浙江大学

本科生毕业设计

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
学院	计算机科学与技术学院
提交日期	2017 年 6 月 5 日

A Thesis Submitted to Zhejiang University
for the Degree of Bachelor of Engineering

TITLE High-performance Server System for Online Interactive Neuron Reconstruction

Author Heng Qu

Student ID 3130000569

Supervisor Nenggan Zheng

Major Computer Science and Technology

College College of Computer Science and Technology

Submitted Date June 5, 2017

浙江大学本科生毕业论文（设计）独创性声明

本人声明所呈交的毕业论文（设计）是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 **浙江大学** 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在文中作了明确的说明并表示谢意。

作者签名：　　日期：　　年　　月　　日

毕业论文（设计）版权使用授权书

本文作者完全了解 **浙江大学** 有权保留并向国家有关部门或机构送交本文的复印件和磁盘，允许本文被查阅和借阅。本人授权 **浙江大学** 可以将毕业论文（设计）的全部或部分内容编入有关数据库进行检索和传播，可以采用影印、缩印或扫描等复制手段保存、汇编毕业论文（设计）。

（保密的毕业论文（设计）在解密后适用本授权书）

作者签名：　　导师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

摘要

根据原始神经元图像信息进行神经元追踪与数字重建是神经科学界热门方向。神经科学家通过神经元追踪与数字重建，可以反推大脑是如何运作的，对理解智慧的产生有重要的帮助。由于神经元拓扑结构的复杂性，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，研究人员还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。另外现有的原始神经图像信息的神经元追踪和数字重建软件大多是运行在单机之上，无法满足多用户协同编辑与修改的需求，不利于结构脑图谱的交流。随着计算机性能和网络速度的提升，在线实时编辑神经网络结构成为了可能。在这样的背景下，设计并实现了一个在线多用户的神经元网络结构编辑分享平台，利用互联网便于数据共享与交流的特点，使得神经科学家可以便捷地进行异地，多用户协同编辑神经元网络结构，并能分享完成重建的结构脑图谱。实验与测试结果表明，该平台足以支撑至少数千名用户同时编辑的需求，并能在毫秒级别的时间内做出相应，达到了实时操作的要求，为神经科学家在神经元协同编辑与成果分享提供了便利。

关键词 生物图像信息 神经元重建 实时编辑平台

Abstract

Neuron tracing and digital reconstruction from original neural image information is a hot direction of neural science. By the reconstruction of brain connectome, neuroscientists can speculate how the brain works which is helpful to understand the intelligence. Because of the complexity of the topological structure of neurons, and in some details on the results of automation reconstruction researchers still need to manually correct and modify the results of digital reconstruction, in order to ensure the accuracy of the digital reconstruction. In addition, researchers need to edit the result of the digital reconstruction such as add or delete some branches, etc. Existing neurons tracking for original neural image and digital reconstruction softwares are mostly run on stand-alone, unable to meet the requirements of multi-user collaborative editing and modification, but also not conducive to the exchange of brain connectom. Improvements in computer performance and network speed make it possible for online real-time editing of neural network structures. In such a background, We have designed and implemented a neural network structure of online multi-user edit sharing platform, using the Internet which can facilitate data sharing and exchange can easily carry out off-site, edit neural network structure with multi-user cooperation, and share brain connectome. Experiments and test reports show that the platform is sufficient to support the needs of at least thousands of users editing at the same time, response in milliseconds, meets the real-time operation requirements which provides convenience for neurologists in neuronal tracking and digital reconstruction.

Keywords bioimage informatics neuron tracing real-time editing platform

目录

第 1 章 绪论

1.1 背景与意义

根据原始神经元图像信息的神经元追踪和数字重建是神经科学界热门方向。神经元的形态反应出它的功能，相同功能的神经元通常具有类似的功能。神经科学家通过结构脑图谱的重建，可以反推大脑是如何运作，对理解智慧的产生有重要的帮助。十九世纪以来，神经科学家们开始推测记忆，甚至个性与智力都和大脑神经元之间的连接有密切的联系。图 1-1 展示了秀丽隐杆线虫的神经结构的神经结构，图中每一个节点均代表一个神经元，每一条线代表一个连接。它仅仅由 300 个神经元组成，之间的连接也仅有 7000 个。

White, John G 与 Southgate 等人在 1986 年时已经利用一系列局部原始电子显微照片对秀丽隐杆线虫的神经系统的进行了完整重建[?]。经过了 30 多年的发展，Yunkyu Sohn，Myung-Kyu Choi 与 Yong-Yeol Ahn 等人于 2011 年利用基于模块化的群态检测算法发现秀丽隐杆线虫中包含了 5 个解剖簇及其对应的实验可识别功能电路，进一步揭示了生物电路如何产生更高阶的复杂行为[?]。即使如此，由于神经网络复杂的拓扑结构，神经科学家们仍旧未能充分探索仅仅由 300 个神经元通过突触交织而成的神经网络结构。而人类大脑由一千亿个神经元组成，神经元之间连接的数量又是神经元数量的一万倍，远比秀丽隐杆线虫的神经结构要复杂的多。因此，设计并实现自动神经元重建算法便成了探索神经结构的重要步骤之一。

Druckmann, Shaul 与 Feng 等人开发的神经元重建算法提供了准确的中线，直径，表面，体积和分支点位置，支持沿着神经元表面分析标记过的分子分布，还可以直接导出到建模软件[?]。图 1-2 展示了这种神经元重建算法的样例结果。Brown, Kerry M 与 Barrionuevo 等人收集了来自不同动物，脑区，神经元类型和可视化方法的六个数据集，为自动化软件所需的测试提供了基准，提高了重建的质量，同时最大限度地减少了人工的参与，极大的促进了神经元重建领域的发展[?]。

