KTH ROYAL INSTITUTE
OF TECHNOLOGY

DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING
First Cycle, 15 Credits

The path to precision: Comparison analysis of automated neural morphology reconstruction software

LOVE LINDGREN
ALICIA LAS HERAS SÁNCHEZ

Stockholm, Sweden 2023

The path to precision: Comparison analysis of automated neural morphology reconstruction software

LOVE LINDGREN
ALICIA LAS HERAS SÁNCHEZ

Degree Project in Computer Science and Engineering, First Cycle
15 credits
Date: July 17, 2023
Supervisor: Alexander Kozlov
Examiner: Pawel Herman
School of Electrical Engineering and Computer Science
Swedish title: Vägen till precision: Jämförelseanalys av automatiserad mjukvara för rekonstruktion av neural morfologi

Abstract

The differences in the shape, form and location of neurons are closely linked to their function. Being able to accurately and efficiently reconstruct neurons digitally in a three-dimensional space is necessary for the acquisition of knowledge in this research field. Automation through software helps optimise efficiency, yet manual reconstructions are often preferred. This thesis therefore aims to help standardise the research field more and facilitate communication and collaborative efforts by evaluating three software, Vaa3D, Neutube and NCTracer, in regards to the reconstruction algorithms' accuracy, efficiency, consistency and user experience with the user interface in order to deduce their advantages and shortcomings. A downloadable and executable Java program, which compares similarities between two reconstructions, and scripts were written to measure these parameters. Vaa3D had higher accuracy and a significantly lower execution time, but Neutube and NCTracer showcased more stability and consistent results. Additionally, NCTracer proved to be more intuitive to use. All software exhibited their own drawbacks, but the information presented can aid in improving the software or the development of new software surpassing prior ones.

Sammanfattning

Skillnader i en neurons form och position är nära kopplat till dess funktion. Förmågan att noggrant och effektivt rekonstruera neuroner digitalt i ett tredimensionellt utrymme är nödvändigt för att erhålla kunskap inom detta forskningsområde. Automatisering via programvara bidrar till effektivisering av detta, men manuella rekonstruktioner är ofta föredragna. Därmed ämnar denna studie att hjälpa standardisera forskningsområdet, underlätta kommunikation och samarbetsinsatser genom att evaluera tre programvaror, Vaa3D, Neutube och NCTracer, i avseende på rekonstruktions-algoritmernas noggrannhet, effektivitet, konsistens och användarupplevelsen av användargränssnittet med avsikt att härleda deras för- och nackdelar. Ett nedladdningsbart och exekverbart Javaprogram som jämför likheter mellan två rekonstruktioner, och skript skrevs för att mäta dessa parametrar. Vaa3D gav en högre noggrannhet och en betydligt lägre exekveringstid, men Neutube och NCTracer hade stabilare och mer konsekventa resultat. Vidare visade NCTracer sig vara mer intuitiv att använda. Alla programvaror uppvisade sina egna brister, men informationen som presenteras kan stödja förbättringen av dessa programvaror eller utvecklingen av nya programvaror som överträffar de äldre.

Acknowledgements

We would like to thank our supervisor, assistant professor Alexander Kozlov at KTH, for his guidance and help along the project. We would also like to extend our gratitude to all the individuals who offered seminars and lectures in the subject; their knowledge and insights have enhanced our learning experience.

Additionally, we would like to acknowledge our family and friends for the constant encouragement and unwavering support. Their understanding has been a constant source of motivation along this whole research.

Division of work

The main objective of this subject is to formulate a question whose answer is relevant to the field of computer science and engineering, so as to conduct a thorough research that mimics situations encountered in a professional working environment, applicable to the real world. As it is commonly done, the approach taken by this subject is to elaborate said research in pairs so as to attain as a side objective the competence of cooperative work. Therefore, the division of work was left to the students to do freely and as it convened them.

I, Alicia Las Heras, elaborated most of the experimental research: I investigated the available software to put under evaluation and selected the most suitable ones for the research question posed, as well as collected the data set and elaborated the actual executions and wrote the necessary code for the analysis. Moreover, I also analysed the results obtained from the experimenting, displayed them in an understandable manner and drew conclusions from them. Regarding the memory, I did everything related to the previously mentioned work (the Method, the Results and the Discussion chapters), as well as the problem statement, scope and approach of the Introduction and all of the Background chapter other than the NeuTube section (2.3.2).

My project partner, Love Lindgren, took on mostly on the writing of the memory: he elaborated both abstracts, the English and its Swedish translation, the introduction and outline of the first chapter, the NeuTube section of the Background chapter and the Conclusion chapter. Moreover, he was responsible for the correction and final amendments of the memory, such as including the feedback received from the peer–reviews and making sure that the writing was adequate and suitable for the engineering and academic standards, as well as the formatting of it by moving it to LaTeX.

