲突的时候尽可能地自动解决冲突，并能将实在解决不了的冲突准确、直观的表述给用户，便于用户手动解决冲突。

2..2 任务

设计完成上述需求的神经元网络结构编辑平台，将原始神经元图像信息应用于本平台，由神经科学家通过本平台编辑，修改完成数字重建的神经元结构，并根据反馈提高平台的可用性。对系统的要求：

1. 交换界面简洁明了，操作简洁高效
2. 对多用户协同编辑提供支持
3. 能够解决多用户协同编辑产生的冲突

4. 用户操作延迟较低，没有明显的卡顿

3. 可行性分析

郑能干老师和赵挺老师作为指导老师，在神经网络结构重建与编辑领域具有多年的工作经验，提供了清晰、明确的工作方向，在可能遇到问题的地方提供指导和帮助，为解决潜在的问题提供了保障。neuTube 作为已经实现的单机版本，明确了项目需求，展示出一些可能存在的问题，使我对该项目的理解更加深入透彻。技术方案里面涉及的很多技术是由赵挺老师所在实验室开发的，甚至是由赵挺老师本人编写完成的。两位指导老师提供的技术指导大大提高了项目可行性。

项目组积累了大量原始神经元切片图像数据，有充分的测试数据完成平台设计与开发。

项目前期对项目可能涉及到的技术做了充分的研究，尝试，对顺利完成项目内容，达成项目目标有充分的信心。

由于本项目不涉及硬件方面，项目经费需求较小。项目涉及到的服务器和开发平台均能利用现有实验室资源完成，可行性较高。

4. 初步技术方案和关键技术考虑

技术方案涉及原始图像信息储存与数字重建结果储存，用户数据信息储存，网络应用开发以及前端可视化展示，共 4 个部分。

4..1 原始图像信息与数字重建结果储存

我们初步计划采用 DVID 储存原始图像信息与数字重建结果。DVID 是一个分布式面向图像的数据服务，主要用于图像分析与可视化。DVID 有如下特点：

1. 便于扩展数据类型，允许用户根据数据特点加速访问速度，减少储存空间，提供方便的 API。这为储存数字重建结果提供了便利。

2. 为分布式数据储存提供了类似于 GIT 的版本控制系统，在此基础之上我们可以解决多用户同时编辑产生冲突的问题。

3. 方便连接其他 API 如 Google BrainMaps 和 OpenConnectome 等。

4. 支持多分辨率图像数据，使得用户可以在不同尺度下观察图像信息。

在 DVID 的基础上，可以构建出多用户的原始图像信息与数字重建结果储存仓库，将数据储存抽象成数据存储服务，便于专注于完成核心算法和逻辑。

4..2 用户数据信息储存

初步计划采用 PostgreSQL 数据库储存用户信息。PostgreSQL 最初由加州大学伯克利分校计算机系开发完成。在支持大部分 SQL 标准之上，提供了许多诸如复杂查询，多版本并行控制，事物完整性等现代特性。由于 PostreSQL 对标准 SQL 支持度较高，可以方便的和 DVID 联系起来，将用户信息和原始图像信息，数字重建结果对应起来。利用 PostgreSQL 支持的储存过程，事物以及多版本并行控制特性，我们可以方便的实现分布式，多用户实时编辑平台，并解决多用户同时编辑可能产生冲突的问题。

4..3 网络应用开发

初步计划采用 Node.js 和 Express 完成网络应用开发。Node.js 是一个基于 Chrome V8 引起的 JavaScript 的运行环境。Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。Express 是一个基于 Node.js 平台的极简、灵活的 web 应用开发框架，提供丰富的 HTTP 快捷方式和任意排列组合的 Connect 中间件，帮助快速、简单的创建健壮、友好的 API。基于这两个工具，可以方便快速的搭建一个无状态的服务器，初步设计出一个分布式计算服务结构如图 1 所示，便于针对用户数量以及数据请求量进行扩展。搭配之前采用的分布式数据库，网络应用平台的可扩展性较高，可以用来应对多用户同时编辑大数据量的数据，并提供相应可视化算法所需的计算服务的应用场景。

4..4 前端可视化展示

这部分工作主要由同组其他同学完成，主要用到的技术有 Three.js 和 React 等，这里不再赘述。

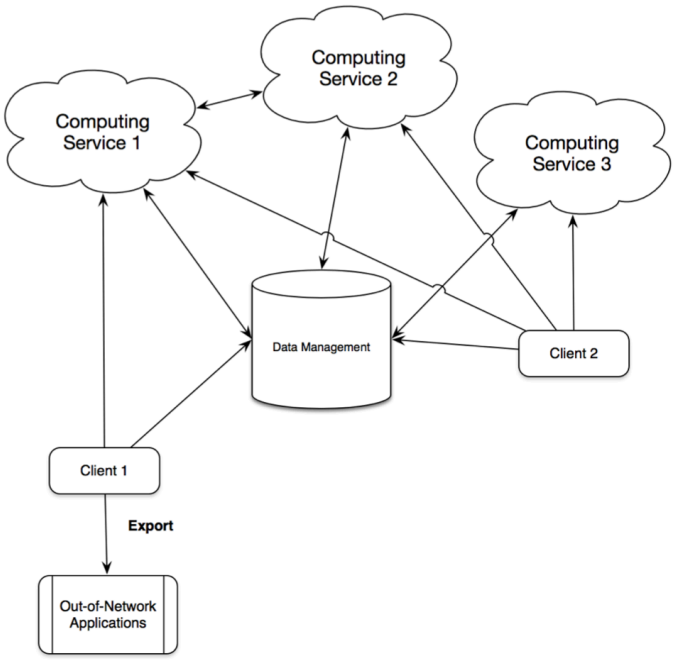

图 1　分布式计算服务结构

5.　预期工作结果

1. 神经元结构编辑平台

设计并实现一个同时具备 2D 和 3D 的可视化界面，为用户提供直观的编辑、绘制功能，支持多用户同时在线并可以协同编辑的神经元结构平台。实验室其他同学完成数据可视化工作，我主要实现平台的后端支持。

2. 同时在线用户数测试报告

对神经元结构编辑平台进行压力测试，根据测试结果进行性能调优，使之支持尽可能多的同时在线用户数量（数千用户同时在线，取决于平台后端计算机的性能和数量）。完成性能调优后提供一份完整的性能测试报告。

3. 用户操作延迟测试报告

对前端可视化操作的响应时间进行压力测试，根据测试结果进行性能调优，使用户操作的响应时间在毫秒级别（大量数据的上传时间取决去客户端所处网络环境）。对各个操作的响应时间及优化结果提供详细的测试报告。

6. 进度计划

本项目的整体计划如下表所示：

时间	主要工作
2017 年 3 月 1 日至 3 月 15 日	完成技术方案设计
2017 年 3 月 16 日至 3 月 29 日	完成多用户平台设计
2017 年 4 月 1 日至 4 月 15 日	初步完成需求的功能点
2017 年 4 月 16 日至 4 月 30 日	根据老师意见进行功能迭代，开始撰写论文
2017 年 5 月 1 日至 5 月 30 日	根据测试结果进行性能调优，修改并完成论文

表 1　项目进度计划

参考文献

[1] CANNON R C, TURNER D A, PYAPALI G K, et al. An on-line archive of reconstructed hippocampal neurons[J]. Journal of Neuroscience Methods, 1998, 84(1－2): 49－54.

[2] DRUCKMANN S, FENG L, LEE B, et al. Structured synaptic connectivity between hippocampal regions[J]. Neuron, 2014, 81(3): 629.

[3] PENG H, LONG F, ZHAO T, et al. Proof-editing is the bottleneck of 3D neuron reconstruction: the problem and solutions[J]. Neuroinformatics, 2011, 9(2): 103－105.

[4] LUISI J, NARAYANASWAMY A, GALBREATH Z, et al. The FARSIGHT Trace Editor: An Open Source Tool for 3-D Inspection and Efficient Pattern Analysis Aided Editing of Automated Neuronal Reconstructions[J]. Neuroinformatics, 2011, 9(2): 305－315.

[5] MYATT D R, HADLINGTON T, ASCOLI G A, et al. Neuromantic －from Semi-Manual to Semi-Automatic Reconstruction of Neuron Morphology[J]. Frontiers in Neuroinformatics, 2012, 6: 4.

[6] PENG H, LONG F. Seeing more is knowing more: V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets[J]. Nature Biotechnology, 2010, 28(4): 348－53.

[7] RODRIGUEZ A, EHLENBERGER D B, HOF P R, et al. Rayburst sampling, an algorithm for automated three-dimensional shape analysis from laser scanning microscopy images[J]. Nature Protocol, 2006, 1(4) : 2152 – 2161.

[8] FENG L, ZHAO T, KIM J. neuTube 1.0: A New Design for Efficient Neuron Reconstruction Software Based on the SWC Format[J]. Eneuro, 2014, 2(1).

本科毕业设计外文翻译

文献原文：

L. Feng, T. Zhao, J. Kim. neuTube 1.0: A New Design for Efficient Neuron Reconstruction Software Based on the SWC Format.

基于 SWC 文件格式的一种高效神经重建的新设计

摘要

大脑回路的绘制需要在神经拓扑形成的复杂网络上进行数字重建。尽管最近几年在自动化算法上取得了一些进展，但是由于缺少高效的算法并支持用户编辑的软件，神经结构的重建依旧是大脑回路绘制的瓶颈。因此我们设计了一种基于 SWC 文件格式的新软件。SWC 格式是一种被广泛用于分析神经结构或者在线神经重建的标准神经形态学的格式。同时在我们的开源软件 neuTube 1.0 中实现了这种设计。这个软件同时具备 2D 和 3D 的可视化和直观的编辑、绘制功能，允许用户有效的根据荧光图像数据重建神经结构并且编辑由其他软件生成的标准的神经元结构文件。通过和其他两种软件工具的比较（Neuromantic 和 Neurostudio）展示出 neuTube1.0 的优点。这个软件现在可以在 http://www.neutracing.com 上获取，并提供了完整的软件文档和视频教程。

1. 简介

从光学显微镜图像中进行神经拓扑结构的数字重建或者绘制是大脑回路构建中的重要一步。在这个任务中，输入是图像信息，输出通常是可以被 SWC 格式可以描述的树形结构。尽管已经开发了大量可以产生 SWC 格式文件的神经重建软件，但是没有充分利用 SWC 格式并且为用户提供高效精确重建的界面。一个最优的用户界面意味着用户可以用最小的认知负担与软件交互，这就需要清楚的数据可视化和直观地操作。换句话说，根据软件提供的视觉信息，用户应该能够迅速找出底层 SWC 模型，知道如何操作模型并得到操作的结果。有了这些标准，我们可以识别许多绘制软件应用程序的缺点。例如 FARSIGHT 其重点是半自动的重建神经元的结构，不提供直观的低级编辑功能来纠正细小的错误。Simple Neurite Tracer，Fiji 的一个流行的插件，也缺乏编辑功能。主流的商业软件 Neurolucida，允许完成手工重建神经元结构的结构。然而它不支持神经元重建的 3D，尽管 3D 交互可以提高重建过程的速度和准确性。其他受欢迎的软件工具如 Neuromantic 和 Neurostudio，同样缺乏先进的 3D