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。

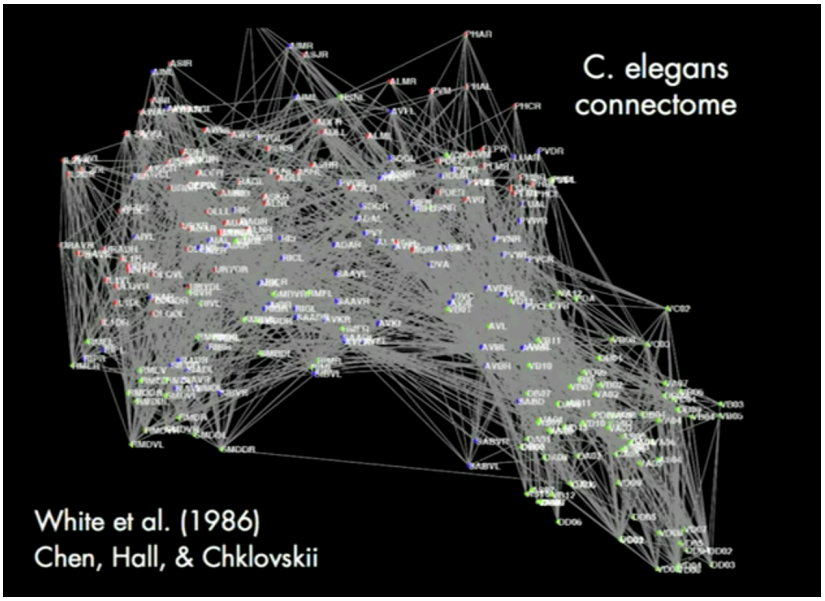

图 1-1 秀丽隐杆线虫的神经结构，其中的点代表了一个神经元结构并标注了相应的名称，线将神经元连接起来，表明对应的神经元通过神经纤维建立了联系

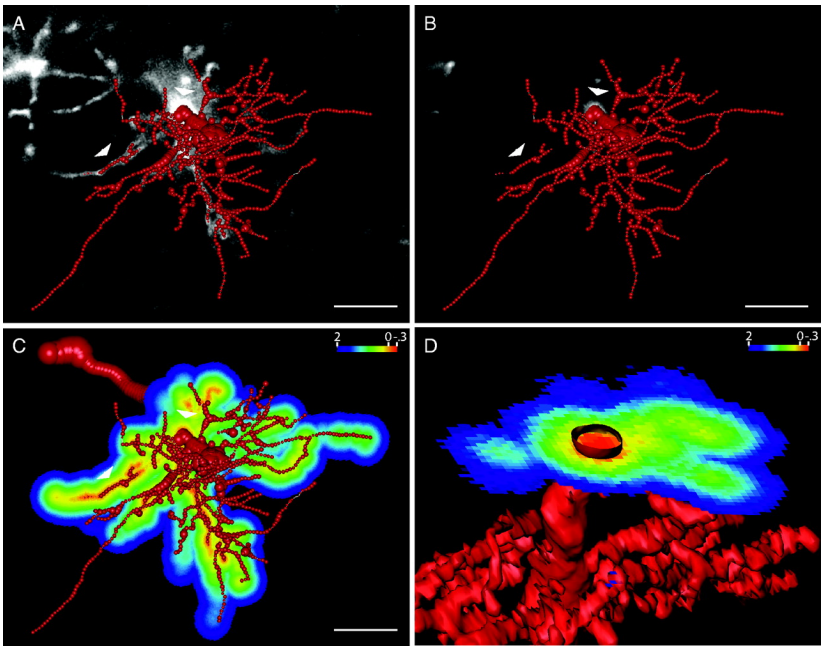

图 1-2 Druckmann 等人的神经元重建算法的样例结果白色的是原始神经元图像，红色的表示完成重建的神经元结构，蓝色和绿色表示对应神经元的表面概率

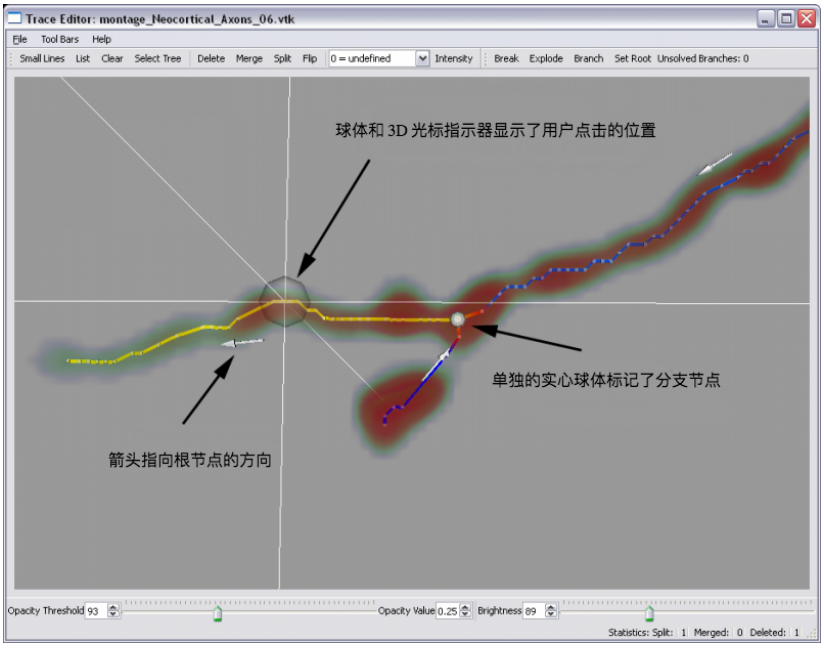

图 1-3 FARSIGHT 软件运行界面箭头指向根节点的方向，单独的实心球标记了分支节点，球体和 3D 光标指示器显示了用户点击的位置

为了便于研究人员进一步研究神经结构，探索智能产生的原因，这就需要在神经元自动重建算法的基础上建立交互式神经元重建系统，进一步提升数字重建结果的准确性，并且将多部分数字重建结果合并起来。

1.2 现有系统的特点与问题

1.2.1 FARSIGHT

FARSIGHT 的软件界面如图 1-3 所示，图中正在编辑的是一小段神经结构，展示了用户点击位置，根节点方向，以及分支节点。

FARSIGHT 的设计目标是保证重建结果的细节，可以快速识别重建结果的错误并能迅速纠正。FARSIGHT 利用基于模式分析辅助集群编辑（PACE）的思想，根据对自动跟踪结果的定量测量和多变量模式分析工具的分析结果，发现其中常见类型的重建错误，提高纠正重建结果的效率[?]。图 1-4 展示了 FARSIGHT 导出的较大规模的神经结构重建结果。

FARSIGHT 的缺点在于，它专注于半自动重建，对于常见的错误修改效率确

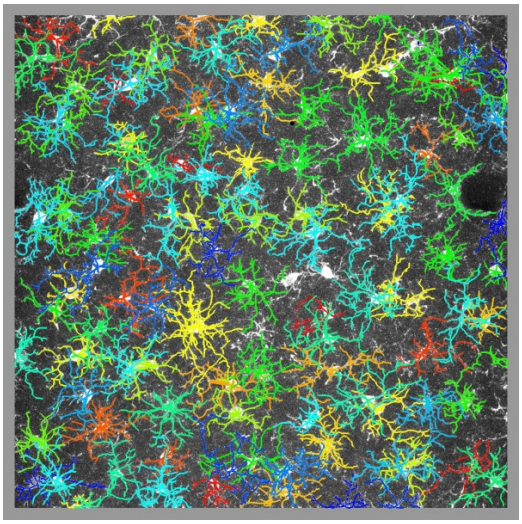

图 1-4 FARSIGHT 导出的重建结果每种颜色均代表了一部分神经结构

实较高，但是如果遇到细小的错误，FARSIGHT 无法识别，也不提供精细化的编辑手段，需要借助于其他软件完成。

1.2.2 neuTube

neuTube 是一种基于 SWC 文件格式的神经元重建软件，同时具备 2D 和 3D 的可视化以及直观地编辑、绘制功能，允许用户高效的根据荧光图像数据重建神经元结构，并且支持编辑其他软件生成的标准神经结构的文件[7]。软件界面如图 1-5 所示。

虽然 neuTube 提供了 2D 和 3D 模式下精细编辑神经结构的功能，但是无法进行多人协同编辑，无法分享完成重建的结构脑胞体。

通过分析不难看出，现有的软件难以支持多名神经科学家同时进行精细化的神经结构编辑。由于神经结构的复杂性，团队合作进行神经结构的编辑，结果合并等是未来神经科学发展的趋势，因此需要设计并开发一个支持多用户协同工作的在线交互式神经元编辑平台便显得尤为重要。

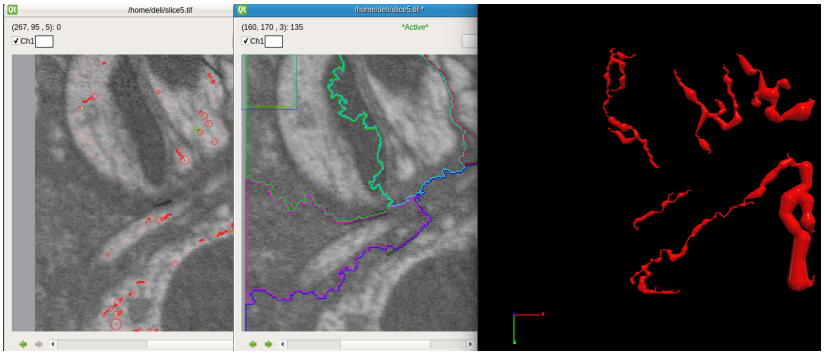

图 1-5 neuTube 软件运行界面图中展示了神经元自动重建与编辑的过程，第一个界面中红色的点是随机产生的种子点，第二个界面是根据种子点对神经元结构进行分割，图三是完成重建的神经元结构并支持用户进行纠错、编辑