Table of contents

Chapter 1

Introduction

The study of the brain dates nearly as far back as the dawn of human civilization itself. The famous Edwin Smith Surgical Papyrus, which contents are speculated to originate from the Third Dynasty during pharaoh Djoser's reign, 27th century BC, describes cases of head trauma among other cases of physical injuries [1]. Its content demonstrates that the early Egyptians, among other medical and anatomical discoveries, recognised the existence of the central nervous system (CNS) and how harm to it can affect the functionality of singular or multiple parts of the human body [1].

After nearly five millennia of empiric practice and innovation, the study concerning the structure and function of the brain and the nervous system [2], referred to as neuroscience, still largely remains a mystery and is widely considered the last frontier of biology [3]. Its vast field of study is based on the assumption that all psychological activity originates from the structure and function of the nervous system [3]. One subdivision in this field of study is neuromorphology, the study of the shape, structure and form of the nervous system.

A neuron, or nerve cell, is responsible for transmitting signals containing information to and from the brain [4], composed by the soma, or cell body, containing the nucleus, the axons, the section of the cell where action potentials are generated and transported, and the dendrites, branched prolongations for the receival of input from the axons and other neurons [5]. Quantifying the shape, form and location of axons and dendrites can allow for the investigation of a neuron's function, but requires precise digitalisation of three-dimensional image morphology, a process known as neural reconstruction.

Neural reconstruction traces the outline of the dendrite and axon from a neuron image, obtained through microscopy, into a format suitable for computational and quantitative analysis. Algorithmic automation promises an increase in both accuracy

and efficiency, but despite the continuous computational advances, fully manual or computed aids are frequently favoured as the method of choice for reconstructions of neural morphology [6].

1.1 Problem statement

> The lack of powerful - and effective - computational tools to automatically reconstruct neuronal arbors has emerged as a major technical bottleneck in neuroscience research.
>
> – DIADEM [7]

Accurate neuromorphological reconstructions require extensive time, resources and human collaboration [6]. The development of computer algorithms for the purpose of automating neuromorphological reconstructions has also proven itself ineffective over the years [7]. Before 2014, the largest neuromorphology database, NeuroMorpho.Org, possessed fewer than 10,000 available reconstructions, where the majority were accredited to manual tracing techniques [8]. Thanks to the advances over the last decade, the number of reconstructions has increased up to over 250,000 cells.

New investment efforts, innovations and contests have recently generated a greater interest in the field's development [7][9]. Consequently, a plethora of new algorithms and software, developed specifically for the purpose of automating neuromorphology reconstructions, have emerged, exhibiting greater performance than previously observed. The aforementioned bottleneck is still ever present and additionally, the quantity of algorithms has generated a lack of standardisation, consistency and cohesion in the field.

"Adherence to open source development principles, adoption of universal data forms, and an organised effort to document and archive existing programs could help alleviate this issue" [7]. The aim of this thesis is therefore to investigate the behaviour of different automated algorithms and to evaluate whether they function as expected by elaborating a comparison analysis on three open-source software, in regards to their efficiency, time– and memory wise, accuracy and consistency of their generated output using the same data set.

The research questions posed in this research and, thus, in seek of answering, are:

- What is the current state of development of the open-source software for neural morphology reconstructions and what advantages and disadvantages do they offer?

- Is the behaviour of the algorithms the same as the one advertised by their descriptions?

1.2 Scope

This research thesis aims to investigate the most appropriate computer algorithm in regards to automated neuromorphology reconstructions. An elaboration of an in-depth state-of-the-art analysis of a descriptive nature for each software's digital reconstruction is not within the scope for this thesis. Rather, the focus was placed on determining whether the software used for the automation of neuromorphology reconstructions, behaved as advertised.

The scope of this thesis is mainly limited by the algorithms and their accessibility. The chosen algorithms are a selection of all available possibilities which aims to draw conclusions up to generalisation. One of the most constraining factors for this project is the budget, given that most algorithms are under commercial licence, these are beyond the scope due to their excessive pricing.

1.3 Approach

In order to address the research question, an empiric search of results is conducted by executing three different software's automatic reconstruction algorithms combined with a collection of test data to generate the neuron reconstructions. The results from each test are evaluated in a quantitative manner and used to compare the algorithms against one another to determine the advantages and disadvantages that each of them present, whether any is superior to the others and which ones could be filtered out when selecting a neuron reconstruction algorithm for scientific research or in the workplace.

The data set is composed of a number of different species collected from the limited offer of neurons from DIADEM [7]. The data set consists of neurons of varying sizes and complexity so as to be able to properly evaluate the correctness and deviations that may present themselves in the algorithms.

The selected algorithms, which are shown in Table 1.1, will be elaborated upon in chapter 2. In this study, the evaluation of these software's respective automatic reconstruction algorithm is based on the following three quantitative parameters and one qualitative that is further elaborated upon in chapter 3:

- **Accuracy.** Evaluation of the algorithms' correctness and precision variance, depending on the neuron complexity and size. For easier verification, the au-

Software	Description
Neutube	Open-source software for reconstruction, analysis and simulation of neural networks [10].
Vaa3D	Open-source software for analysis, manipulation and visualisation of 3D biological images and objects [11].
NCTracer	Open-source software for automated and manual tracing of neurites from single 3D stacks [12].