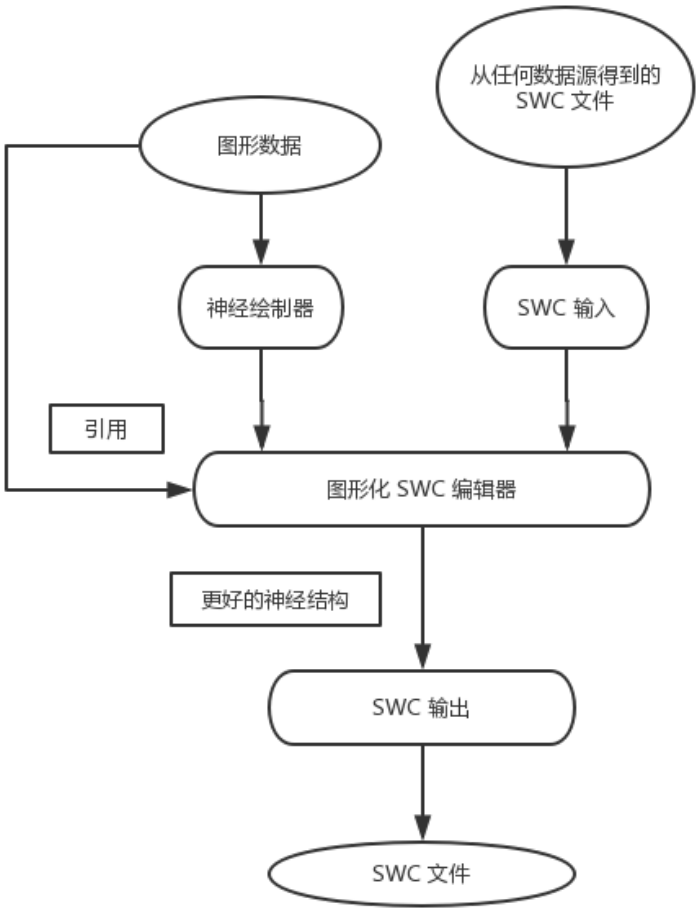

图 1　SWC 框架的整体布局

编辑功能。Vaa3D 提供创新的交互式 3D 神经元绘制功能，但是这些功能在 2D 层面用来确定密度或其他细小结构并不可用。

在这里，我们基于 SWC 格式提出一种新的、综合软件设计用来克服当前软件的多种局限性。在我们称之为 SWC 框架的基础上，我们开发了 NeuTube，一种结合了鲁棒的自动绘制算法的高效的自动重建的软件工具，提供了通用的用户友好的 2D 和 3D 编辑功能。

2. 材料和方法

2..1　SWC 框架

SWC 框架的整体布局如图 1 所示。这种结构的软件以原始图像或一个 SWC 文件作为输入，输出的是满足用户需求的神经元结构。SWC 框架遵循的原则是“所见

即所得”，用户正在编辑的结构被可视化出来，检查结果时不需要任何的第三方软件来展示结果。因此，SWC 框架包括以下特征: 清楚的 SWC 结构以及原始数据参考的可视化，明确定义的可操作单元，从用户输入和编辑操作直观的对应起来。除了图像可视化，这些特性是建立在 SWC 格式的基础之上。在这里，我们为了详细描述如何构建操作 SWC 模型，首先用抽象的方式定义了一个模型。

2..1.1 SWC 模型的抽象定义

从数学的角度来看，SWC 模型可以用点集来定义：$\{n_i = (x_i, y_i, z_i, n_j)|i = 1, ..., N, j = 0, ..., N, i \neq j, x_i, y_i, z_i, r_i \in R\}$ 每一个点 n_i，都是一个中心在 (x_i, y_i, z_i)，半径为 r_i 的球。n_0 是一个空点，用来表示一个神经结构的根，n_j 用来表示 n_i 的父节点。一个向上的路径从 n_i 到 n_j 可以用一个数组 $(n_{k_1}, ..., n_{k_n})$，其中 n_{k_i+1} 表示 n_{k_i} 的父节点 $k_1 = i, k_n = j$。一个合法的神经树形结构，是不允许有环的存在。在这个模型中，基础结构单元是点，点定义了我们如何设计可视化和交互的方案。

2..1.2 SWC 操作

假设 S_1 和 S_2 是两个点集，对于一个神经结构的操作可以定义为：

$$f(S_1) = S_2$$

举例来说，$f(\{n_1, ..., n_n\} = \phi), \phi$ 代表空集，定义了删除操作。然而一些操作可能产生一些表示非法神经结构的点集。如何构造合法的操作取决于如何用合适的数据结构描述模型。在我们的框架中，我么使用冗余的元组来储存点集：$n = (G(n), P(n), C(n), S(n))$, 其中 $G(n) = (x(n), y(n), z(n), r(n))$。这种方式定义了一个在 $(x(n), y(n), z(n), r(n))$ 处，半径为 $r(n)$，父节点为 $p(n)$ 的节点 n。n 的第一个子节点为 $C(n)$，下一个兄弟节点为 $S(n)$。这样的冗余的设计是为了提高遍历节点的效率。例如，要查询子节点时，该程序只需要检查它的第一个孩子和遍历其兄弟节点即可。而在非冗余结构中，每个节点只与其父节点相连，程序可能需要检查树中的每个节点。

编辑一个节点 n 的值可以用修改对应的元组来定义。我们将对 $G(n)$ 的改变称为几何操作，对 $P(n), C(n), S(n)$ 的变换成为结构操作。一个几何的变换是很直观的，但是一个在结构上的操作很可能造成非法的神经结构。例如仅改变 $P(n)$ 会打破 $P(C(n)) = n, P(n) = P(S(n))$ 的规则。为了避免这样的问题，我们在三个层次上构建对 SWC 的操作。第一级包含三个基本操作，可以用以下公式定义：

$$f_p(\{n\}|n') = f_p(\{G(n), P(n), C(n), S(n)\}|n' = \{(G(n), n', C(n), S(n))\})$$

$$f_p(\{n\}|n') = f_p(\{G(n), P(n), C(n), S(n)\}|n' = \{(G(n), P(n), n', S(n))\})$$

$$f_s(\{n\}|n') = f_p(\{G(n), P(n), C(n), S(n)\}|n' = \{(G(n), P(n), C(n), n')\})$$

在这个层次上，并不保证结构的合法性。

第二层次包括简单有效的操作。假设 $F_{p_0}(n)$ 代表将节点 n 的父节点设为 n_0, 即

空节点, 如果 $C(P(n)) = n$。即 n 是父母的第一个孩子。

$$F_{p_0}(n) = \begin{cases} f_s(\{n\}|n_0) \circ f_p(\{n\}|n_0) \circ f_c(\{P(n)\}|S(n)), C(P(n)) = n \\ f_s(\{n\}|n_0) \circ f_p(\{n\}|n_0) \circ f_c(\{S^{-1}(n)\}|S(n)), otherwise \end{cases}$$

其中，$f \circ g$ 代表一个符合操作，$S(S^{-1}(n)) = n$。为了更清楚的定义一个单节点，在不增加歧义的情况下，$F_{p_0}(n)$ 被定义为一个函数而不是一个点集。设置父节点的操作被定义为

$$F_p(n|n') = f_c(\{C(n')\}|n) \circ f_s(\{n\}|C(n')) \circ f_p(\{n\}|n') \circ F_{p_0}(n)$$

设置 n 作为 n' 的第一个子节点虽然可以用其他的操作合成出来，但是在实践中，定义更多的操作是十分有用的。

$$F_s(n|n') = f_s(\{n'\}|n) \circ f_s(\{n'\}|S(n)) \circ f_p(\{n'\}|P(n)) \circ F_{p_0}(n')$$

第三层次由一组复合操作组成，其中包括任何操作组成的第二层次的操作。在这个层次，我们将操作分为两种类型形态依赖和形态独立。如果操作的结果依赖于节点的位置或尺寸，那么这个操作被称为形态依赖，否则被称为形态独立独立。将一个操作分解为基本操作有助于保证对神经元结构进行的操作的有效性。更重要的是，分解基础操作有助于实现撤销和重做任意操作。撤销操作要求撤消任意复杂度的操作。例如，删除的逆运算需要恢复多个相邻的节点。直接推导逆操作不仅需要大量的工作，也容易导致很难绘制错误。将一个操作分解为一系列基本操作之后，我们可以很容易构造撤消操作的逆转序列。

2..1.3　用户交互

绘制软件基本功能是根据用户的输入改变神经元的形态学结构，这通常是由鼠标点击和键盘输入。因为我们定义的操作为将一组节点映射到另一个节点，用户交互从节点的选择开始，这就需要两个组件：SWC 可视化和对用户输入的响应。高质量的可视化神经元是成功的神经元编辑最重要的特征。查看结构的能力显著减少了检查错误所需要时间。提供 2D 和 3D 视图在不同方面体现了独特的优势。例如，3D 视图非常适合显示一个全球结构, 而 2D 视图适合在结构密集的地方的精确显示。选择一个节点最直观的方法是将鼠标光标移到节点上然后单击。这需要将屏幕光标的坐标映射到 3D SWC 空间。同时应该支持选择一组节点作为一个操作的输入。选择后，用户可以用一些输入触发一个操作。所以操作变成了

$$f(S_1\Theta) = S_2$$

Θ 指的是用户输入的参数。举例来说，$f(\{n\}|(x,y,z)) = \{n, F_p((x,y,z,r(n)), n_0, n_0, n_0)|n\}$，定义了从节点 n 扩展一个分支到 (x,y,z)。

2..1.4　从图像信号创建 SWC 节点

对于任何独立的神经绘制软件软件，必须允许从原始图像信号重建神经元结构。在 SWC 框架中这个函数可以定义为

$$g(S_1|\Theta, I) = S_2$$

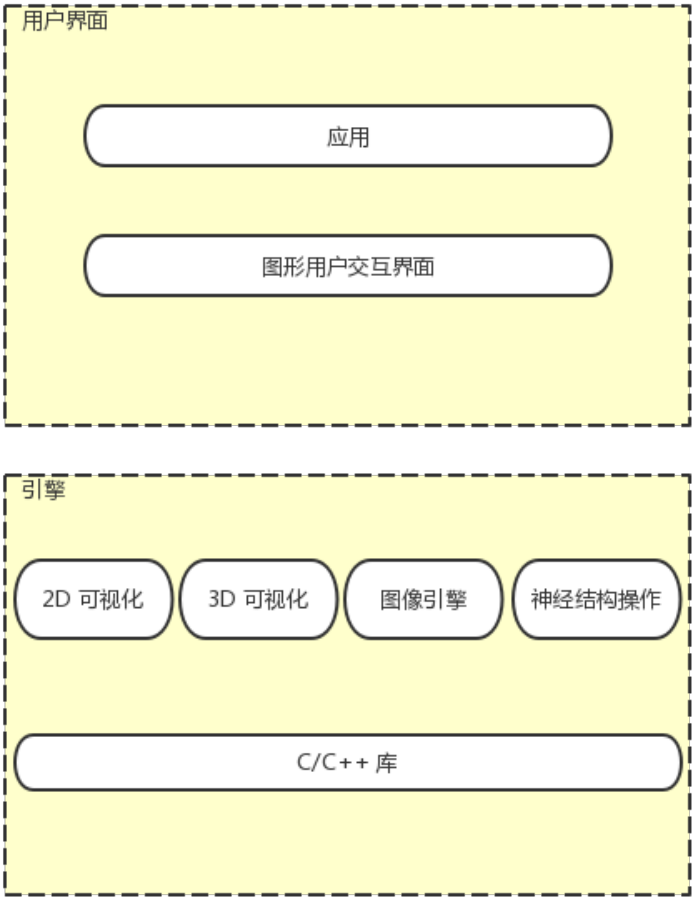

图 2　SWC 核心模块

其中 I 是图像信号。需要注意的是，这里实际上定义了 SWC 操作的超级。如果一个函数是和图像无关的，那么就和之前定义的 SWC 操作相同。和图像有关操作的一个例子是计算最短路径，如果定义 $S_1 = \{n_i, n_j\}$ 为原点和终点集，$S2 = \{n_i, n'1, ..., n_k, n_j\}$ 为源点到终点的最短测地线上的的采样点。那么采样点的半径可以用自动估计或者线性查找来计算，这就取决于操作是如何定义的。