1.3 论文结构

本文旨在设计并实现在线多用户的神经元网络结构编辑分享平台，利用互联网便于数据共享与交流的特点解决一些现有神经元编辑软件的问题，使得神经科学家可以便捷地进行异地，多用户协同编辑神经元网络结构，并能分享完成重建的结构脑图谱，解决了传统交互式神经元编辑软件无法进行团队合作的问题，提升了神经科学家的工作效率。由于项目涉及到数据可视化与后台服务器搭建，自然地将整个项目分成两部分，这里主要实现后台服务器的搭建，为前端可视化操作提供相应的 API 支持。

第一章讨论了交互式神经元重建系统的背景与意义以及现有的神经元编辑软件存在的问题，第二章讨论了项目整体架构并简单介绍所用到的技术。第三章讨论技术实现的细节以及如何根据性能测试结果进行性能优化，第四章描述了在性能优化之后的系统整体性能，第五章展示并分析了在实际使用中平台的性能表现，第六章讨论了接下来的工作并分析了项目中存在的不足。

1.4 本章小结

本章论述了神经元追踪和数字重建对于神经科学方面的重要意义以及神经网络复杂的拓扑结构造成的困难，粗略讨论了一些自动化神经元追踪和数字重建

算法的特点。通过分析现有的交互式神经元重建软件的优点以及存在的问题，体现了在线多用户交互式神经元编辑平台的优势，并描述了本文结构。

第 2 章 整体架构与技术选型

2.1 整体架构

项目整体结构如图 2-1 所示，共分为神经信息数据库，用户信息数据库以及网络服务器三部分组成。图中主要包含了两个方向的数据流，一个是用户信息数据流，另外一个是神经信息数据流。

2.1.1 用户信息数据流

用户信息数据流在用户信息数据库与网络服务器之间流动，主要负责三件事情，一是验证用户身份，二是获取用户资源列表，三是检查用户行为是否有足够的权限。用户信息数据流可以抽象成如下几步：

1. 用户向网络服务器发送请求，登录平台或注册新用户
2. 网络服务器向用户信息数据库验证用户身份或添加新的用户信息
3. 用户信息数据库返回验证结果给网络服务器
4. 网络服务器将结果反馈给用户

2.1.2 神经信息数据流

神经信息数据流在神经信息数据库，用户信息数据库以及网络服务器三部分之间流动，负责维护三对关系，用户和原始神经图片信息的关系，用户和数字重建结果直接的关系以及数字重建结果和原始图片信息之间的关系。在获取资源的时候需要用户信息数据流辅助进行权限控制。神经信息数据流可以抽象成如下几步：

1. 用户向网络服务器请求神经元信息
2. 网络服务器利用用户信息数据流中保存的用户信息返回对应用户资源列表
3. 用户信息数据库返回用户资源列表给网络服务器
4. 网络服务器根据用户资源列表向神经信息数据库查询对应神经信息
5. 神经信息数据库返回对应神经信息给网络服务器
6. 网络服务器返回用户所需的神经信息用于前端的可视化展示

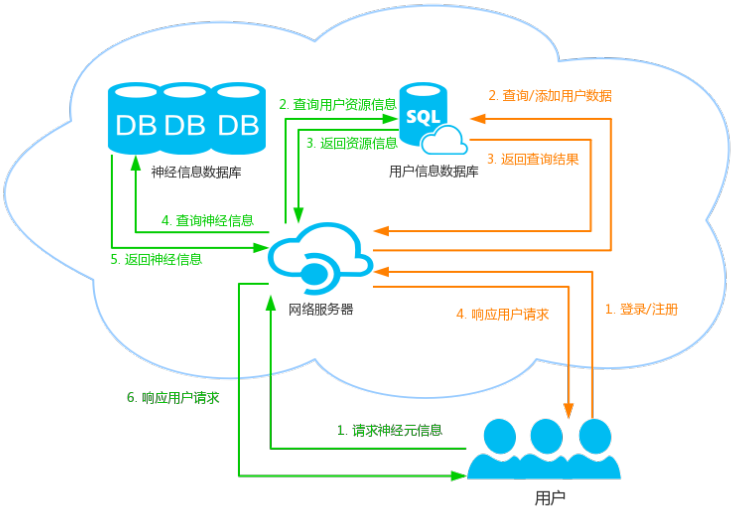

图 2-1　项目整体架构包含了神经信息数据库，用户信息数据库与网络服务器三部分，展示了用户信息数据流与神经信息数据流两个主要的数据流

2.2　技术选型

2.2.1　神经信息数据库

神经信息数据库包含两部分数据，一部分是原始大脑切片显微镜图像，另外一部分是初步完成数字重建的 SWC 文件。使用 DVID 作为数据库，储存这些信息。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[7]。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，构建出原始大脑切片显微镜图像与数字重建结果的储存仓库，将数据储存抽象成数据存储服务，使得可以专注于完成核心算法和逻辑。

2.2.2 用户信息数据库

用户信息数据库包含多用户管理以及用户资源管理。使用 PostgreSQL 数据库储存这部分信息。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。

2.2.3 网络服务器

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。

2.3 本章小结

本章讨论了在线交互式神经元编辑平台的整体架构与两条数据流，简单分析了神经信息数据库，用户信息数据库和网络服务器三个主要部分所涉及到的技术及其特点。

第 3 章 平台搭建与性能调优

3.1 数字重建结果的表示

为了储存完成数字重建的结构脑胞体，并为前端的可视化操作提供相应的 API 支持，需要定义一种存储结构脑胞体的格式，这种格式需要满足如下特点：

1. 便于储存

2. 便于前后端之间传输

3. 支持对神经元的操作，并支持历史版本记录

根据“所见即所得”的原则，Peng, H. 和 Long, F. 等人设计出了 SWC 格式[?]。SWC 的结构清晰，可操作单元定义名曲，用户输入与编辑操作有清晰明确的对应关系。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。

Feng, Linqing 与 Zhao 等人用点集高度抽象了结构脑胞体，将其表示成树状结构，进一步提升了对神经元结构的表达能力[?]。用点集定义结构脑胞体如下所示：$\{n_i = (x_i, y_i, z_i, n_j) | i = 1, ..., N, j = 0, ..., N, i \neq j, x_i, y_i, z_i, r_i \in R\}$，点与神经元之间的连接构成了一个树形结构，合法的神经元结构不允许出现环。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，用来表示神经结构的根，n_j 表示 n_i 的父节点。从 n_i 到 n_j 表示一段向上的路径，这可以用一个数组 $(n_{k_1}, ..., n_{k_n})$ 来表示, 其中 n_{k_i+1} 代表 n_{k_i} 的父节点 $k_1 = i, k_n = j$。在这个模型中，基础结构单元是点，点定义了神经元结构的储存方式以及交互方式。

但是仅仅定义结构脑胞体无法满足多用户编辑的需求。因为在多用户同时编辑的时候，可能存在操作冲突即两个用户同时对相同的结构脑胞体进行操作，这是需要区分出不同用户的操作，因此需要进一步将结构脑胞体的操作抽象出来并储存在数据库之中。

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，$f(S_1) = \phi$, 式中 ϕ 代表空集，f 表示将点集 S_1 变换成了空集，意味着将点集 S_1 删除，因此 f 定义了删除操作。在定义接下来的操作之前，需要明确点的