Table 1.1: Selection of software tools to be evaluated.

tomatically generated neuron reconstruction output is compared with a gold standard (GS) reconstruction [7].

- **Efficiency.** Evaluation of the algorithm's execution time and usage of computer resources, CPU– and memory–wise, in regards to neuron complexity and size.
- **Consistency.** Assessment of the program execution's resulting variance and reproducibility.
- **Ease of use.** The software will be appraised with a subjective and qualitative approach regarding the ease of installation, usability of the tools, and accessibility and intuitiveness for beginners.

1.4 Outline

Firstly, the Background is approached, which starts with the initial definition of necessary concepts for the comprehension of essential vocabulary, followed by a description of the available software on the market, as well as an explanation and motivation behind the selected algorithms for the analysis. Finally, relevant prior research strongly related to this thesis is commented on.

The Method chapter contains the description of the methodology applied to conduct the experiments in order to answer the question, and the tools and frameworks utilised.

Finally, the results section of the research that compiles the outcome of the experiments, followed by a discussion chapter, interpreting and summarising the most pertinent results and recommendations for future studies, and a final conclusion segment.

Chapter 2

Background

2.1 Terminology

A. Neuromorphology
The study of the shape, form and structure of the nervous system, commonly used for identification and classification given the differences in the dendritic and axonal shapes are thus closely linked to a neuron's functions [13]. In particular, axonal and dendritic arbours are key functional components of neural processing and fundamental determinants of neural circuits [14].

B. Neuron
The fundamental units of the brain and nervous system, responsible for receiving sensory input from the external world, sending motor commands to our muscles, and transforming and relaying the electrical signals at every step in between [5]. Neurons are known to be a subset of brain cells, so as to say, only around 10% of the total cells in the brain are neurons. Neurons are composed of three parts: dendrites, axons and soma.

C. Dendrite
Known in the scientific community for being short and branched plasmatic prolongations of the brain cell in charge of the receival of stimuli. In other words, the nerve ending of the brain cells whose main objective is to act as receptors of nerve impulses.

D. Axon
Fibre that extends from the brain cell and is responsible for the transportation of electrical impulses to other neurons. So as to say, they transmit the synapses between neurons.

E. Soma
The soma or cell body, refers to the bulbous portion of the neuron, containing the cell nucleus.

F. Terminal
Also known as axon terminal, synaptic bouton or terminal bouton, refers to the most distal portion of the neuron. These enlarged axon endings, often club or button shaped, are critical for neural communication given that they enable neurons to release a chemical known as neurotransmitters to target cells.

G. Gold standard reconstruction
Manually generated reconstruction with minimal subjective choices and biases in its creation, usually provided with the neuron's image stack [7]. Often referred to as ground truth.

H. SWC file format
Standardised file format used to store neural morphology reconstructions, composed of header entries, containing the metadata and information about the overall structure represented, and the data entries, a representation of the resulting reconstruction structure in a tree-like set of nodes where each node corresponding to a point in the structure is defined by a its three-dimensional coordinates in the XY plane and Z axis, a radius value and its parent connection [15].

2.2 Available algorithms

In order to facilitate the task of researchers, or any other users, of selecting an appropriate tool suitable for their necessities, Table 2.1 compiles the existing algorithms used in recent years for the reconstruction of neural morphologies, based on different research papers and other sources [6][16], including the developer, year of release, degree of automation, distribution licence, availability for obtention and a research paper describing the tool. Only software that allow either automated or semi-automated reconstruction has been included given that manual reconstruction falls outside the scope of this thesis.

Name	Developer	Year	Degree of automation	Licence	Available	Reference
Amira Module	ThermoFisher Scientific	1999	S, A	C	Y	Stalling et al., 2007
Autoneuron (NeuroLucida 360)	MBF BioScience	2015	A	C	Y	Glaser & Glaser, 2004
Farsight Toolkit	Rosyam Lab	2012	A	O	Y*	Bjornsson et al., 2008
HCA Vision	CSIRO	2005	A	F	N	Valloton et al., 2007
Filament tracing, Imaris	Oxford Instruments	2013	S, A	C	Y	The University of Queensland, 2022
Multineurite	Harvard Medical School	2006	A	F	N	Xiong G. et al., 2006
NeuronJ	Biomedical Imaging Group	2004	S	O	Y	Meijering et al., 2004
NeuronStudio	CNIC	2006	S, A	F	N	Douglas Ehlenberger., 2009
Neuromantic	Darren Myatt	2012	S	F	Y	Myatt et al., 2012
Neuromorph	Bio Electron Microcopy Laboratory	2015	S	F	Y	Jorstad A. et al., 2015
Neuritetracer	Fournier Lab	2008	S	F	Y	Pool M. et al., 2008
NCTracer	Neurogeometry Lab	2014	S, A	F	Y	Neurogeometry Lab, 2014
NeuTube	Feng Lab	2015	S, A	O	Y	Feng et al., 2015
SNT	Howard Hughes Medical Institute	2020	S	O	Y	Ferreira et al., 2021
Vaa3D	Howard Hughes Medical Institute	2007	S, A	O	Y	Peng et al., 2014
Degree of automatisation: A = Automated, S = Semi-automated; Licence: C = Commercial use, F = Free, O = Open-source Available: Y = Yes, N = No, *Only source code						

Table 2.1: Description of the algorithms, including the developer laboratory, year of release, degree of automatisation, distribution licence, availability for obtention and a research paper describing the tool.