2..2　软件实现

2..2.1　结构

如图 2 所示，在 SWC 框架上，我们在 4 个核心模块：2D 可视化，3D 可视化，图像分析和神经结构实现了图形化用户界面的 neuTube 1.0。

2..2.2　2D 可视化

二维可视化模块提供的显示 3D 图像和神经元结构切片，以及允许用户与 2D 显示交互的功能。这个模块便于用户仔细检查和精确编辑。例如，用户可以放大感兴趣

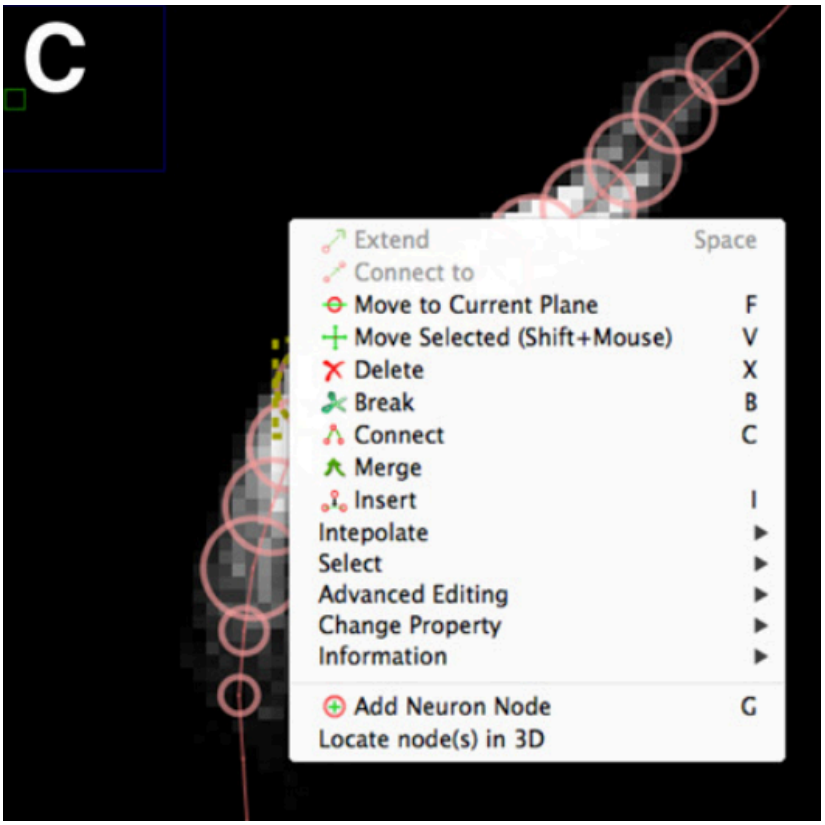

图 3　基本几何元素与相关功能

的区域，定位精确跟踪点，或者应用细粒度组件优化神经元结构。2D 可视化的目的是展示重建的结果和数据是否相符而不是现实的神经元结构。如图3 所示，我们只使用两个几何元素：线和圆圈来代表一个神经元的形态。

二维可视化便于显示恰好在一个平面上的点，但是不适合展示和平面垂直的地方。我们使用两个策略来解决这个问题。首先，为每个被平面穿过的点展示展示一个圆圈。相对应的圆越大，横截面面积越大，这就通过圆的面积告诉用户节点到平面的距离。其次，我们使用颜色来区分一个节点是否在平面上。颜色越饱和，越不透明，节点距离平面越远。着色选项根据用户的反馈和调整的人工调整，然后使用不可变的软件软件参数生成。为了允许用户查看神经元结构的球状结构，我们也将项目整个骨架片视图到一个切片上。

2..2.3　3D 可视化

如图 4、5 所示，3D 可视化模块是为了实时渲染 3D 图像和神经结构。在 3D 可视化窗口中，用户可以直接展示神经轨迹并编辑，并且任何的改变会同步反映在 2D 可视化窗口中，反之亦然。

2..2.4　图像分析

这个模块提供了自动绘制的神经元或神经元分支，允许用户以最小的交互获得神经元结构。例如，用户只需要选择单击分支上的一点便可以选择一个分支。本文之前论述的一个主要改进是用 SWC 框架中定义的树状模型替代了圆柱模型。此外，我

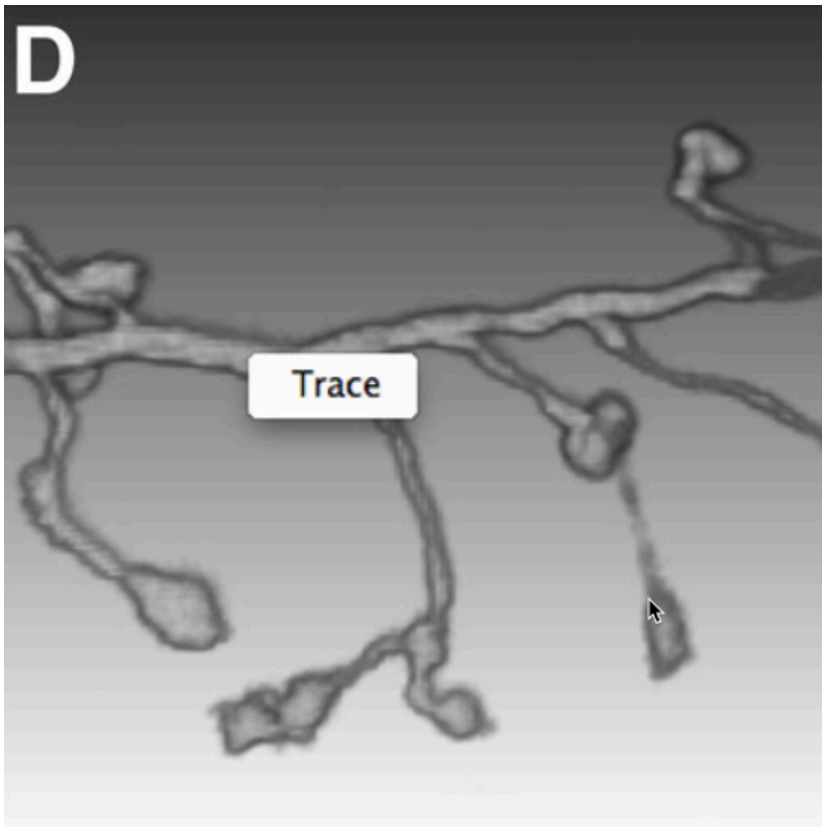

图 4　原始图像实时渲染效果

们实现了一个基于最短路径方法的用来自动重建的点对点绘制函数。这个函数类似于 Simple Neurite Tracer 和 Vaa3d，但我们可以在 2D 和 3D 同时使用。

2..2.5　神经结构操作

这个模块提供了对任意节点的神经结构编辑功能。用户可以通过鼠标点击或键盘快捷键来改变神经结构的几何和拓扑结构。根据前文所述，这个模块将操作分成不同的层次。为了减轻工作量，我们构建了一下高层次的操作。插入：在很多情况下，一个神经分支或者一个端分支是足够光滑到用线性结构表示。利用这个性质，如图 6 所示，可以允许用户快速纠正多个节点的几何属性。设置分叉点：两段神经元交叉的地方很容易出现分叉点错误的情况。完整的手工纠错包括选择两个节点并将它们合并在一起。设置分叉点的简化了用户的操作，如图 7 所示，用户只需要创建一个分支点即可。重置分叉点：这个操作提供了修正一个分支点的另一种方式。在这个操作中，如图 8所示，用户选择一个节点，程序将尝试移动相邻分支结构到对应节点。程序自动会根据角度自动确定移动哪个分支。连接多个节点：连接两个接节点是最基本的操作之一，但是仍然需要耐心的完成多个步骤，包括选择节点和触发连接的命令。人工操作的数量会随着节点需要连接的数量增加。为此，如图 9 所示，neutube 根据两两距离的最小生成树提供了自动连接多个节点的操作。消除拐角：定义将三个节点依次连接形成的锐角为拐角。中间的节点称为转折点，另外两个节点称为侧节点。如图 10所示，删除操作是将转折点用两个侧节点差值形成的新节点替代。消除交叉点：