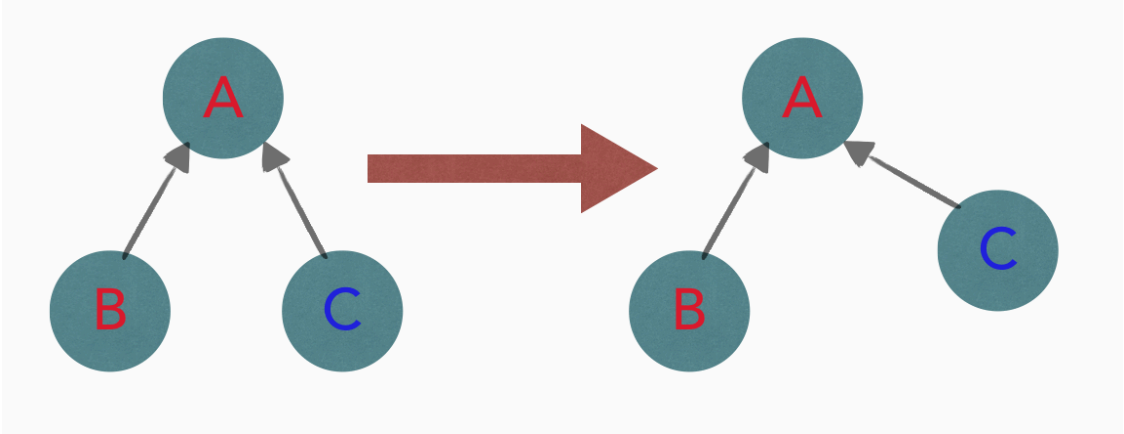

图 3-1 由三个神经元组成的几何变换实例根节点是神经元 A，神经元 C 的位置向右上方发生了一些改变，此时只需要改变表示神经元 C 元组的 G(n) 部分即可，神经元 A，B 不需要改变

表示方式。简单来讲，点的表示只需要记录位置信息与父节点即可，但是为了提高遍历节点的效率，在点的表示中加入了一些冗余信息，例如子节点与兄弟节点。$n = (G(n), P(n), C(n), S(n))$，其中 $G(n) = (x(n), y(n), z(n), r(n))$。$n$ 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 $S(n)$。查询子节点时，这样的表示方式只需要检查它的第一个子节点并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，算法需要检查树中的每个节点。

根据上述对操作的抽象，可以将编辑节点 n 的值可以用修改对应的元组来定义。对 $G(n)$ 的改变称为几何变换，对 $P(n), C(n), S(n)$ 的变换成为结构变换。几何变换是较为直观的，仅仅代表了神经元在空间上的位移，图 3-1 展示了最简单的几何变换。结构变换改变了结构脑胞体之间相互连接的关系，图 3-2 展示了一个结构变换，此时单独修改编辑过的神经元对应的元组信息是不够的，会造成非法的结构脑胞体。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，需要进一步对结构脑胞体的结构操作在不同层次上进行抽象。

第一层次操作包含三个基本操作，分别表示了对 $P(n), C(n), S(n)$。这三种操

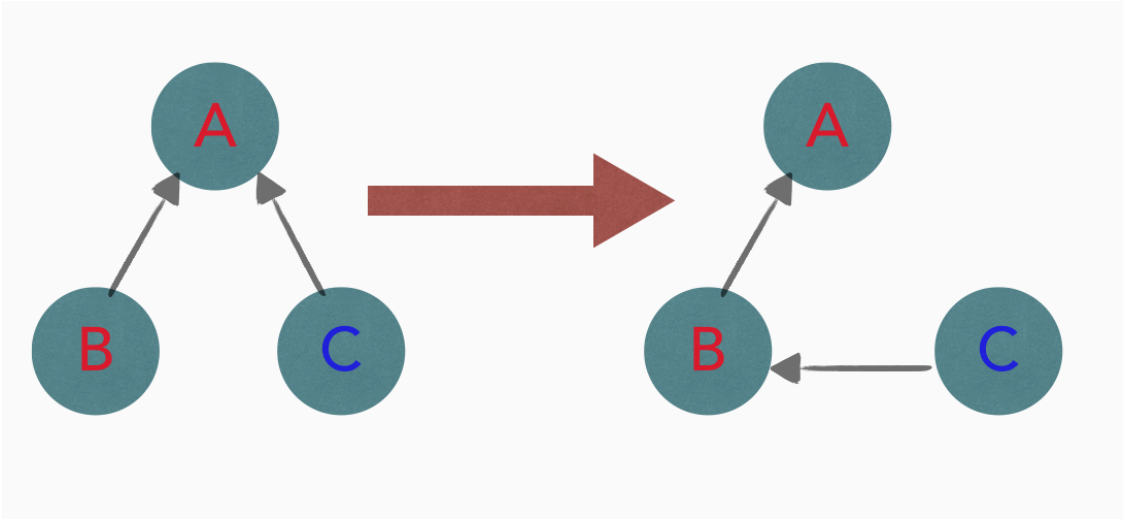

图 3-2 由三个神经元组成的结构变换实例根节点是神经元 A，神经元 C 的父节点从神经元 A 变成了神经元 B，此时只需要改变表示神经元 C 元组的 P(n) 会造成非法的神经元结构，需要相对于的调整神经元 A 与神经元 B 的元组信息

作可以用以下公式定义：

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

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

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

在这个层次上，仅仅表示了三种最基本的结构操作，分别是修改子节点，修改父节点与修改兄弟节点，这一层次的操作只修改了被编辑神经元的数据，无法保证结构的合法性。

第二层次操作将三种基本的操作进行了封装，保证了结构的合法性。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 是父母的第一个子节点,

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

其中，$f \circ g$ 代表一个复合操作，$S(S^{-1}(n)) = n$。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，定义更多的操作对于储存以及前后之间的交互提供了极大的便利。

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

对于图 3-2 中所示的结构变换所对应的第三层次操作包含了三个第一层次的操作：

1. 修改神经元 C 的父节点
2. 修改神经元 B 的兄弟节点
3. 修改神经元 B 的子节点

而对于图 3-3 中所示的结构变化与图 3-2 所示的改变对称。但是由于神经元 C 是根节点神经元 A 的第一个子节点，所以与图 3-2 包含的第一层次操作不同，由 4 个第一层次操作构成：

1. 修改神经元 C 的父节点
2. 修改神经元 B 的子节点
3. 修改神经元 A 的子节点
4. 修改神经元 C 的兄弟节点

通过对简单操作进行抽象，使得不需要关注与有大量细节并且可能造成非法结构的第一层操作，避免了许多可能出现的问题，并且减少了一些操作合法性的检查，减少了计算量。

第三层次由一组复合操作组成，其中包括任何第二层次的操作复合而成的操作。将一个操作分解为基本操作有助于保证编辑操作的有效性。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，这样可以很容易构造撤消操作的逆转序列。

第三层操作由第二层复合而来，第二层操作由第一层复合而来，而第一层操作只包含三种情况，这样便可以用三种操作来表示结构结构变换，加上可以直观

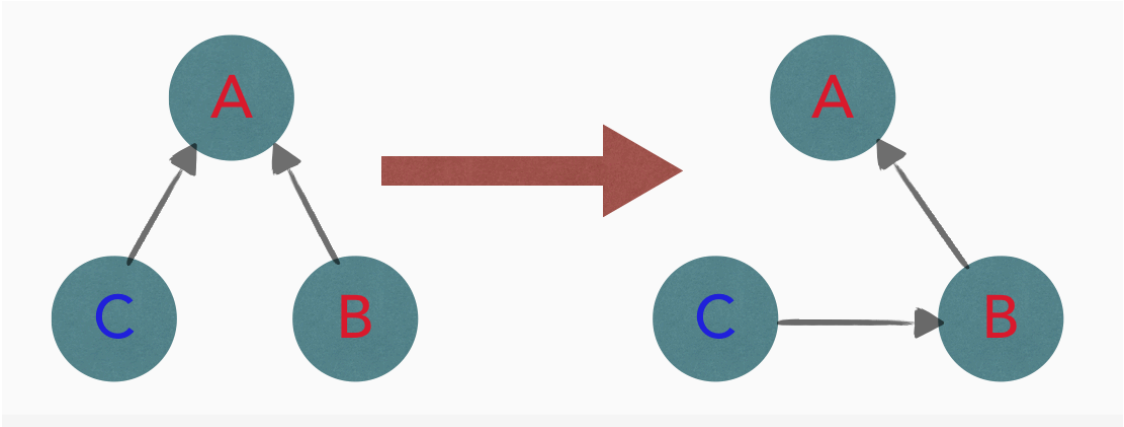

图 3-3 另一种由三个神经元组成的结构变换实例根节点是神经元 A，神经元 C 作为根节点神经元 A 的第一个子节点在第二层次的操作需要修改神经元 A 的子节点

的用三种操作表示的几何变换，共可以用六种操作表示所有的对神经元结构的操作。这样便可以针对这六种简单操作设计不同的 API 供前端调用，使得针对操作的特点进行优化成为了可能。根据第三层次的抽象，流程处理用户的编辑请求的算法如 3.1 所示。