2.3 Evaluated software

This section presents the different software selected for the evaluation of their features and behaviour. The selection of these tools is based on their open-source nature, taking into consideration the excessive prices of commercial licences, and their availability, since most of the options were either too old and not obtainable anymore or too outdated. The main distinction among these software lies in their neural reconstruction method, that will be explained further in this section.

The main purpose of these tools is morphology reconstruction, a process that consists of tracing neuronal structures by following the branching patterns of neurons and reconstructing their three-dimensional shape; essentially like tracing the outline of a drawing with a marker. This process is akin to unraveling a complex network of interconnected networks, similar to tracing the intricate threads of a tapestry. By mapping these pathways, clearer understanding on the flow of information along the brain's neural fabric can be gained.

2.3.1 Vaa3D

3D Visualization-Assisted Analysis, Vaa3D for short, is a cross-platform open-source software for the visualisation, analysis and manipulation of Bioimages and Surface Objects such as 3D image stacks, developed and maintained by Howard Hughes Medical Institute – Janelia research campus – and the Allen Institute for Brain Science. This software contains powerful modules for image analysis, such as cell segmentation, neuron tracing or quantitative measurement, and data management.

Initially, Vaa3D computed the reconstructions by using a graph–augmented deformable model (GD), a method that consisted in obtaining the optimal solution of the geodesic shortest path problem, defined as the combination of Euclidean distances and closeness to local centres of image intensity distribution. So as to achieve this, it first executed a shortest path algorithm to find the optimal solution and then optimised a discrete deformable curve model to obtain a visually satisfactory reconstructions [17].

However, in order to achieve the fully automated neuromorphology reconstructions, the latter Vaa3D release implements an All-path pruning method (APP) [17]. A Complete reconstruction refers to the state of a reconstruction in which all visible regions in the image are covered; an Over-complete reconstruction, on the other hand, is a complete reconstruction in which some of the structural components, such as a neuron branch or a node, are redundantly covered. The APP method consists of two phases: the first one, producing an initial over-complete reconstruction (ICR), and the second one, pruning the reconstruction by designing a maximal-covering

minimal-redundant (MCMR) graph so as to remove these redundancies.

The ICR step takes a 3D neuron image and a seed-location that can be detected automatically. To produce the ICR of the neuron, the average intensity value (t_a) as the threshold for the image foreground. In other words, any part of the image whose intensity value is over ta is considered to belong to the neuron structure. Then, an undirected weighted graph, $G = (V, E)$, where the vertices (V) are the neuron's voxels is created. An edge connects two vertices if and only if they are spatial neighbours and a weight is assigned according to the following formula:

$$\begin{aligned} e(v_0, v_1) &= \|v_0 - v_1\|(\frac{g_1(v_0) + g_1(v_1)}{2}), \\ g_1(p) &= e^{(\lambda_1(1-I(p)/I_{max})^2)} \end{aligned} \tag{2.1}$$

The first term is the euclidean distance between two vertices and the equation 2.2 constraints that the edge weight between bright voxels has a smaller value than between dark voxels.

Finally, in order to create a shortest path map, a Dijkstra algorithm [18] is used to find the shortest path between the seed and each of the vertices belonging to G. All the paths are organised into a tree graph solution that composes the ICR.

Following, there is the MCMR step which is divided into three phases: Dark-leaf pruning (DLP), Covered-leaf pruning (CLP), and Inter-node pruning (INP). DLP consists in redefining the previous threshold, t_a, to the lowest visible threshold, t_v, to iteratively remove all leaf nodes whose intensity is below t_v. Next, CLP. Defining a radius-adjustable sphere centred and enlarging it gradually until 0.1% of the image voxels are darker than ta, it allows for the location of structural components significantly covered by others. A component a is said to be covered by b if they satisfy:

$$\frac{\Omega(a)}{\Omega(a)} \geq 0.9 \tag{2.2}$$

Where Ω represents the occupied volume by the component. If no node covers any leaf node, said leaf node has to be kept, however, if a leaf node is covered by one or more nodes, it can be safely pruned. By following this iterative method the tree can be pruned of any redundancies.