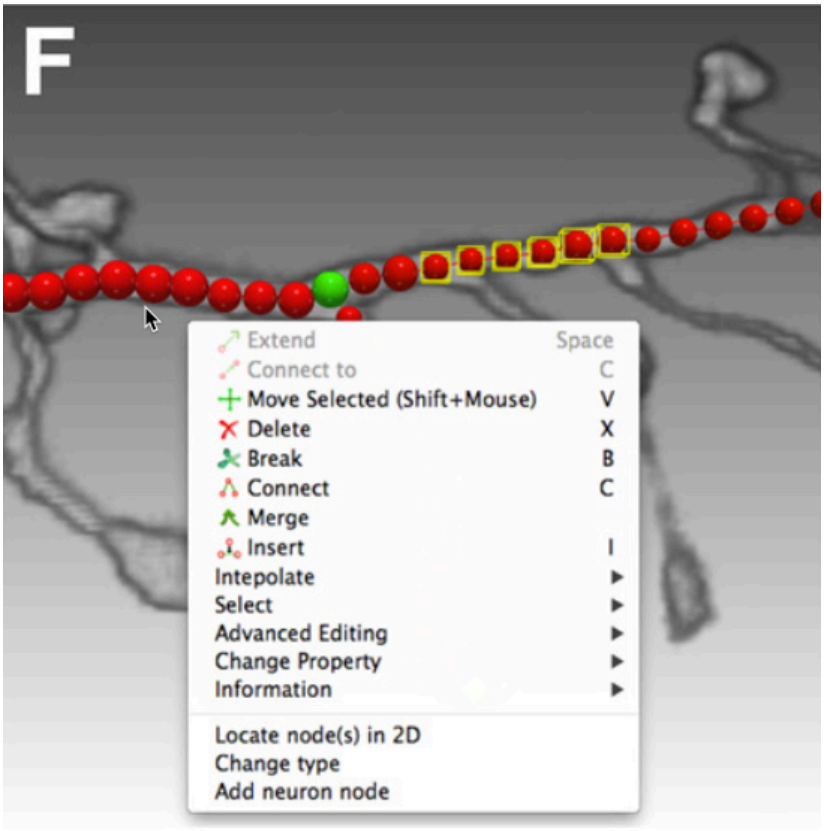

图 5　数字重建结果实现渲染效果

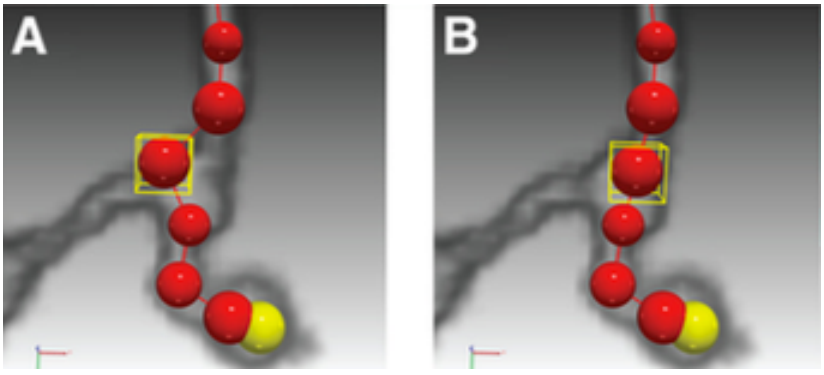

图 6　神经结构操作示例 1

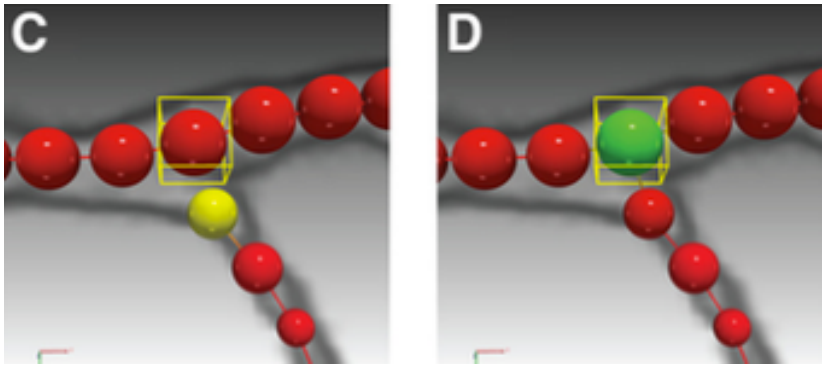

图 7　神经结构操作示例 2

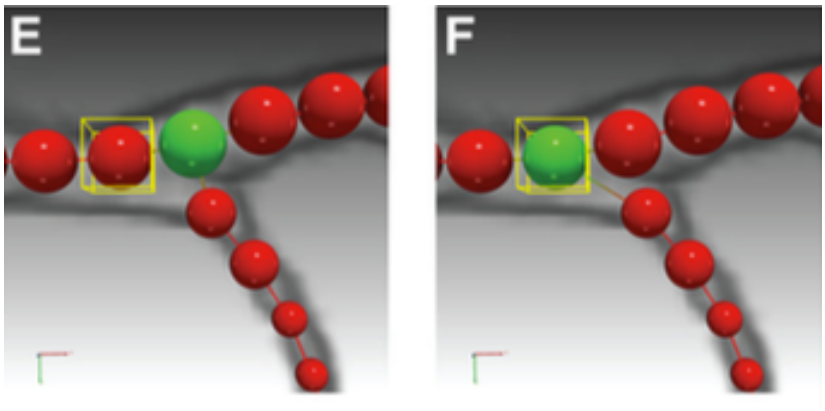

图 8　神经结构操作示例 3

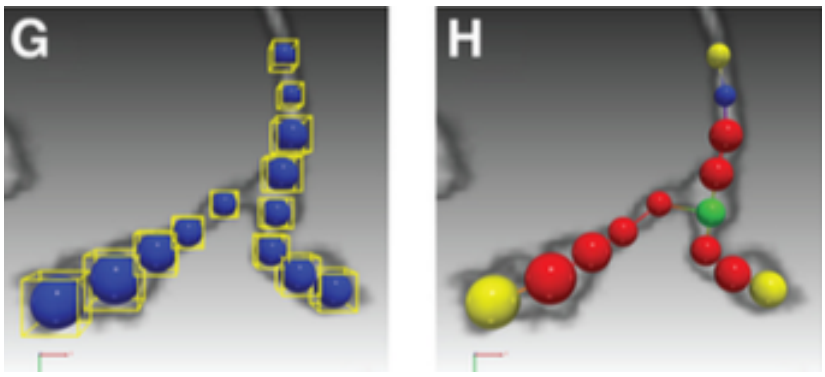

图 9　神经结构操作示例 4

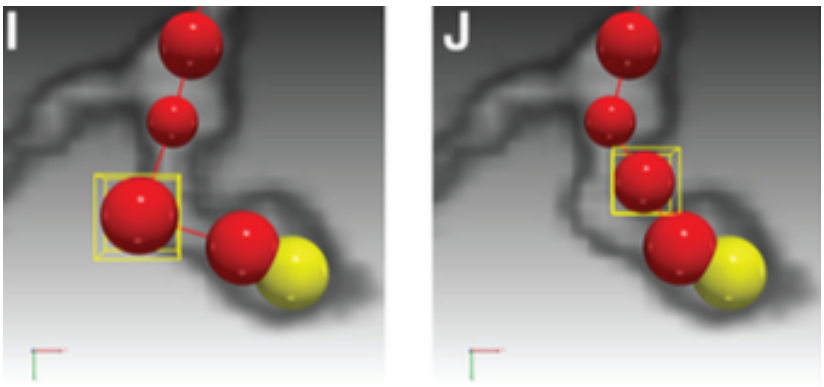

图 10　神经结构操作示例 5

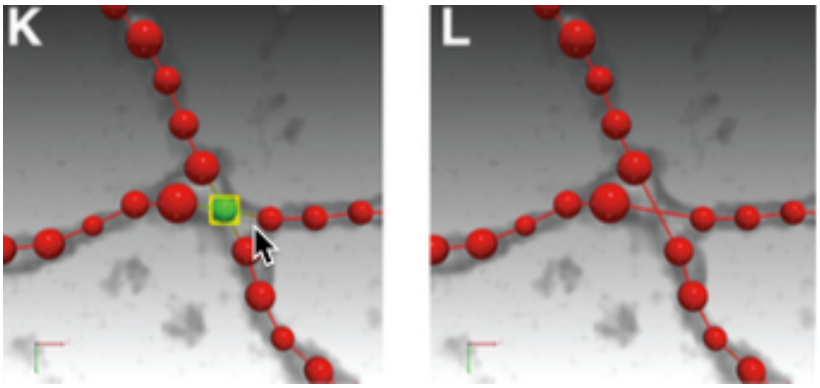

图 11　神经结构操作示例 16

交叉是一种常见的绘制错误，当两个分支在某一点里的很近时。纠正一个分支需要连接和分离多个节点。如图 11所示，为此我们添加了消除交叉点的操作。

2..2.6　实现

软件由 C/C++ 编程语言和多个第三方库编写完成。主要的第三方库是 Qt，Qt 提供了跨平台的 GUI 开发框架。3D 可视化模块建立在 OpenGL 2.0 用 GLSL 编写完成。我们编写高效的生成两种基本几何元素（球和圆柱）的着色器，实现了一个可以快速渲染神经结构的引擎。我们所有的几何基元都有调整不透明度的选项，可以根据可视化的顺序需求生成一个合理的半透明的场景。对于实现渲染复杂的半透明的场景的功能，我们也实现了双深度剥落和加权平均融合算法，这是两种常用的顺序无关的透明算法。这两个算法不需要高端图形卡提供的特殊硬件特性，提供了 neuTube 1.0 能够呈现复杂场景的软件可移植性。用户可以根据实际场景，在运行时切换性能更好的算法。为了在 3D 场景中显示包含了原始图像信息和重建神经信息的图像信号，我们把这些信息作为 3D 文理信息导入到 GPU，并由立体着色器渲染。立体着色器提供了有关立体成分的算法，包括直接体视化，最大密度投影和其不透明变量，局部最大密度投影等。每个算法都有其独特的优点。举例来说最大密度投影允许用户查看通常代表细神经分支的弱信号，直接体视化展示在低噪声下展示大脑结构。用户还可以在 3D 视角下交互式的跟用鼠标单击的方式跟踪。

3. 结果

我们将 neuTube 1.0 和其他神经重建软件程序（Neuromantic 和 Neurostudio）进行比较，因为这两个软件虽然缺失了一些重要的功能，但仍然是最接近 SWC 框架的两个程序。我们用来自 DIADEM 数据库的 4 个 3D 图像作为测试材料，让四位用户在相同的时间限制下进行升级结构重建。和真实的应用场景相似，用户在不能分辨或修正错误时停止重建。这个实现反映了软件可视化和编辑功能的灵活程度。我们用包括分支节点和终点等关键节点的准确性来衡量重建结果的准确度。假设共有 N 个节点，其中 M 个节点和真实值匹配，这么重建错误可以用下式表达：

$$Error = \frac{T_d(F_p + F_n) + \sum_{m=1}^{m} d_m}{N}$$

如图 12 所示，其中 F_p 和 F_n 是错误的正例和错误的反例的数量，相应的 T_d 代表了匹配点对之间允许的最大距离，d 代表了第 m 个匹配点对之间的的距离。我们的错误衡量机制基于 DIADEM，但是对于评价神经重建的交互性上有两个主要的修改。一个修改是我们的关键的匹配机制是全局的，而 DIADEM 有一个确定的从根节点开始顺序，这就造成了在上游节点的权重更高。对于用户来说，上游节点的缺失和下游节点的缺失是同等重要的。因此，我们的衡量机制是顺序无关的，所有的节点有同样的权重。另一个不同是我们的衡量机制通过我们之前接受的匹配阈值 (T_d) 将拓扑结构错误和位置错误结合起来。这个阈值 T_d 和 DIADEM 中的区域阈值类似，但是我们并不赋予固定的值，而是根据根据主体或者应用来调整。通过比较各种参数，

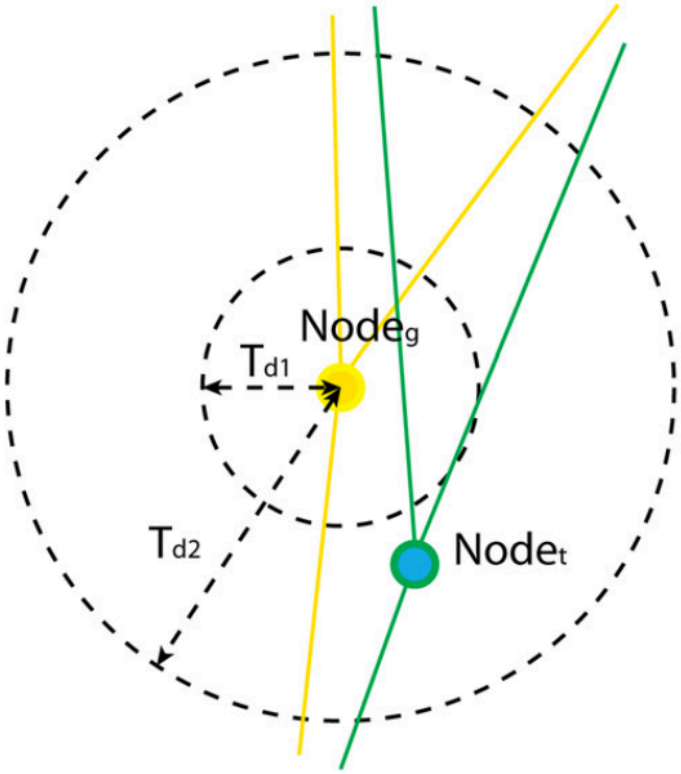

图 12　匹配点对之间的距离定义

neuTube 1.0 实现了比 Neuromantic 和 Neurostudio 更高的神经重建准确度。阈值越大 neuTube 1.0 的优势越明显，这显示了 neuTube 1.0 可以帮助用户通过比其他软件更多的关键点来获得更准确的神经结构精度。

Novel Tools and Methods

neuTube 1.0: A New Design for Efficient Neuron Reconstruction Software Based on the SWC Format[1,2,3]

Linqing Feng,[1,*] Ting Zhao,[2,*] and Jinhyun Kim[1,3]

DOI:http://dx.doi.org/10.1523/ENEURO.0049-14.2014

[1]Center for Functional Connectomics, Korea Institute of Science and Technology (KIST), Seoul, 136-791, Korea, [2]Janelia Research Campus, Howard Hughes Medical Institute, Ashburn, Virginia, 20147, and [3]Neuroscience Program, University of Science and Technology, Daejeon, 305-350, Korea

Abstract

Brain circuit mapping requires digital reconstruction of neuronal morphologies in complicated networks. Despite recent advances in automatic algorithms, reconstruction of neuronal structures is still a bottleneck in circuit mapping due to a lack of appropriate software for both efficient reconstruction and user-friendly editing. Here we present a new software design based on the SWC format, a standardized neuromorphometric format that has been widely used for analyzing neuronal morphologies or sharing neuron reconstructions via online archives such as NeuroMorpho.org. We have also implemented the design in our open-source software called neuTube 1.0. As specified by the design, the software is equipped with parallel 2D and 3D visualization and intuitive neuron tracing/editing functions, allowing the user to efficiently reconstruct neurons from fluorescence image data and edit standard neuron structure files produced by any other reconstruction software. We show the advantages of neuTube 1.0 by comparing it to two other software tools, namely Neuromantic and Neurostudio. The software is available for free at http://www.neutracing.com, which also hosts complete software documentation and video tutorials.