这里值得注意的是需要储存在数据库中的是第二层次的编辑操作，作用在结构脑胞体上的是第一层次操作。考虑到需要支持撤销，多用户操作合并等功能，保存第一层次操作无法保证编辑操作的完整性，为实现这些功能增加了难度。另外，如果只撤销了编辑操作的一部分很可能造成非法的结构脑胞体。而编辑操作作为一种高层次的抽象，无法直接作用在结构脑胞体之上，需要转换成更底层的可以直接反应结构脑胞体元组之上的第一层操作。

3.2 数据库结构设计

3.2.1 用户信息数据库

为了实现多用户管理以及用户资源管理，共设计实现了三张数据表，分别是用户信息表，原始图像数据表以及结构脑胞体数据表。这些数据表主要用户实现用户信息数据流，并和 DVID 储存的神经信息数据库相互关联，为神经信息数据流提供了资源管理和权限控制等相应的支持。

Algorithm 1 将用户编辑操作转换成操作序列

```
获取可用编辑操作列表 // 第二层次操作
if 有所需的编辑操作 then
    选取所需编辑操作
else
    用户自定义编辑操作 // 通过组合第一层次操作构建第二层次操作
    while 自定义操作不合法 do
        根据提示修改掉操作不合法的地方
    end while
    保存用户的自定义编辑操作
end if
将编辑操作储存到数据库中
将编辑操作转换成第一层次操作
修改结构脑胞体
```

表 3-1　用户信息表

字段名	数据类型	备注
username	STRING	主键
password	STRING	
salt	UUID	用于保证用户账户安全

1. 用户信息表用户信息表如表 3-1 所示，共有三个字段，分别储存了用户名，密码和用户保证账户安全的盐值。

2. 原始图像数据表原始图像数据表如表 3-2 所示，共有三个字段，分别储存了创建者，图像名和权限控制的用户角色。

3. swc 数据表原始图像数据表如表 3-3 所示，共有五个字段，分别储存了创建者，图像名，创建时间，swc 文件名以及用户评论。根据创建时间，可以建立同一图像下 swc 文件的拓扑顺序，为多用户同时编辑以及合并冲突分支提供了基础。

表 3-2 原始图像数据表

字段名	数据类型	备注
username	STRING	创建者，外键，用户信息表中的 username
image	STRING	
role	STRING	用于权限控制

表 3-3 原始图像数据表

字段名	数据类型	备注
username	STRING	创建者，外键，用户信息表中的 username
image	STRING	原始图像名，外键，原始图像数据表中的 image
createdAt	TIME	创建时间
swc	TEXT	swc 文件名
comments	STRING	备注

3.2.2 神经信息数据库

神经信息数据库储存了原始神经切片图像信息，完成数字重建的结构脑胞体以及用户的编辑操作。DVID 提供了 imagetile 和 Key-Value 两种类型的数据类型，原始神经切片图像信息选择使用 imagetile 数据类型，结构脑胞体以及用户的编辑操作选择了 Key-Value 数据类型。

1. 原始神经切片图像信息如图 3-4 所示，对于 imagetile 类型的数据提供了数据预览功能。图中展示的神经组织在 z 方向上是一层层的切片，在 xy 方向上是完整的一整张图片。由于在精细神经编辑编辑时需要获取局部信息，并对局部信息进行放大，因此提供的 API 需要提供获取图像的范围以减少数据的传输量，提升响应速度。

2. 结构脑胞体与用户的编辑操作由于完成数字重建的结构脑胞体结构多样，数据量较大，涉及多种可能进行的编辑操作且需要支持用户自定义编辑操作，传统的结构化数据库较难支持这样的功能。DVID 支持的 Key-Value 数据类型提供了较为简单的 NoSQL 储存服务，满足了目前的需要。NoSQL 系统的一个关键特性是“无共享”，这使得储存服务，满足了目前的需要。NoSQL 系统可以在许多

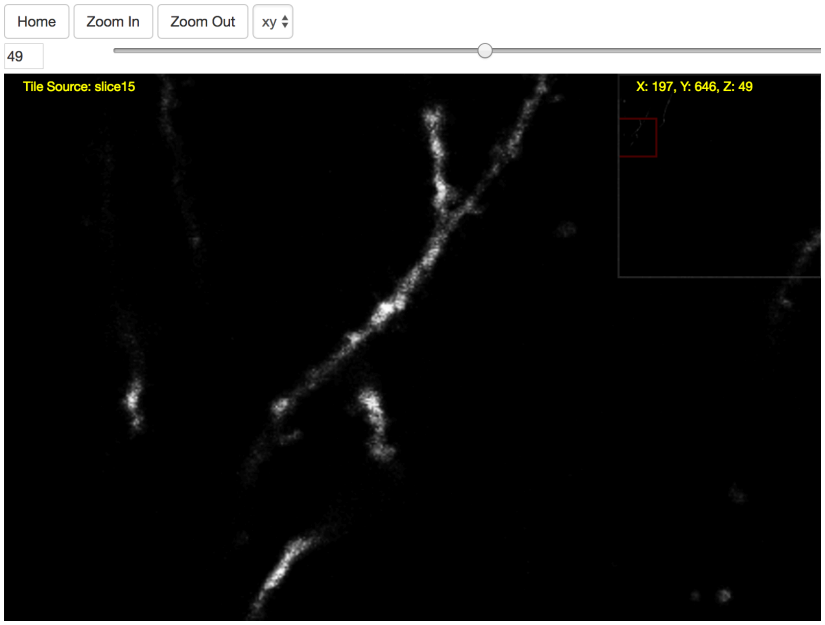

图 3-4　DVID 对于 imagetile 类型数据提供的数据预览功能图中展示了一部分神经组织

服务器上复制并分割数据，这使得 NoSQL 系统每秒可以支持大量简单的读/写操作[?]，这正好符合结构脑胞体与用户的编辑操作数据量大，无固定格式的特点。

3.3　基础平台的搭建

在确定了结构脑胞体的表示方式并完成了数据库结构的设计之后，基础平台的搭建更多是工程上的考虑与工作。项目初期针对可能会涉及到的技术做了广泛而细致的调查与研究，最终确定了在上一章中的技术选型，选择了合理且先进的技术使得开发工作完成的十分顺利，可以将时间用在测试与性能调优。

3.3.1　跨浏览器兼容

由于网络浏览器市场竞争激烈，浏览器的开发速度非常快，新增功能不需要供应商之间的协调。引入新功能的优先级通常会优先于错误修复，导致浏览器不稳定，不符合 Web 标准，频繁崩溃并且具有许多安全漏洞。为了标准化 JavaScript，Ecma International 在 ECMA-262 和 ISO / IEC 16262 中提出了 ECMAScript [?]。在项目中希望使用 ECMAScript 来编写网站，使平台更加符合互联网标准。然而在