Finally, to reduce the complexity of the neuron reconstruction, the redundant inter-nodes which connect leaf nodes to branching nodes or the root are removed. Starting from a leaf node a whose parent node is b, if b is significantly covered according to equation 2.3 below, then node b, which is an inter-node, is removed and a is assigned as a parent b's original parent. If b is not to be removed, its parent, node

c, is checked on whether it is pruned or not based on the coverage relation with b. For each leaf node, this iterating process is conducted until a branching point or the root are reached.

$$\frac{\Omega(a)}{\Omega(a)} \geq 0.1 \tag{2.3}$$

The threshold used for this step is much lower than the applied during CLP phase so as to prune the lowest number of leaf nodes and the highest possible number of inter-nodes.

2.3.2 Neutube

NeuTube is an open–source software for reconstructing neurons from fluorescence microscope images [10]. This GUI application's framework is designed with the SWC file format in mind. The software can take a raw image stack as input from which the user can generate the neuron reconstruction through tracing, either manually or automatically. Additionally, it can also import SWC files to allow for the manipulation of reconstructions The generated neuron reconstruction can thereafter be saved and exported as a SWC file [16].

It possesses an engine consisting of four core modules: 2D visualisation, 3D visualisation, neuron structure manipulation and image analysis. Image analysis provides functions for tracing neurons or neuron branches to deduce its structure, either manually or automatically to allow for minimal user interaction [16].

Previously, this algorithm utilised a cylindrical filter, because in cross–section a neuron fibre would look like a Gaussian–diffused spot and therefore utilises the Laplacian of Gaussian (LoG) [19]. In the past, the unit vector was expressed by equation 2.4, where $z \in [-\frac{h}{2}, \frac{h}{2}]$, and h is used to reflect the length of the cylinder.

$$U(x, y, z) = (1 - (x^2 + y^2))e^{-(x^2+y^2)} \tag{2.4}$$

After retrieving the cylinder, it is scaled and rotated to optimise the tracing's score. This is accomplished through the Polak–Ribière conjugate gradient descent method [20]. The score, S, is thereafter numerically estimated due to a lack of closed form expression. The following approximation equation, equation 2.5, is the formula expressing the partial derivative where u represents the position along the reconstructed fibre, used as follows:

$$\frac{\delta S}{\delta u} = \frac{S(u + \Delta u) - S(u - \Delta u)}{2\Delta u} \tag{2.5}$$

Afterwards this step is iterated along the cylinder's central axis, with step size $\frac{h}{2}$, where the process is repeated until it reaches the end of the neuron. This form

of tracing only ever applies to an image stack containing one neuron; if more neurons are present in the image, one has to determine whether the cylinders adhere to the same neuron or another one through a pixel threshold which detects the neurons' seed and fibre based on pixel intensity and examination of the eigenvalues in a Hessian matrix at scale (3x3x3). The threshold is computed by the triangle method over a histogram containing the local maxima of these two feature scores for all image pixels. A pixel which scores above the threshold for either of these features is recognised as belonging to a neuron, upon which the seed location is placed.

The seed locations are optimised by initialising the pixel with the best score. The seeds are sorted based on their optimised model scores and traced, where any possible seed location that is covered by the tracing algorithm is removed from consideration for the following neurons. The process is repeated until all remaining seed locations are exhausted.

Once all fibres have been identified, they can be assembled into a tree structure which prohibits the formation of cycles. In order to determine how all neurite fibres are connected, a distance threshold of 20 pixels is established to rule out neurite fibres that overextend this distance and instead consider these as belonging to different neurons. If a fibre is within this threshold, two measurements are used to verify that they are connected. The first one, tests whether a path can be found from one end to the other that is highlighted by bright pixels; the second measurement is calculating the geodesic distance between points x_0 and x_1:

$$geo_g(x_0, x_1) = \min_{c \in \{all\, possible\, paths\}} \int_c g[I(c(t))] \|dc(t)\| \tag{2.6}$$

Where $c(0) = x_0$, $c(1) = x_1$, $dc(t) = dc_x^2(t) + dc_y^2(t) + dc_z^2(t)$ represents the differential of the arc length of $c(t)$, $I(t)$ is the image intensity at point x and g is a function used to define how the geodesic distance is dependent on the image intensity.

The accuracy of the tracing algorithm is highly dependent on its ability to accurately separate the foreground and background [20]. The shortest path c is expected to be on the foreground and can be expressed with the following sigmoid function:

$$g(x) = \frac{1}{1 + e^{\frac{x-\alpha}{\beta}}} \tag{2.7}$$

Where α and β are used to reflect this separability. By using the signal the neuron is projecting as foreground and the surrounding as background, and calculate their average values, denoted as c_f and c_b.