Key words: bioimage informatics; neuron tracing; SWC; user-friendly software

Significance Statement

Compared to other existing tools, the novel software we present has some unique features such as comprehensive editing functions and the combination of seed-based tracing and path searching algorithms, as well as their availability in parallel 2D and 3D visualization. These features allow the user to reconstruct neuronal morphology efficiently in a comfortable "What You See Is What You Get" (WYSIWYG) way.

Introduction

Digital reconstruction, or tracing, of neuron morphologies from light microscope images is an important step in the mapping of brain circuits. In this task, the input is images and the output is usually a tree structure, which can be

Received October 30, 2014; accepted December 23, 2014; First published January 02, 2015.

[1]Authors report no conflict of interest.

[2]Author Contributions: L.F., T.Z., and J.K. designed research; L.F. and T.Z. performed research; L.F. and T.Z. analyzed data; L.F., T.Z., and J.K. wrote the paper.

[3]This work was supported by the KIST Institutional Program (Project No. 2E24210) and WCI Program (NRF Grant Number: WCI 2009-003). T.Z. is supported by Howard Hughes Medical Institute.

*L.F. and T.Z. contributed equally to this work.

The authors would like to thank Camden Markham for proof reading the manuscript and thank Osung Kwon and Bokyoung Lee for testing the software.

Correspondence should be addressed to either Dr. Ting Zhao, Janelia Research Campus, Howard Hughes Medical Institute, 19700 Helix Drive, Ashburn, VA 20147, E-mail: zhaot@janelia.hhmi.org; or Dr. Jinhyun Kim, Center for Functional Connectomics, Korea Institute of Science and Technology (KIST), 39-1 Hawolgokdong, Seoul, 136-791, Korea, E-mail: kimj@kist.re.kr.

described by the SWC file format (Cannon et al., 1998). Although numerous neuron reconstruction software tools have been developed for producing SWC files (Meijering, 2010), none of them have taken full advantage of the SWC format to optimize the user interface for efficient and accurate reconstruction. An optimal user interface means that the user can interact with the software with minimal cognitive load, which requires data visualization to be clear and operations to be straightforward. In other words, with the visual information provided by the software, the user should be able to quickly figure out the underlying SWC model, how the model can be manipulated, and the results of manipulations. With these criteria in mind, we may identify disadvantages of many tracing software applications. For example, FARSIGHT (Luisi et al., 2011), which focuses on semi-automated reconstruction of neurons, does not provide intuitive low-level editing options to correct subtle errors. Simple Neurite Tracer, a popular plugin of Fiji (Longair et al., 2011), also lacks editing functions. Neurolucida, a mainstream commercial software tool, allows complete manual reconstruction of a neuron structure. However, it does not support neuron reconstruction in a 3D visualization window, despite the fact that 3D interaction has been demonstrated to improve both the speed and accuracy of the reconstruction procedure (Long et al., 2012). Some other popular software tools, such as Neuromantic (Myatt et al., 2012) and Neurostudio (Wearne et al., 2005), similarly lack advanced 3D editing functions. Vaa3D (Peng et al., 2010) provides innovative interactive neuron tracing functions in 3D, but these functions are not available in 2D to resolve dense or faint structures.

Here, we propose a new and comprehensive software design based on the SWC format as a solution to the diverse limitations of current tracing software. Based on this design, or, as we call it, the SWC framework, we have converted our previously reported software, neuTube (Kim et al., 2012), into a novel tool that enables efficient reconstruction by combining robust automatic tracing algorithms and versatile user-friendly editing functions in both 2D and 3D. This paper formally presents the redesigned software, neuTube 1.0, not only as a major upgrade of the previous release, but also as the first software to implement the SWC framework.

Materials and Methods

The SWC framework

The overall layout of the SWC framework is shown in Figure 1. Software with this architecture takes a raw image or an SWC file as input and outputs a neuron structure satisfying the user. The design of the SWC framework follows the principle of "What You See Is What You Get" (WYSIWYG) (Peng et al., 2011), i.e., what the user is editing is explicitly visualized and no third-party viewer is needed to check the results. Therefore, the SWC framework consists of the following features: clear visualization of SWC structures, clear visualization of source images as reference data, explicit definition of operation units, and intuitive map from user inputs to editing operations. Except image visualization, these features are designed based on the SWC format, which describes a simple directed tree model, called the SWC model. Here, we described in detail how to construct operations on the SWC model by first defining the model in an abstract way.

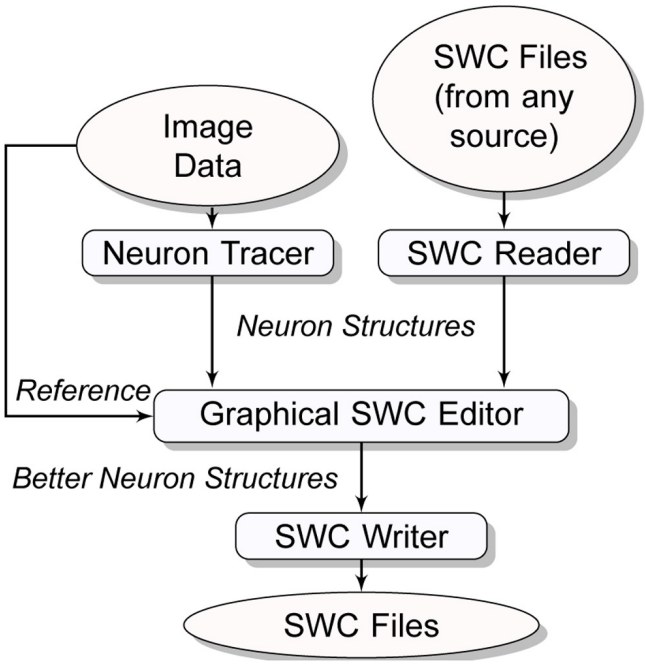

Figure 1 Workflow of reconstructing or editing a neuron structure in the SWC framework, which defines GUI software that takes either a raw image or an SWC file as input and generates an acceptable neuron structure through user interactions. The user can save the neuron structure into standard SWC files during or after reconstruction.

Abstract definition of the SWC model

From a mathematical point of view, the SWC model can be defined as a set of nodes $\{\mathbf{n}_i = (x_i, y_i, z_i, r_i, \mathbf{n}_j) | i = 1, ..., N, j = 0, ..., N, i \neq j, x_i, y_i, z_i, r_i \in R\}$, where each node $\mathbf{n}_i$ is a sphere with the center (x_i, y_i, z_i) and the radius r_i. $\mathbf{n}_0$ is an empty node for defining the root of a neuron structure, and $\mathbf{n}_j$ is called the parent of $\mathbf{n}_i$. An upstream path from $\mathbf{n}_i$ to $\mathbf{n}_j$ is an array of node $(\mathbf{n}_{k_1}, ...\mathbf{n}_{k_n})$, where $\mathbf{n}_{k_{i+1}}$ is the parent of $\mathbf{n}_{k_i}$, $k_1 = i$, $k_n = j$. To form a valid tree structure of a neuron, no loop is allowed, i.e., there is at most one upstream path from one node to another. In this model, the basic structural unit is a node, which defines how we should design visualization and interactions.

SWC operation

Assuming S_1 and S_2 are two sets of nodes, the operation of a neuron structure is defined as

$$f(S_1) = S_2$$

DOI:http://dx.doi.org/10.1523/ENEURO.0049-14.2014

For example, $f(\{\mathbf{n}_1, ..., \mathbf{n}_n\}) = \phi$, where ϕ denotes the empty set, defines a removal operation. However, some operations may result in a new node set that forms an invalid neuron structure. How to construct a valid operation depends on the data structure describing the model. In our framework, we used a redundant tuple to store a node, which is $\mathbf{n} = (G(\mathbf{n}), P(\mathbf{n}), C(\mathbf{n}), S(\mathbf{n}))$, where $G(\mathbf{n}) = (x(\mathbf{n}), y(\mathbf{n}), z(\mathbf{n}), r(\mathbf{n}))$ defines that the node is located at $(x(\mathbf{n}), y(\mathbf{n}), z(\mathbf{n}))$ with radius $r(\mathbf{n})$, $P(\mathbf{n})$ is the parent node of $\mathbf{n}$, $C(\mathbf{n})$ is the first child of $\mathbf{n}$ and $S(\mathbf{n})$ is the next sibling of $\mathbf{n}$. A sibling of $\mathbf{n}$ shares the same parent with $\mathbf{n}$, i.e. $P(\mathbf{n}) = P(S(\mathbf{n}))$. The redundancy is designed to improve computational efficiency of visiting a node. For example, to query a child of a node, the program needs only to check its first child and traverse other children through the sibling link, while in a non-redundant representation where each node is only linked to its parent, the program may need to check every node in the tree.

Editing a node $\mathbf{n}$ is defined as changing the value of the corresponding tuple. We call any change on $G(\mathbf{n})$ a geometrical operation and any change on $P(\mathbf{n})$, $C(\mathbf{n})$or $S(\mathbf{n})$ a structural operation. While a geometrical operation is straightforward, a structural operation may cause invalid neuron structures. For example, changing $P(\mathbf{n})$ alone may break the rule that $P(C(\mathbf{n})) = \mathbf{n}$ and $P(\mathbf{n}) = P(S(\mathbf{n}))$. To avoid this problem, we construct SWC operations at three levels in terms of operation complexity. The first level consists of three elementary operations linking a node $\mathbf{n}$ to another node $\mathbf{n}'$, as defined as follows

$$f_p(\{\mathbf{n}\}|\mathbf{n}') = f_p(\{(G(\mathbf{n}), P(\mathbf{n}), C(\mathbf{n}), S(\mathbf{n}))\}|\mathbf{n}') = \{(G(\mathbf{n}), \mathbf{n}', C(\mathbf{n}), S(\mathbf{n}))\}$$

$$f_c(\{\mathbf{n}\}|\mathbf{n}') = f_c(\{(G(\mathbf{n}), P(\mathbf{n}), C(\mathbf{n}), S(\mathbf{n}))\}|\mathbf{n}') = \{(G(\mathbf{n}), P(\mathbf{n}), \mathbf{n}', S(\mathbf{n}))\}$$

$$f_s(\{\mathbf{n}\}|\mathbf{n}') = f_s(\{(G(\mathbf{n}), P(\mathbf{n}), C(\mathbf{n}), S(\mathbf{n}))\}|\mathbf{n}') = \{(G(\mathbf{n}), P(\mathbf{n}), C(\mathbf{n}), \mathbf{n}')\}$$

At this level, structure validity is not guaranteed.

The second level consists of simple valid operations. Assuming $F_{p_0}(\mathbf{n})$ is the operation of setting the parent of $\mathbf{n}$ to $\mathbf{n}_0$ (the empty node), if $C(P(\mathbf{n})) = \mathbf{n}$, i.e. $\mathbf{n}$ is the first child of its parent, then

$$F_{p_0}(\mathbf{n}) = \begin{cases} f_s(\{n\}|\mathbf{n}_0) \circ f_p(\{\mathbf{n}\}|\mathbf{n}_0) \circ f_c(\{P(\mathbf{n})\}|S(\mathbf{n})), & C(P(\mathbf{n})) = n \\ f_s(\{n\}|\mathbf{n}_0) \circ f_p(\{\mathbf{n}\}|\mathbf{n}_0) \circ f_s(\{S^{-1}(\mathbf{n})\}|S(\mathbf{n})), & \textit{Otherwise} \end{cases}$$

where $f \circ g$ denotes a composite operation and $S(S^{-1}(\mathbf{n})) = \mathbf{n}$. To define an operation on a single node more explicitly, $F_{p_0}(\mathbf{n})$ is defined as a function of a node instead of a node set without adding any ambiguity.