特性名称	chrome	es5-shim	Kong 4.3	Kong 4.4	Kong 4.9	Kong 4.13	KQ 4.14[2]	IE 7	IE 8	IE 9	IE 10
	98%	32%	23%	23%	42%	66%	66%	0%	12%	83%	99%
Object/array literal extensions	5/5	0/5	2/5	2/5	2/5	3/5	3/5	0/5	0/5	5/5	5/5
Object static methods	13/13	1/13	0/13	0/13	3/13	13/13	13/13	0/13	2/13	13/13	13/13
Array methods	11/12	12/12	9/12	9/12	10/12	10/12	10/12	0/12	0/12	12/12	12/12
String properties and methods	2/2	1/2	1/2	1/2	2/2	2/2	2/2	0/2	1/2	2/2	2/2
Date methods	3/3	3/3	2/3	2/3	2/3	3/3	3/3	0/3	0/3	3/3	3/3
Function.prototype.bind	Yes	Yes	No	No	No	Yes	Yes	No	No	Yes	Yes
JSON	Yes	No	No	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Immutable globals	3/3	0/3	0/3	0/3	3/3	3/3	3/3	0/3	0/3	3/3	3/3
Miscellaneous	8/8	1/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	7/8	7/8
Strict mode	19/19	0/19	0/19	0/19	0/19	0/19	0/19	0/19	0/19	0/19	19/19

图 3-5　部分浏览器对 ECMAScript5 的支持程度

目前的互联网环境中，各个浏览器对 ECMAScript 的支持并不好，很多的神经学家依旧在使用较为老旧的浏览器，ECMAScript 在这样的环境中无法运行。图 3-5 展示了一些较为老旧的浏览器对 2009 年推出 ECMAScript5 的支持程度，IE7 完全不支持，甚至是较为现代的 Chrome 浏览器也没有完全支持。Juriy Zaytsev 等人详细讨论了不同浏览器对不同版本 ECMAScript 中每个特性的支持程度，各个浏览器的兼容性不容乐观[?]。HTML 和 css 也存在这样的问题，很多现代的特性在老旧的浏览器中无法使用。

为了解决这个问题，既可以让前端可视化人员能使用先进的特性，也可以保证在老旧的浏览器环境中可以正常运行，需要将现代的语法标准针对每种不同的浏览器，转换到每种浏览器自己的标准上。配合使用 webpack [?] 这一工具，针对使用率较高的浏览器的标准完成了这一工作。前端可视化人员可以随意的使用最先进的支持的并不好，很多的神经学家依旧在使用较为老旧的浏览器，ECMAScript 7，HTML 5 以及 CSS 3 等技术来完成可视化和神经结构编辑任务而不用考虑老旧的浏览器是否支持这些技术，经过 webpack 及其一些列定制化的插件[?, ?]，将其转换到不同浏览器自己的标准上，较为完善的解决了这一问题。图 3-6 描述了 webpack 的工作流程，将各种所需的依赖打包在一起，通过不同的插件将浏览器不支持的新特性转换到最基础的语法上，保证了各个浏览器的兼容性。在转换的过程中还实现了代码混淆和打包压缩，一定程度上提高了安全性和效率。

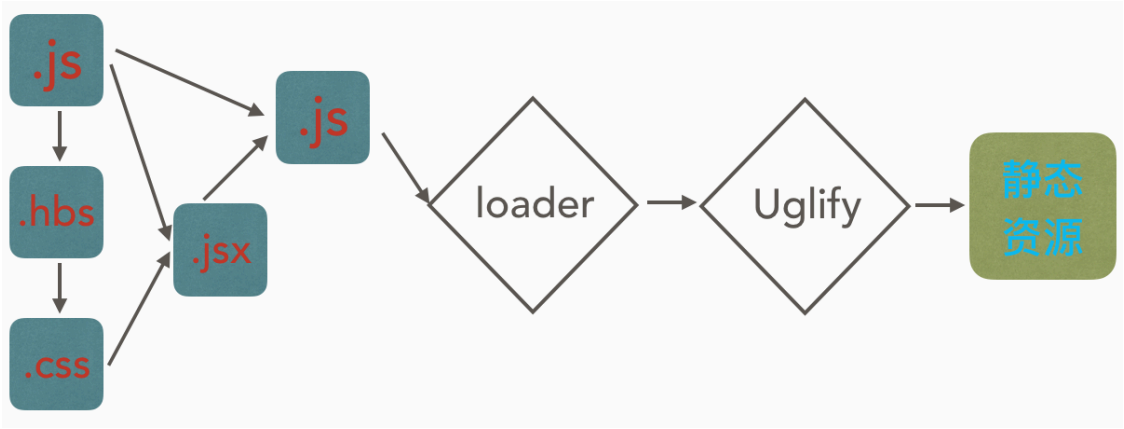

图 3-6 webpack 的处理流程，.js 表示所需要的 JavaScript 模块，.hbs 表示网页模板系统，.jsx 表示 React [?] 的各个组件，.css 表示网站样式资源。中间的 loader 和 Uglify 表示代码转译，混淆和压缩的插件。右边的静态资源文件代表发送向客户端的文件。

3.3.2 原始图像信息传输

由于在编辑神经元结构的过程中需要对照原始图像信息，而每一张图片仅仅代表神经组织的一个切片，如果在 3D 模式下进行编辑会设计到许多张原始图片，因此图像信息的传输占据了网络服务器服务器绝大多数的流量。因此实现原始图像信息的传输在平台搭建的过程中占据了重要地位。

原始神经元图像，如图 3-7 所示，有两个特点：

1. 背景面积大，实际的神经元组织在图像中的面积较小。

2. 绝大多数图像是灰度图，数据只有一个通道，甚至有部分图像是黑白图像，只有 0 和 1。

针对神经元图像的这个特点，可以针对性的传递图片中的神经元组织，忽略掉大面积的黑色背景，这样做会大大降低所需要传递的数据量。由于神经元组织在荧光显微镜下显示为白色，图片背景为黑色，因此神经组织的表示如下：

$$F = f | f \in image \ \&\& \ f > threshold$$

背景表示如下：

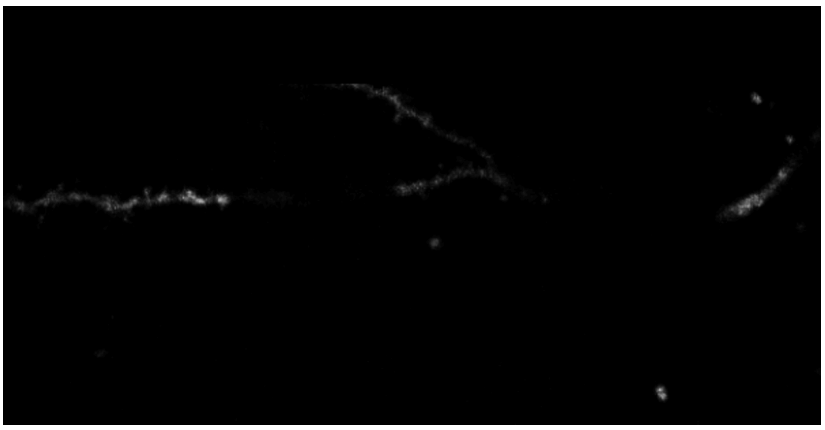

图 3-7 原始神经元图像样例

$$B = b|b \in image \ \&\& \ b \leq threshold$$

对前景神经组织采用 CRS [?] 的方式，采用 A 与 IA 两个数组储存。

但是这样做一方面需要判断图片中的神经组织，增大后端服务器的计算量，另一方面对于一些不同显微镜下的图片神经元在图像中的比例较大，这种优化方式甚至会加大数据传输量。最终采取的解决方案是在后端服务器计算资源空闲的时候对原始图像信息进行处理，对于处理完毕且图像中神经元组织较小的图片只传输神经元组织信息，反之传递原始图像，具体过程如算法 3.3.2 所示。。这样做在绝大多数场景下取得了较高的性能。

3.4 平台硬件架构

在软件编写完成之后，共使用三台计算机作为网络节点部署了在线交互式神经元编辑平台。其中一台计算机部署神经信息数据库和用户信息数据库，另外两台部署网络应用服务器用于负载均衡。每台计算机的详细配置如下：

1. 操作系统：Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-72-generic x86_64)

2. 中央处理器：Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz

3. 内存：63149 MB

4. 网卡：Ethernet controller: Broadcom Corporation NetXtreme II BCM57800 1/10 Gigabit Ethernet * 4