NeuTube's algorithm is stated to improve upon this algorithm by replacing the aforementioned cylindrical filter with the SWC framework in order to construct the

nodes in a neuron tree structure [16]. The model can be represented as a set of spherical nodes with the coordinates (x_i, y_i, z_i) and radius r_i. no represents the neuron structure's root and n_j is referred to as the parent of n_i. For more details on how the SWC framework defines its three layers of operation to ensure its structure validity, read [16].

$$S = n_i = (x_i, y_i, z_i, r_i, n_i)|i = 1, ..., N, j = 0, ..., N, i \neq j, x_i, y_i, z_i, r_i, n_i \in R \tag{2.8}$$

To ensure that the software can reconstruct neurons from raw image signals, I, and parameter functions specified by the user input, Θ, the following function was created:

$$g(S_1|\Theta, I) = S_2 \tag{2.9}$$

While this creates a superfamily of operations, the only noteworthy one is for creating the shortest geodesic path, similar to Vaa3D, but modified for the SWC framework. Using the aforementioned function, equation 2.9, $S_1 = \{n_i, n_j\}$ defines the source and target node and $S_2 = \{n_i, n_1^{'}, n_k^{'}, n_j\}$ is the resulting path.

2.3.3 NCTracer

Neural Circuit Tracer, also known as NCTracer, for short, is an open-source software developed by the Neurogeometry laboratory of Northeastern University, which aims to automate the process of three-dimensional reconstructions of neurites at a larger scale.

The method utilised by this software falls under the category of Image segmentation algorithms, which consists of partitioning the image into its constituent components [21], in particular, delineating what voxels of the image belong to the neurite in question and which belong to the background [22].

The algorithm consists of several steps. Firstly, the pre-processing, if necessary, of the image stack, a process which may include alignment of the images, stack deconvolution, colour–conversion or elimination of noise, such as unwanted cell-bodies or other non-neurite structures. Secondly, the image is filtered using a multi-scale centre surround filter (CSF), the LoG, so as to enhance the linear structures for the tracing process:

$$\begin{aligned} CSF(\vec{r}|\sigma) &= \frac{e^{\frac{\|\vec{r^2}\|}{2\sigma^2}}}{(2\sigma^2)^{3/2}}(1 - \frac{\|\vec{r}\|^2}{3\sigma^2}) \\ O(\vec{r}) &= \max(CSF(\vec{r}|\sigma) \cdot I(\vec{r})) \end{aligned} \tag{2.10}$$

When σ, that represents the size of the filter, matches the calibre of neurites, it can smooth out the intensity within the boundaries of the neurites, sharpen the boundaries, and reduce the background noise [22]. If the image stack contains different sized neurites, the fixed σ of LoG would only encompass a reduced range of calibres. To circumvent this problem, the output of different filters are combined and applied to the image by taking the maximum output at every voxel as seen in equation 2.10.

Following, the initial trace is computed by means of the Fast Marching Method based on the solution of the Eikonal boundary value problem [23], seen in equation 2.11.

$$\begin{gathered} |\nabla T(r)| I(r) = 1 \\ T(\partial S) = 0 \end{gathered} \tag{2.11}$$

Where r represents a position in the processed image, I denotes the normalized value of the intensity, therefore ranging between 0 and 1. ∂S is the boundary from which the light rays originate, resulting in an arrival time of 0 at the boundary. ∇ is the gradient operator. $T(r)$ represents the time map, which contains information about the shortest time of arrival of the light rays from the boundary to different structures of the neuron. Since higher speeds of light propagation correspond to higher intensities, the arrival time will be smaller along the high-intensity structures of the neuron [24].

Firstly, the boundaries or seed points are marked automatically based on the image intensity along the structure of the neuron. Then, the arrival time front is allowed to travel a certain distance, D_{max}, which has to be larger than the calibre of neurites so as to not produce short incorrect branches and not much larger than the shortest branch to be resolved by the algorithm. The path connecting from the seed to the furthest point of the front is found by performing gradient descent on $T(i, j, k)$. Following, the path is added to the boundary and the Fast Marching algorithm is re-initialised from the new boundary. This iterative process continues until the intensity of the final added branch is below the threshold, which is 20% of the average intensity of the trace, or the maximum number of steps set by the user has been reached. The last boundary is considered to be the initial trace of the neuron.

The $T = 0$ boundaries are guaranteed to collide if they are connected by paths with higher intensity than the background, although high levels of background intensity can cause erroneous collisions. These errors can be corrected during the optimisation phase.

The optimisation of the initial trace is essential for the final step, the merge of individual branches, defined as a neurite connecting the root or a branch-point to a successive branch, into tree structures, based on their orientation, intensity and cur-

vature, since the calculations for these parameters strongly relies on the smoothness of the tracing. This final step is the most important stage for the automated tracing process. Firstly, the identification of the end-points of all branches to group them into spatially segregated clusters. For this, a graph is constructed, where nodes are the end–points and the edges represent the connections between neighbouring nodes, whose distance is lower than the threshold set by the user. The algorithm proceeds by merging two end–point clusters, to three end-point clusters, to the higher order clusters. For each cluster, all possible merger scenarios are considered, along with their cost, according to the formula in equation 2.12. The algorithm performs the lowest cost merger, unless it results in a loop, which then attempts the subsequent lowest cost merger.

$$Cost = \sum_i (\alpha_i D_i + \beta_i \sum_j |\cos(\chi_{i0}) - \cos(\chi_{ij})| + \Upsilon_i (1 - \frac{I_i}{I_0}) + \delta_i K_i) \quad (2.12)$$

Where i represents every different branch end-point merger within a single merge scenario. D_i represents the distance between all end-point pairs and ij denotes the angle formed by the branch j in the i-th merger. By default, i_0 is $\frac{360}{n}$ degrees for n branch end-point mergers. I_0 represents the average intensity of the image while I_i is the average intensity along the trace. K_i denotes the curvature of the optimally connected traces of merger i, calculated only for the intermediate vertices and N represents the number of branch end-points unmerged. Finally, the parameters α, β, δ, and ϵ which are determined by the learning algorithm during training. There are two available trained models, $L6$ and OP.