The operation of setting a parent is

$$F_p(\mathbf{n}|\mathbf{n}') = f_c(\{C(\mathbf{n}')\}|\mathbf{n}) \circ f_s(\{\mathbf{n}\}|C(\mathbf{n}')) \circ f_p(\{\mathbf{n}\}|\mathbf{n}') \circ F_{p_0}(\mathbf{n})$$

This operation also sets $\mathbf{n}$ as the first child of $\mathbf{n}'$. In principle, this operation is sufficient for building all other operations. But in practice, it is useful to define one more operation, for setting a sibling:

$$F_s(\mathbf{n}|\mathbf{n}') = f_s(\{\mathbf{n}\}|\mathbf{n}') \circ f_s(\{\mathbf{n}'\}|S(\mathbf{n})) \circ f_p(\{\mathbf{n}'\}|P(\mathbf{n})) \circ F_{p_0}(\mathbf{n}')$$

The third level is a set of composition operations, which include any operation composed of the operations from the second level. At this level, we categorize the operations into two types, morphology-dependent and morphology-independent. An operation is morphology-dependent if the result of the operation depends on the positions or sizes of the nodes; otherwise it is morphology-independent.

Decomposing an operation into elementary operations helps guarantee the validity of neuron structure manipulation, and more importantly, helps implement the undo/redo functionality on arbitrary operations. An undo operation requires inverting the corresponding operator, which can be complicated because of the consistency requirement. For example, the inverse operation of deleting multiple nodes would require recovery of all the neighbors of the nodes. Direct inference of such an inverse operation not only takes significant effort, but also leads to errors that can be difficult to track. After decomposing an operation into a sequence of elementary operations, we can construct the undo operation easily by reversing the sequence.

User interaction

The fundamental function of tracing software is changing neuron morphology with user inputs, which are usually composed of mouse clicks and key inputs. Since we defined an operation as the mapping of one set of nodes to another set, user interaction starts with node selection, which requires two components: SWC visualization and user input response. High-quality visualization of a neuron might be the most important feature of successful neuron editing. The ability to view the structures clearly greatly reduces examination time needed to identify errors. It is also necessary to provide both 2D and 3D views because each provides unique advantages. For example, a 3D view is well suited for displaying a global structure and a 2D view provides precise inference of dense local structures.

The most intuitive way to select a node is to move the mouse cursor to the node and then click. This requires mapping the screen cursor coordinates into the 3D SWC space. Multiple selections should also be supported to specify a set of nodes as the input of an operation. After selection, the user can trigger an operation with some input. So the operation becomes

$$f(S_1|\Theta) = S_2$$

where Θ is the set of parameters supplied from user input. For example, $f(\{\mathbf{n}\}|(x, y, z)) = \{\mathbf{n}, F_p(((x, y, z, r(\mathbf{n})), \mathbf{n}_0, \mathbf{n}_0, \mathbf{n}_0)|\mathbf{n})\}$ defines an operation of extending a branch from $\mathbf{n}$ to a node at (x, y, z).

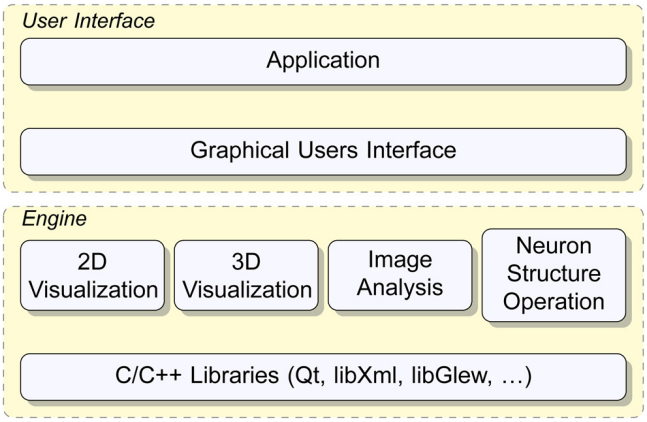

Figure 2 neuTube 1.0 is a GUI application built upon four major modules: 2D visualization, 3D visualization, image analysis, and neuron structure operation.

Create SWC nodes from image signal

For any standalone neuron-tracing software, it is essential to allow reconstructing neuron structure from raw image signals. In the SWC framework, this function can be formulated as

$$g(S_1 \mid \Theta, I) = S_2$$

where I is the image signal. Note that this actually defines a superfamily of SWC operations. The function is the same as an SWC operation if it is independent of I. An example of an image-dependent operation is shortest path creation, such as the one used by Simple Neurite Tracer (Longair et al., 2011), where $S_1 = \{\mathbf{n}_i, \mathbf{n}_j\}$ defines the source and target node and $S_2 = \{\mathbf{n}_i, \mathbf{n}'_1, ..., \mathbf{n}'_k, \mathbf{n}_j\}$ forms the resampled shortest geodesic path from $\mathbf{n}_i$ to $\mathbf{n}_j$. The radii of $\mathbf{n}'_1, ..., \mathbf{n}'_k$, which are denoted as $r(\mathbf{n}'_1), ..., r(\mathbf{n}'_k)$ in the node definition, can be estimated automatically or linearly interpolated, depending on how the operation is defined.

Software implementation

Architecture

Based on the SWC framework, we have built neuTube 1.0 as a GUI application upon four core modules: 2D visualization, 3D visualization, image analysis, and neuron structure operation (Fig. 2).

2D visualization. The 2D visualization module provides functions of displaying a 3D image and neuron structures slice by slice, as well as functions allowing the user to interact with the 2D display. This module facilitates close examination and precise editing. For example, driven by this module, the user can zoom into a region of interest to view details, locate tracing point precisely, or apply fine-tuning on a neuron structure. As the purpose of 2D visualization is to show the matching quality between the reconstruction and the data rather than a realistic neuron structure, we only used two geometrical primitives, lines and circles, to represent the morphology of a neuron (Fig. 3*B*). The 2D visualization is useful for showing the exact planar position of a node, yet not suitable for showing the position perpendicular to the plane. We used two strategies to address the issue. First, each node of the neuron is displayed as a circle when the plane cuts through the node. The circle is as large as the corresponding cross section of the node, informing the user by its size how far the node is from the plane. Second, we used colors to distinguish whether a node is centered on the current plane (on-plane) or not (off-plane): the node is shown with a fully saturated and opaque color when it is on-plane; otherwise the node color is semi-transparent and less saturated (Fig. 3*B*). The coloring options were tuned manually according to the user feedback and then used as immutable parameters of the software. To allow the user view the global structure of a neuron under reconstruction, we also project the whole skeleton onto the slice view, but with a thin and semi-transparent mode to minimize its interference with in-focus structures.

3D visualization. The 3D visualization module is designed to provide real-time rendering of 3D images and neuron structures. The user can perform tracing (Fig. 3*D*) and editing (Fig. 3*F*) in the 3D visualization window directly, in which any change in the neuron structure will be reflected in the 2D visualization window simultaneously, and vice versa.

Image analysis

This module offers automatic tracing of a neuron or a neuron branch to allow the user to obtain neuron structures with minimal interaction. For example, to select a branch, the user only needs to specify a point on the branch with one click. The algorithm and design were described in Zhao et al., (2011) and Kim et al., (2012). In this paper, one major improvement over the previously reported version (Kim et al., 2012) is the replacement of the cylindrical model by the tree model defined in the SWC framework. In addition, we have implemented a point-to-point tracing function based on the shortest path method used previously in automated reconstruction (Zhao et al., 2011). This function is similar to semi-automated tracing in the Simple Neurite Tracer and Vaa3d, but we have made it available in both 2D and 3D views by following the SWC framework.

Neuron structure manipulation

The module of neuron structure manipulation provides functions for the arbitrary editing of neuron nodes (Fig. 3*C*,*F*). The user can change the geometry and topology of a neuron structure with intuitive mouse clicks or keyboard shortcuts. This module supports operations described in the SWC framework and separates them into different levels.

We have also built high-level operations from elementary ones to reduce the labor required for structural operations. These operations are as follows.

Interpolate. In many cases, a neuron branch or a segment thereof is smooth enough to be represented by piecewise linear structures. Interpolation takes advantage of this property and allows the user to quickly correct geometrical attributes of multiple nodes (Fig. 4*B*) by specifying the nodes that need interpolation (Fig. 4*A*).

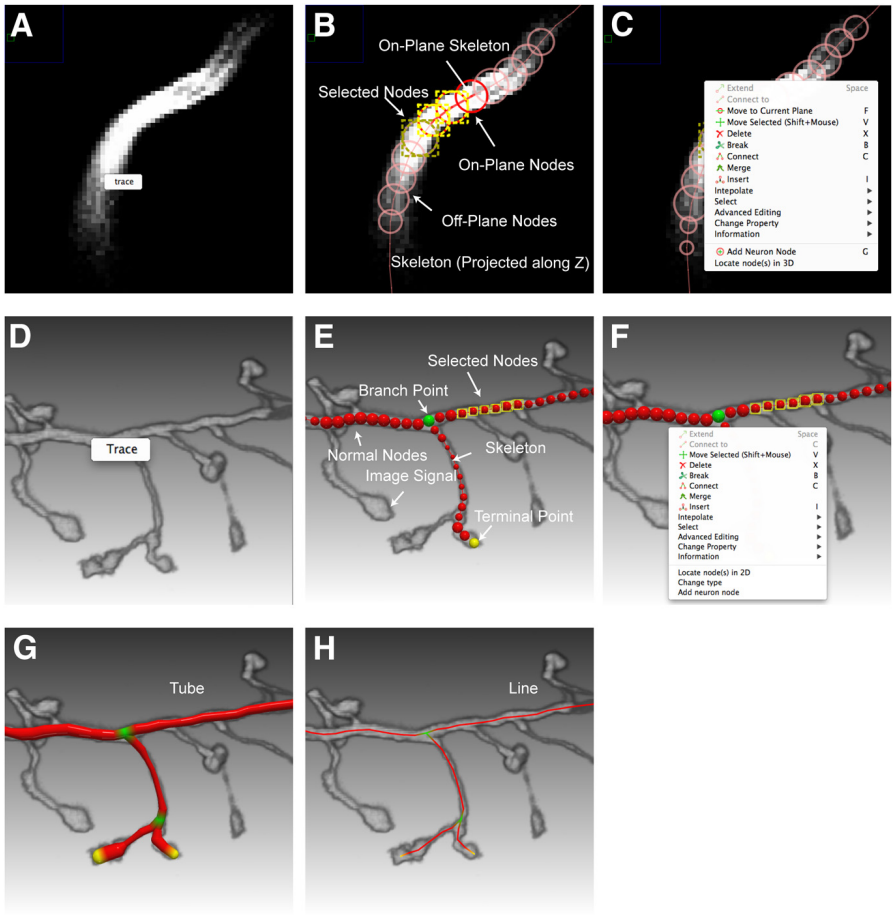

Figure 3 Tracing and editing interface of neuTube 1.0. ***A***, Interactive tracing in 2D view. ***B***, 2D view of SWC nodes. On-plane and off-plane nodes are distinguished by color saturation and transparency. A node with a yellow bounding box indicates that it is selected. ***C***, The context menu for editing in the 2D view, which can be triggered by a right mouse click. ***D***, Interactive tracing in 3D view. ***E***, 3D visualization of the tracing results. Branch nodes and terminal nodes are shown in green and yellow colors, respectively. Selected nodes are shown with their bounding boxes. ***F***, The context menu for editing in the 3D view. ***G***, A neuron shown as connected tubes. ***H***, A neuron shown as lines.

Set branch point. It often happens that a branch point is missed when the end of a segment is close to the interior of another segment. A completely manual editing operation would consist of selecting two nodes and joining them together. The operation of setting branch point simplifies this work by connecting the selected node (Fig. 4*C*) to the latest node in isolated branches when the connection creates a branch point (Fig. 4*D*).