Algorithm 2 图像信息发送过程

```
读取用户请求
从 DVID 中读取对应的神经元图像信息
if 图像缓存中存在处理过的图像信息 then
    发送缓存中的图像信息
else
    if 图像中背景占据绝大多数位置 then
        将图片发送到服务器处理队列，在服务器空闲时进行处理
    end if
    发送原始图像信息
end if
```

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Req/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	31000	31000	0	0%	110.32	5	23	39	367	739	4028	68	150
login	1000	1000	0	0%	3.559	8	58	68	104	170	202	58	27
Images	10000	10000	0	0%	35.587	101	125	159	166	184	243	163	45
Swcs	10000	10000	0	0%	35.587	98	115	142	157	162	252	154	49
SwcContent	10000	10000	0	0%	35.587	20	47	256	634	1053	4028	166	235

图 3-8　未经过任何优化的性能数据。图中 Login 代表用户登录的请求，Images 代表获取用户原始图像列表相关 API 的响应时间，Swcs 代表用户对神经元结构进行操作的相关 API 的响应时间，SwcContent 代表用户获取结构脑胞体的响应时间。

5. 硬盘：1.75 TB 机械硬盘

3.5　性能优化

在测试平台搭建完成后，使用 Gatling[?] 进行了压力测试，测试结果如图 3-8 所示。测试中一共模拟了 100 名用户同时访问平台，每位用户模拟进行不同的操作共计 301 次，所有用户共计 30100 次操作，平均每秒进行 2508.333 次操作。

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Req/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	11000	11000	0	0%	215.686	5	15	22	59	86	196	21	17
login	1000	1000	0	0%	19.608	9	58	69	97	155	196	59	25
SwcContent	10000	10000	0	0%	196.078	5	14	19	33	56	118	17	9

图 3-9　经过优化的获取结构脑胞体响应时间，从 166ms 降低到了 17ms，降低了 89.75%

3.5.1　优化获取结构脑胞体的响应时间

从图 3-8 中可以看出获取结构脑胞体的响应时间较长，最长响应时间甚至达到了 4028ms，这对于用户来说是可以明显感受到的卡顿，甚至是不可接受的卡顿，这使得需要对获取结构脑胞体的 API 进行性能优化。考虑到这个 API 需要传输大量的数据，响应时间过长是由传输数据量较大导致的性能问题，可以考虑从减少传输的数据量入手。经过调查相关技术，最终采用两种方式进行优化。第一个方案是采用浏览器端的数据库 indexDB 进行缓存[?]，将已经获取过的结构脑胞体储存在客户端中，减少了向服务器的请求次数。另一方面，根据 Mogul，Jeffrey C 等人提出的在 HTTP 中提出的有关使用的编码和压缩方式的技术对结构脑胞体的数据进行了压缩，使得传输的数据量更小。结合这两点，将获取结构脑胞体的平均响应时间从 166ms 降低到了 17ms，降低了 89.75%，具体数值如图 3-9，详细的响应时间分布如图 3-10 所示，获取结构脑胞体的响应时间大大降低。图 3-11 描述了经过优化的获取结构脑胞体响应时间随时间的变化，高响应时间的请求集中在开始时，后续的请求响应时间几乎为零，这显示了使用 indexDB 对数据进行缓存所带来的巨大性能提升。

3.5.2　使用 Redis 和负载均衡降低响应时间

在优化获取结构脑胞体的响应时间的过程中使用了浏览器端的缓存进行优化，这启发我们使用服务器端的缓存来进一步优化。Redis 是一个支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库[?]。由于用户大概率修改编辑最新导入的数据，因此可以将近期的原始图像数据信息缓存在 Redis 之中，减少了请求数据库的次数，提升了整体性能。此外将数据库与网络服务器部署在不同的服务器上，降低了单台服务器的压力，大大提升了每秒响应请求的数量，使之可以支撑更多的用户同时访问。由于网络服务器的处理压力仍然较大，进一步将网络服

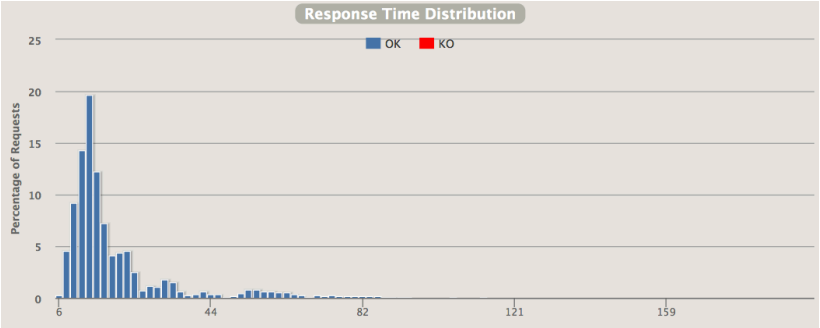

图 3-10　经过优化的获取结构脑胞体响应时间分布，绝大多数的响应时间在 6 44 ms 之间，用户感受不到明显的卡顿

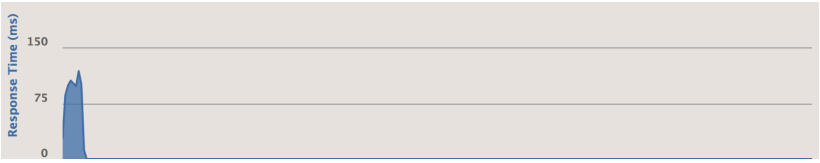

图 3-11　经过优化的获取结构脑胞体响应时间随时间的变化，高响应时间的请求集中在开始时，后续的请求响应时间几乎为零

务器的处理任务分散到两台服务器上，进一步降低了网络服务器的计算压力，优化过后的测试结果如图 3-12 所示，详细的测试分析在第四章论述。

3.6　本章小结

本章主要讨论了如何将用户的编辑操作转化成可以储存在数据库中的操作序列，描述了用户操作在三种不同层次上的抽象，根据实际需要选择不同层次的抽象，合理高效的储存了用户编辑操作，保证了结构脑胞体的有效性，并支持多用户操作合并，操作撤回等功能。讨论了用户信息数据库与神经信息数据库的实现细节，平台部署环境与方法，并根据性能测试报告进行了一系列的性能优化。

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Req/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	30010	30010	0	0%	326.196	0	3	4	9	18	151	4	4
login	10	10	0	0%	0.109	8	65	76	130	147	151	67	39
Images	10000	10000	0	0%	108.696	1	3	5	13	23	73	5	4
Swcs	10000	10000	0	0%	108.696	0	3	4	8	20	63	4	3
SwcContent	10000	10000	0	0%	108.696	0	2	3	4	8	22	2	2

图 3-12　经过优化的整体响应时间报告，图中 Login 代表用户登录的请求，Images 代表获取用户原始图像列表相关 API 的响应时间，Swcs 代表用户对神经元结构进行操作的相关 API 的响应时间，SwcContent 代表用户获取结构脑胞体的响应时间。可以看出绝大多数响应时间都在 100 ms 一下，性能大大提升。

第 4 章 实验成果

阿斯达

第 5 章 性能分析

在完成第 第 3 章 章中描述性能的性能优化后，将其部署在测试平台下，测试平台由 3 台计算机构成，硬件配置如下所示：

1. 操作系统：Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-72-generic x86_64)

2. 中央处理器：Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz

3. 内存：63149 MB

4. 网卡：Ethernet controller: Broadcom Corporation NetXtreme II BCM57800 1/10 Gigabit Ethernet * 4