2.4 The DIADEM Metric

Along with the datasets and GS reconstructions, a downloadable and executable Java program is provided, which compares a generated reconstruction with its corresponding GS by assigning a percentual score based on determining their topological similarities [25].

The evaluation of an arbour in regard to its match in the GS reconstruction is done by registering and scoring each individual composing node and its parent branch individually according to their connectivity with their belonging region of the tree. Each node in the GS is registered to a node from the automated reconstruction, which requires the aforementioned node to be located within a cylindrical spatial distance that represents the region in which a bifurcation might be created taking into consideration the possible error added by resolution in the XY plane and Z

axis. Starting at the most proximal bifurcation, the registering process proceeds to the first's bifurcation children and suchlike until all nodes are matched.

Registered nodes are expected to have matching paths to matching ancestor nodes, located by traversing the tree upwards, toward the root, from the GS node and the potential match from the automated reconstruction until ancestors of either trees are within the accepted distance threshold of each other. Once an ancestor is found, the path test is performed in order to confirm or discard the candidate matching node.

A test path is considered acceptable if the error along a component, the XY plane or the Z axis, relative to the full path length is low enough. The error is computed as the difference in one component between the GS and potential test path divided by the full GS length path. The threshold for the acceptable error is set based on the straightest possible path and the longest possible path that stays in-bounds of the neurite. In order to adjust the test path length to take into consideration the difference in position between ancestor and descendant, the test path is shortened by the distance between the descendant and the trajectory point minus the GS node distance from the trajectory point. The trajectory point stands for the spot of the GS reconstruction at a threshold distance of the registered node from the GS reconstruction. If multiple nodes from the reconstruction are potential matches, the spatially closest node is registered.

After registration, the nodes that may have been unable to be matched to a node from the automated reconstruction are taken into a second round of consideration. The matching path is found by traversing the GS reconstruction toward the root from the target node. For every ancestor that has been matched to a node from the automated reconstruction, the GS tree is traversed from the respective node towards the terminal ends until a descendant node is within a threshold distance of a node from the automated reconstruction and fulfils the matching criteria previously described. If no descendant nodes from the automated reconstruction match the path between the aforementioned ancestor and the two GS descendants, further down nodes are checked for a match. The process is continued until a path match is found or until all potential descendants are exhausted. When a path is found, the reconstruction is considered to be correct on that node although it lacks a direct match in the GS given there exists a test path that captures the same path up to the node location.

2.5 Related works

Duncan and Ascoli (2011) [6] elaborated an analysis of descriptive nature of the state-of-the-art of automated and semi–automated neural reconstructions. An in-depth approach to the algorithms and tools available at that moment is presented, accompanied by a description of their most significant and relevant traits. The most

remarkable technologies for automated reconstructions are thoroughly described as well as their existing issues and other considerations regarding tissue preparation and imaging methods of the cells. Moreover, attention is brought on the validation methods for reconstructions taking into consideration a GS and the criteria available for concluding the accuracy of the digitalisation.

The authors express their remarks on the trajectory of the field and how they could be improved by substituting manual error correction by basing reconstructions on previously reconstructed neurons of the same class so as to speed up the process.

Feng, Zhao and Kim (2015) [16] elaborated an in-depth analysis of their recently developed and launched tool, NeuTube, by comparing it to two other powerful neuron reconstruction software programs, one commercial, Neuromantic, and one free licensed, NeuronStudio, selected due to their similarities in design and overall features. Based on the DIADEM Metric [25], the accuracy of the tracing was assessed by how well the critical areas, such as branching points, were semi-manually traced in relation to the GS reconstruction. The conclusion drawn showed the superiority of NeuTube over the other two software, particularly in accuracy.

Chapter 3

Method

For the purpose of obtaining information and addressing the formulated questions, a dataset was collected specifically for conducting an analysis of the software under consideration. The execution of the test experiments were conducted on a HP Pavilion x360 Convertible 14-dw0xxx portable computer with 8 gigabytes of RAM using Windows 10 as the operative system.

3.1 Dataset

This research used three different datasets. Each dataset is formed by a collection of three-dimensional images of neural structures from numerous nervous system regions. A concise and structured outline of the main characteristics of each dataset is presented below in Table 3.1, including the size of the set, the species extracted from and the region they belong to. Each neuron structure is accompanied by its respective GS reconstruction.

A more in-depth approach on their content is discussed in the following sections.