Reset branch point. This operation provides another way to correct a branch point. In this operation, the user selects a node (Fig. 4*E*) and the program will try to move the neighboring branching structure to the selected node (Fig. 4*F*). The program automatically determines which branch to move based on their angles.

Connect multiple nodes. Connecting two nodes is one of the most basic operations, yet one that requires multiple steps, including selecting the nodes and triggering the connection command. When there are more and more nodes to connect, the number of human interactions increases proportionally. Therefore, neuTube 1.0 provides an operation for automatically connecting multiple nodes (Fig. 4*G*) by their edges in the minimal spanning tree of their pairwise distance graph (Fig. 4*H*).

Remove turn. A turn is defined as three sequentially connected nodes that form an acute angle. The node in the middle is the turning point and the other two nodes are the flank nodes. The operation of removing a turn is to set the turning point (Fig. 4*I*) as the interpolation of the flank nodes (Fig. 4*J*). When the turning point is a branch point, the flank nodes are its two neighbors that form the sharpest turn.

Resolve crossover. Crossover is a common tracing error in tracing when two branches are close at a certain point (Fig. 4*K*). Correcting a crossover requires several operations of connecting and breaking nodes. Therefore, we added an operation of automatic inference of crossover (Fig. 4*L*) to make the editing easier.

Implementation

The software is written in the C and C++ programming languages with several third-party libraries. The main third-party library is the Qt library (http://qt-project.org), which provides a cross-platform framework for GUI de-

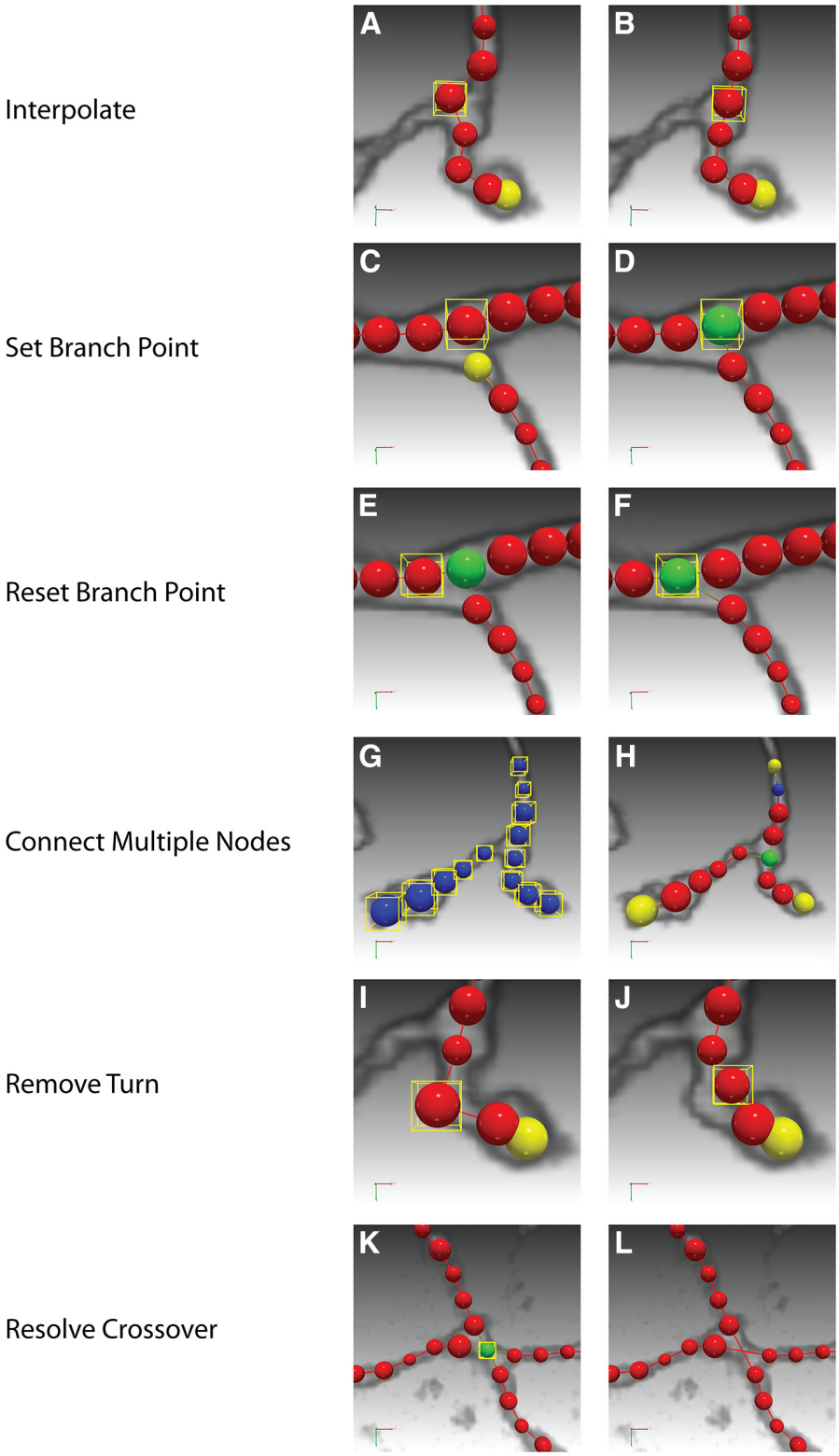

Figure 4 Examples of high-level operations of neuTube 1.0. To illustrate the operation, we visualize the nodes of a neuron in different colors according to the topology: blue for root nodes, green for branch nodes, yellow for leaf nodes, and red for other nodes. Selected nodes are highlighted by a yellow bounding box. For the corresponding operation as named in each row, the panels on the left (***A***, ***C***, ***E***, ***G***, ***I***, ***K***) show the selected nodes to operate and the panels on the right (***B***, ***D***, ***F***, ***H***, ***J***, ***L***) show the result of operation.

velopment. The 3D visualization module is built upon OpenGL 2.0 (http://www.opengl.org) and its shading language, GLSL (http://www.opengl.org/documentation/glsl). We developed a fast engine for rendering neuron

Table 1: Feature comparison of neuTube 1.0 with Neuromantic and Neurostudio

Software	Undo/Redo	2D editing	3D editing	3D image interaction	2D neuron visualization	3D visualization
neuTube 1.0	Unlimited	Yes	Yes	Yes	Slice-by-slice	Volume and neuron structure
Neuromantic	1 step	Yes	Limited[b]	No	Slice-by-slice	Neuron structure
Neurostudio	1 step	Limited[a]	Limited[b]	No	Projection	Volume and neuron structure

[a]Cannot change node size.
[b]No topological operation.

structures by writing highly efficient shaders for two geometric primitives, sphere and conical frustum. The vertex shader finds bounding boxes of the geometric primitives on the screen, and then the fragment shader calculates ray-quadric intersections for each pixel inside the rasterized bounding box. All of our geometric primitives have adjustable opacity options and can be visualized in the order needed to generate a reasonable semi-transparent scene. For realistic rendering of complicated semi-transparent scenes, we have also implemented dual depth peeling and weighted average blending (Bavoil and Myers, 2008), which are two commonly used order-independent transparency methods. Since the two methods do not require special hardware features of high-end graphical cards, they provide neuTube 1.0 with the ability to render complicated scenes realistically without comprising the software portability. The user can switch from one method to the other in runtime to determine which one is better for the current scene.

To show an image signal in 3D, a volume, which contains the original image of the neurons to reconstruct, is uploaded to GPU as 3D texture and is rendered by a volume shader. The volume shader provides several volume composite methods, including direct volume rendering (DVR), maximum intensity projection (MIP) and its opaque variant, local maximum intensity projection (LMIP) (Sato et al., 1998). Each method has its own advantages. For example, MIP opaque allows the user to see weak signals that are typical of thin neural branches, LMIP is an extended version of MIP that can clearly depict spatial interrelations of neural branches, and DVR illustrates bright structures with low noise (Fishman et al., 2006). Users can also trace interactively in a 3D view by providing a seed point for tracing with a single mouse click, which represents a ray passing through the 3D volume. The seed point used for tracing is determined as the first location with maximum intensity along the ray.

Results

We compared our neuTube 1.0 to other neuron reconstruction software programs, namely, Neuromantic and Neurostudio. These two programs were chosen because their designs are close to the SWC framework, although they lack several important features available in the framework (Table 1). Four 3D images from the DIADEM datasets (Brown et al., 2011) were traced using all three software programs by four users given the same time constraint. Similar to the situation of real applications, the user could decide to stop tracing whenever he/she could not identify or fix an error. This reflects how well the software visualizes the reconstruction along with the data and the flexibility of the editing functions. The accuracy of tracing was measured by how well the critical points, including branching points and termini, were reconstructed compared to ground truth reconstructions. We extracted branching and terminal points as two point sets from each tracing result and matched them to the ground truth by solving the linear assignment problem (LAP) using the Jonker-Volgenant Algorithm (Jonker and Volgenant, 1987). Assuming there are a total of N points with M of them matched to the ground truth, the reconstruction error is calculated as:

$$\text{Error} = \frac{T_d(F_p + F_n) + \sum_{m=1}^{M} d_m}{N}$$

where F_p and F_n are the number of false positives and the number of false negatives, respectively, T_d is the maximal distance allowed between two matched points (Fig. 5*A*), and d_m is distance between the mth matched pair of points. In this calculation, the term $T_d(F_p + F_n)$ is the cost of missing critical points and $\sum_{m=1}^{M} d_m$ is the cost of position offset.

Our error metric is designed on the basis of the DIADEM metric (Gillette et al., 2011), but with two major modifications for better evaluation of interactive neuron reconstruction. One modification is that our metric matches critical points globally, while the DIADEM metric matches critical points in a certain order, which starts from the root position and may give an upstream node more importance. For user editing, missing an upstream node and missing a downstream one usually mean the same type of error. Our matching method is order-independent and treats these nodes equally. The other different feature of our metric is the combination of topological errors and position errors, with the introduction of the matching threshold (T_d), as the weight of mismatches. The threshold T_d is similar to the threshold region of the DIADEM metric, but we do not assign it a fixed value, which is often subjective or application dependent. Instead, we define the error metric as a function of T_d.

By comparing scores across a wide range of threshold values, we showed that neuTube 1.0 achieved consistently better reconstruction accuracy than Neuromantic and Neurostudio (Fig. 5*B*). The advantage of neuTube 1.0 is more significant when the threshold is larger, indicating that neuTube 1.0 helps the user obtain more accurate neuron structures by identifying more critical points than the other two software tools.