5. 硬盘：1.75 TB 机械硬盘

相比第 第 3 章 章中的硬件环境，更多的是计算机之间的物理距离更近，计算机之间的网络延迟更低，更利于分布式平台的搭建。

5.1 请求响应时间分析

Grigorik 等人通过分析大量用户数据，指出用户即使平时生活中接触不到毫秒级别的时间，但是仍能明显感受到毫秒级别的延迟。延迟在 100ms 以内的时候，用户会认为立即得到了响应；在 100 300 ms 之间，用户会感觉到轻微的延迟；300 1000 ms 之间，用户仅仅认为系统工作正常，但体验较差；当延迟达到 1000 ms 之上的时候，用户会去做其他工作，不时查看任务是否已经完成；如果延迟达到了 10,000ms 的时候，用户开始怀疑任务程序是否出现了问题，放弃执行任务[?]。因此，为了满足神经科学家流畅的进行对神经元的实时编辑任务，需要使服务器响应时间控制在 100ms 以内。需要注意的是，用户真正感受到的响应时间还包括前端可视化工作所需的计算时间以及浏览器的渲染时间等，这对服务器的响应时间有了更高的要求。

测试主要针对获如下三类调用量较大的 API 进行测试：

1. 取用户原始图像列表相关 API 2. 用户对神经元结构进行操作的相关 API 3. 获取完整结构脑胞体的响应时间

在这部分的测试中，模拟 100 名用户进行编辑，每名用户的编辑动作进行 3

Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Req/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	30010	30010	0	0%	326.196	0	3	4	9	18	151	4	4
login	10	10	0	0%	0.109	8	65	76	130	147	151	67	39
Images	10000	10000	0	0%	108.696	1	3	5	13	23	73	5	4
Swcs	10000	10000	0	0%	108.696	0	3	4	8	20	63	4	3
SwcContent	10000	10000	0	0%	108.696	0	2	3	4	8	22	2	2

图 5-1 请求响应时间，图中 Login 代表用户登录的请求，Images 代表获取用户原始图像列表相关 API 的响应时间，Swcs 代表用户对神经元结构进行操作的相关 API 的响应时间，SwcContent 代表用户获取结构脑胞体的响应时间。

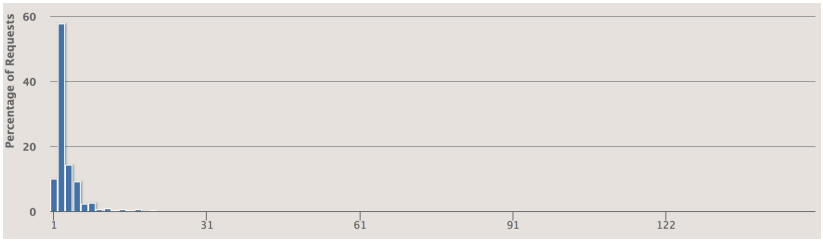

图 5-2 所有请求响应时间的分布，可以明显的看出绝大多数请求的响应时间集中在 0-20 ms 之间，极少数请求的响应时间超过了 100 ms。

种操作各 100 个，所有用户共计进行 30,000 次 API 调用。测试结果如图 5-1 所示，响应时间的分布如图 5-2 所示。从图中可以明显的看出绝大多数请求的响应时间集中在 0-20 ms 之间，极少数请求的响应时间超过了 100 ms。按照 Grigorik 分析大量用户数据得出的标准，满足了实时操作的要求。

进一步分析响应时间随时间的变化，如图 5-3 所示，可以看出在测试开始时，由于没有缓存，请求的响应时间较高，随着时间的推移，绝大多数的请求被服务器端或者浏览器端缓存下来，响应时间趋于平缓。这说明了使用 Redis 与浏览器端缓存对性能的提升明显，将需要花费一百毫秒才能完成的计算任务降低到可以在毫秒级别内响应，另一方面也降低了服务器的计算压力，使得用户的体验更加流畅。

5.2 响应请求数量分析

为了支持多用户同时编辑，平台需要支持多用户同时在线，能够处理大量的并发请求。测试结果如图 5-4，可以看出，经过性能优化之后

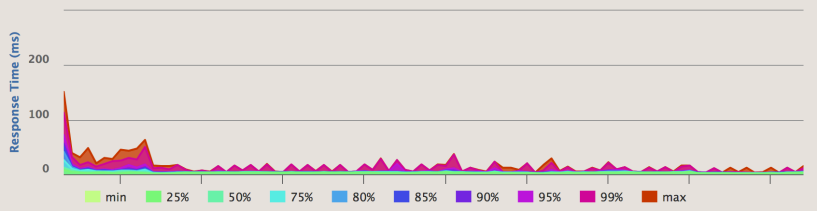

图 5-3 请求响应时间随时间的变化，不同颜色标明了不同比例请求所使用的时间。测试开始时，由于没有缓存，请求的响应时间较高，随着时间的推移，绝大多数的请求被服务器端或者浏览器端缓存下来，响应时间趋于平缓。

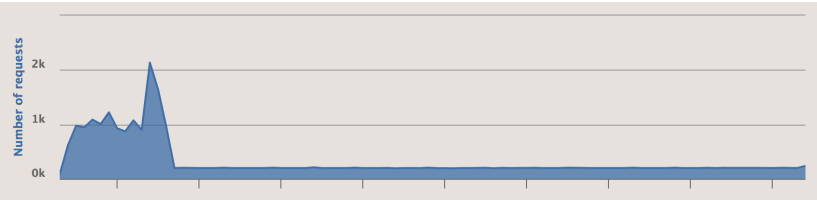

图 5-4 每秒钟响应请求数量

第 6 章 总结与展望

本文设计并实现了一个在线多用户的神经元网络结构编辑分享平台，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，并能分享结构脑图谱，共同探索神经元结构下的奥秘。完成平台搭建后，对平台进行了详细的压力测试，并根据压力测试的结果进行性能调优使之可以支持数千名用户的实时编辑操作。

限于时间关系，平台完成部署后并没有真正投入使用，但是已经与一些神经科学家取得了联系，即将投入神经科学有关结构脑胞体重建和编辑的教学任务中。限于实验室硬件环境尚未完全部署完毕，只有三台计算机可以使用，不能充分发挥分布式结构的优势，在实验室硬件部署完毕后，可以将平台部署在更多的计算机上，充分发挥分布式架构可扩展的特性使之可以支撑更多的用户同时在线。同时结构脑胞体自动重建算法仍然使用的是单机版，脑胞体自动重建算法的并行化已经用 Spark 实现，即将可以在平台中使用，这将进一步提升平台的并行化程度。

参考文献

[1] White J G, Southgate E, Thomson J N, et al. The structure of the nervous system of the nematode Caenorhabditis elegans[J]. Philos Trans R Soc Lond B Biol Sci, 1986, 314(1165) : 1 – 340.

[2] Varshney L R, Chen B L, Paniagua E, et al. Structural properties of the Caenorhabditis elegans neuronal network[J]. PLoS Comput Biol, 2011, 7(2) : e1001066.

[3] Druckmann S, Feng L, Lee B, et al. Structured synaptic connectivity between hippocampal regions[J]. Neuron, 2014, 81(3) : 629 – 640.

[4] Brown K M, Barrionuevo G, Canty A J, et al. The DIADEM data sets: representative light microscopy images of neuronal morphology to advance automation of digital reconstructions[J]. Neuroinformatics, 2011, 9(2-3) : 143 – 157.

[5] 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(2-3) : 305 – 315.

[6] 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).

[7] janelia. dvid[EB/OL]. . `https://github.com/janelia-flyem/dvid/`.

[8] Stonebraker M, Kemnitz G. The POSTGRES next generation database management system[J]. Communications of the ACM, 1991, 34(10) : 78 – 92.

[9] NodeJS. NodeJS Website[EB/OL]. 2017. `https://nodejs.org/en/`.

[10] ExpressJS. ExpressJS Website[EB/OL]. 2017. `https://expressjs.com/`.

[11] Tilkov S, Vinoski S. Node. js: Using JavaScript to build high-performance network programs[J]. IEEE Internet Computing, 2010, 14(6) : 80 – 83.

[12] 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.

[13] Cattell R. Scalable SQL and NoSQL data stores[J]. Acm Sigmod Record, 2011, 39(4) : 12 – 27.