Application example

We have used neuTube 1.0 to map the fine-scale synaptic connectivity between hippocampal regions (CA3–CA1) of

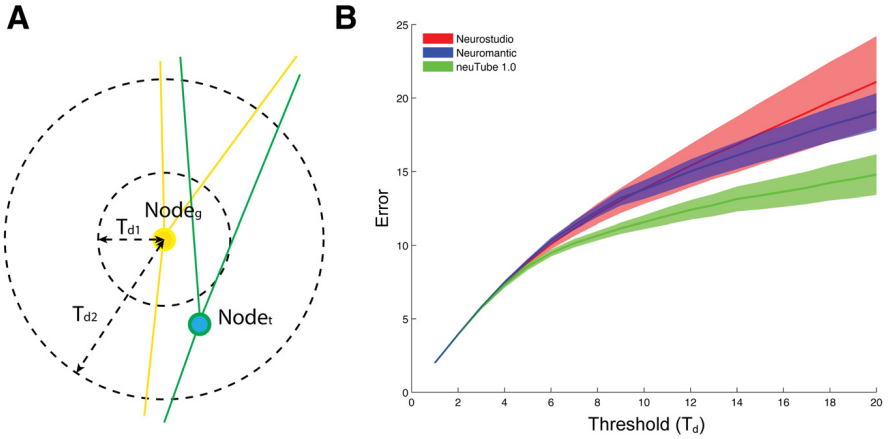

Figure 5 neuTube 1.0 helps produce significantly more accurate neuron structures than Neuromantic and Neurostudio do. ***A***, Node$_g$ is a critical point from ground truth neuron and Node$_t$ is a critical point from tracing result. These two points can be matched when $T_d = T_{d2}$ because the distance between them is less than T_{d2}. They cannot be matched when $T_d = T_{d1}$. ***B***, The solid curves show average errors measuring the discrepancy between the critical point sets from user reconstruction and ground truth under different distance thresholds. The surrounding envelopes are the 95% confidence intervals. The error curve of neuTube 1.0 (green) is consistently lower than the other two, with $p < 0.01$ (t test) when $T_d \geq 6$ (compared to Neurostudio) or $T_d \geq 5$ (compared to Neuromantic).

the mouse brain (Druckmann et al., 2014). To analyze the spatial synaptic connectivity pattern, mammalian GFP reconstitution across synaptic partners (mGRASP) (Kim et al., 2012) was used to label the synapses, and red fluorescence protein (i.e., dTomato) was used to label the postsynaptic dendrites (Fig. 6*A*). neuTube 1.0 was used to reconstruct 3D structures of postsynaptic neurons (Fig. 6*B*). In our application, we detected the mGRASP-labeled synapses using our mGRASP detection package (Feng et al., 2012) (Fig. 6*C*), and then assigned each synapse to a reconstructed neuron by calculating its intensity-weighted distances to all nearby neurons (Fig. 6*D*). To make the mapping more accurate in the step of synapse assignment, we need to reconstruct not only the selected neurons but also the remaining dendrite branches or background neurons (Fig. 6*B*) because the distance to the nearest selected neuronal branch alone can mis-assign synapse puncta (Feng et al., 2014). A practical solution to this is to reconstruct all dendrite branches from the 3D image first and then edit the target neurons, which must be reconstructed correctly. neuTube turned out to be the right tool for this problem because the SWC framework specifies that the software can start the reconstruction from any SWC file. With the help of neuTube 1.0, we have built a fine-scale mapping of the hippocampal CA3–CA1 circuit and, with further statistical analysis, revealed spa-

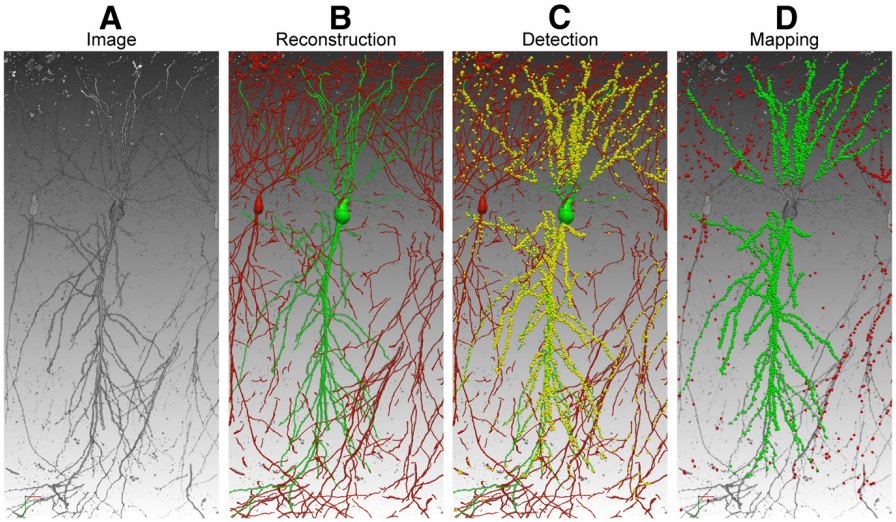

Figure 6 Mapping brain connectivity with neuTube 1.0. ***A***, The original 3D confocal image contains postsynaptic neurons. ***B***, The target neuron (green) was traced semi-automatically. The red branches belong to background neurons. ***C***, mGRASP-labeled synapses (yellow) were detected automatically, with sizes enlarged for better visualization. ***D***, The synapses were mapped to the target neuron (green) and background neurons (red).

tially structure and clustered synaptic connectivity patterns between CA3 and CA1 (Druckmann et al., 2014).

Discussion

We designed the SWC framework and implemented it in neuTube 1.0 (www.neutracing.com) to improve the efficiency of reconstructing neuron structures accurately. Guided by the framework, the software combines 2D/3D visualization, semi-automated tracing algorithms, and flexible editing options to simplify the task of neuron reconstruction. The SWC framework is not designed to solve the problem of high-throughput neuron tracing, which is different and more challenging. As revealed by the recent DIADEM competition (Liu, 2011), a reliable and generally applicable high-throughput neuron-tracing tool may not be available in the near future. While waiting for the ideal solution, neuroscientists will benefit from better neuron reconstruction tools. Therefore, the goal of the SWC framework is to provide a general architecture, which can adopt state-of-the-art image analysis methods and modern software techniques, for building better interactive neuron reconstruction tools.

Our framework has one limitation, which is that it can only produce neuron structures defined in the SWC format. However, this is usually not a significant concern because the SWC model suffices for most purposes, such as comparing neuron shapes, performing Sholl analysis, uploading neuron structures to NeuroMorpho.org (Ascoli et al., 2007), and simulating neuron activities. Many researchers prefer the SWC format rather than more complicated models because it helps to avoid overfitting to imaging artifacts: the resolution of optical microscopy is usually not high enough to reveal fine details. Even when a structure more complex than the SWC model is needed, reconstructing the neurons in the SWC model is still useful as an initial input for later shape refinement (Evers et al., 2005).

Our experiment showed that the results from neuTube 1.0 were generally better than those from Neurostudio (Myatt et al., 2012) and Neuromantic (Wearne et al., 2005), but it is still worth noting the strengths of these software programs. Neuromanic allows multi-tile tracing to reconstruct neurons from more than one field of view. This is particularly useful for reconstructing a large neuron that requires horizontal stage movement to cover all branches. Neurostudio offers only limited free editing functions, but its ability to trace multiple branches from one seed point is a very useful feature to reduce labor, and its intrinsic radius estimation based on rayburst sampling (Rodriguez et al., 2006) can be implemented in any other software to refine the neuron structure.

As the functions of multi-branch tracing and rayburst radius estimation naturally fit in the SWC framework, we plan to include them in a future upgrade of neuTube 1.0. Additionally, there are ongoing efforts to extend the software to broader applications, including tracing neurons in bright-field images and analyzing neuron morphologies, such as identifying neuron types from electron microscope reconstructions (Zhao and Plaza, 2014).

Because a user can import results from other software into neuTube 1.0 to do further editing, neuTube 1.0 is also a complementary tool to other automated or interactive neuron tracing tools. For instance, the Vaa3d software has added neuTube 1.0 as a plug-in in recent releases (vaa3d.org). Also, other developers can improve their own software by adopting the SWC framework. To facilitate any such adoption, we have made the source code of neuTube 1.0 available at https://github.com/janelia-flyem/NeuTu.

References

Ascoli GA, Donohue DE, Halavi M (2007) NeuroMorpho.org: a central resource for neuronal morphologies. J Neurosci 27:9247–9251. CrossRef Medline

Bavoil L, Myers K (2008) Order independent transparency with dual depth peeling. NVIDIA OpenGL SDK:1–12.

Brown KM, Barrionuevo G, Canty AJ, De Paola V, Hirsch JA, Jefferis GSXE, Lu J, Snippe M, Sugihara I, Ascoli GA (2011) The DIADEM data sets: representative light microscopy images of neuronal morphology to advance automation of digital reconstructions. Neuroinformatics 9:143–157. CrossRef Medline

Cannon RC, Turner DA, Pyapali GK, Wheal HV (1998) An on-line archive of reconstructed hippocampal neurons. J Neurosci Methods 84:49–54. Medline

Druckmann S, Feng L, Lee B, Yook C, Zhao T, Magee JC, Kim J (2014) Structured synaptic connectivity between hippocampal regions. Neuron 81:629–640. CrossRef Medline

Evers JF, Schmitt S, Sibila M, Duch C (2005) Progress in functional neuroanatomy: precise automatic geometric reconstruction of neuronal morphology from confocal image stacks. J Neurophysiol 93:2331–2342.CrossRef

Feng L, Kwon O, Lee B, Oh WC, Kim J (2014) Using mammalian GFP reconstitution across synaptic partners (mGRASP) to map synaptic connectivity in the mouse brain. Nat Protoc 9:2425–2437. CrossRef Medline

Feng L, Zhao T, Kim J (2012) Improved synapse detection for mGRASP-assisted brain connectivity mapping. Bioinformatics 28: i25–i31. CrossRef Medline

Fishman EK, Ney DR, Heath DG, Corl FM, Horton KM, Johnson PT (2006) Volume rendering versus maximum intensity projection in CT angiography: what works best, when, and Why1. Radiographics 26:905–922. CrossRef Medline

Gillette TA, Brown KM, Ascoli GA (2011) The DIADEM metric: comparing multiple reconstructions of the same neuron. Neuroinformatics 9:233–245. CrossRef Medline

Jonker R, Volgenant A (1987) A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing 38:325–340.CrossRef

Kim J, Zhao T, Petralia RS, Yu Y, Peng H, Myers E, Magee JC (2012) mGRASP enables mapping mammalian synaptic connectivity with light microscopy. Nat Methods 9:96–102. CrossRef Medline

Liu Y (2011) The DIADEM and beyond. Neuroinformatics 9:99–102. CrossRef Medline

Long F, Zhou J, Peng H (2012) Visualization and analysis of 3D microscopic images. PLoS Comput Biol 8:e1002519.CrossRef Medline

Longair MH, Baker DA, Armstrong JD (2011) Simple neurite tracer: open source software for reconstruction, visualization and analysis of neuronal processes. Bioinformatics 27:2453–2454. CrossRef Medline

Luisi J, Narayanaswamy A, Galbreath Z, Roysam B (2011) The FARSIGHT trace editor: an open source tool for 3-D inspection and efficient pattern analysis aided editing of automated neuronal reconstructions. Neuroinformatics 9:305–315. CrossRef Medline

Meijering E (2010) Neuron tracing in perspective. Cytometry A 77: 693–704. CrossRef Medline

Myatt DR, Hadlington T, Ascoli GA, Nasuto SJ (2012) Neuromantic: from semi-manual to semi-automatic reconstruction of neuron morphology. Front Neuroinform 6:4.[PMC] [CrossRef] [Medline]

Peng H, Long F, Zhao T, Myers E (2011) Proof-editing is the bottleneck of 3D neuron reconstruction: the problem and solutions. Neuroinformatics 9:103–105. CrossRef Medline

Peng H, Ruan Z, Long F, Simpson JH, Myers EW (2010) V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. Nat Biotechnol 28:348–353. CrossRef Medline

Rodriguez A, Ehlenberger DB, Hof PR, Wearne SL (2006) Rayburst sampling, an algorithm for automated three-dimensional shape analysis from laser scanning microscopy images. Nat Protoc 1:2152–2161. CrossRef Medline

Sato Y, Shiraga N, Nakajima S, Tamura S, Kikinis R (1998) Local maximum intensity projection (LMIP): a new rendering method for vascular visualization. J Comp Assist Tomogr 22:912–917. Medline

Wearne SL, Rodriguez A, Ehlenberger DB, Rocher AB, Henderson SC, Hof PR (2005) New techniques for imaging, digitization and analysis of three-dimensional neural morphology on multiple scales. Neuroscience 136:661–680. CrossRef Medline

Zhao T, Plaza SM (2014) Automatic neuron type identification by neurite localization in the drosophila medulla). arXiv:1409.1892 [q-bio.NC]

Zhao T, Xie J, Amat F, Clack N, Ahammad P, Peng H, Long F, Myers E (2011) Automated reconstruction of neuronal morphology based on local geometrical and global structural models. Neuroinformatics 9:247–261. CrossRef Medline