Dataset	Size	Species	Nervous system region
Cerebellar Climbing Fibres	2	Rat	Cerebellar Cortex
Neocortical Layer 1 Axon	16	Mouse	Neocortical layer 1
Olfactory Projection Fibres	9	Mouse	Olfactory Bulb

Table 3.1: Data sets used, the number of neurons contained, the species and nervous system region they belong to.

3.1.1 Cerebellar Climbing Fibres

Climbing fibres are axon terminals that belong to the inferior olive neurons responsible for providing excitatory input to Purkinje cells of the cerebellar cortex [26]. The Cerebellar Climbing Fibres (CF) dataset was originally collected to analyse the morphological properties of these climbing fibres and anatomical relationships from the axonal origin in the inferior olive through the terminations in the cerebellar cortex [27].

The dataset consists of two image stacks of the corresponding neural structures located in the Cerebellar Cortex of a rat, and their respective manually traced digital reconstructions, owned by Giorgio A. Ascoli, from the Molecular Neuroscience Department in the Krasnow Institute for Advanced Study of George Mason University.

Transmitted Light Brightfield method is applied for image acquisition. A portion of neural structures are labelled, using Biotinylated Dextran Amine, then mounted on microscope slides and viewed using a digital camera with 100x magnification lens, 2.0x zoom factor and immersion oil placed between the objective lens and the slide to increase the numerical aperture up to 1.3 and improve the resolution of the image by allowing more light to be captured and focused on the sample (100x oil (NA=1.3) with 2.0x zoom). Every image stack contains only one axonal arbour to be traced.

3.1.2 Neocortical Layer 1 Axon

"Axons arborize extensively in Layer 1 of the neocortex and form a remarkably dense and poorly characterised network" [26]. two-photon lasers scanning microscopy in vivo allow for an effective labelling and visualisation of the axon network using a microscope equipped with a photomultiplier tube with a 40x magnifying objective lens. Motivated by the study of cellular and circuit mechanisms of experience-dependent plasticity and repair in the mammalian neocortex [28], this dataset is owned by the researchers at MRC Clinical Sciences Center, Imperial College London that conducted the study.

This dataset is composed of two different subsets: *Subset 1*, consisting of 6 image stacks, and *Subset 2*, with 10 image stacks, each of them containing numerous axonal trees to be traced. Every image stack within a subset represents a tile for a mosaic.

The reconstruction of the neuron structures was executed on the separate tiles as well as on the full mosaic so as to evaluate any differences in accuracy and efficiency. As mentioned, each image stack contains several axonal trees. Each axonal tree is reconstructed in a different file: the first dataset contains 33 GS reconstructions, named from 1 to 33, and the second, 22, from A to U. Therefore, in order for a proper evaluation of the reconstructions to be conducted, it was necessary to relate

the corresponding GS reconstructions to their respective image stacks. An overview of the matching reconstructions of the stacks is presented in table 3.2.

Subset 1		Subset 2	
Image stack	**GS Reconstruction**	**Image stack**	**GS Reconstruction**
1	3, 4, 5, 7, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21	1	A, K, M, Q, U
2	16, 19, 20, 21, 25, 26, 28, 33	2	K, Q, U
3	18, 19, 21, 26, 33, 34	3	A, G, J, L, O, P, S, T
4	8, 9, 22, 23, 27, 29, 31, 33	4	A, B, C, G, I, J, L
5	2, 3, 4, 6, 7, 8, 9, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32	5	A, B, C, D, E, F, I, J
6	1, 2, 3, 4, 5, 8, 9, 11, 20	6	A, C, D, E, F
		7	A, B, E, G, I
		8	A, B, C, E, I, J, M, N, R
		9	A, M, Q, S
		10	A, C, H

Table 3.2: Image stacks belonging to every subset and their respective GS reconstructions.

3.1.3 Olfactory Projection Fibres

Olfactory projection fibres (OPF) are a type of white matter tract that relay olfactory information to the mushroom body and lateral horn [29] and connect the olfactory bulb with the cortex where the odour signals are processed and integrated [30][31]. Originated from two studies of the organisation of the central olfactory system in

Drosophila [29][32], this dataset is owned by the Department of Zoology of the University of Cambridge and the Department of Biology of the Stanford University.

Mosaic Analysis with Repressible Cell Marker (MARCM) method [33] allowed the generation of many brains containing single Green Fluorescent Protein (GFP) labelled neurons. The recollection of the stacks was executed using two-channel confocal microscopy method: the slides were viewed through a microscope with a photomultiplier tube with a 40x oil (NA=1.3) objective lens with 1.5x zoom.

Nine different drosophila olfactory axonal projection image stacks and their corresponding GS reconstruction are contained in the dataset. Each image stack only contains one single arbour to be traced.

3.2 Accuracy

Neuron complexity, size and noisy data present in each image stack can perchance influence the resulting reconstructions and therefore the accuracy and precision variance of the selected algorithms. In order to evaluate the accuracy, the automatically generated neuron reconstruction output is compared with a GS reconstruction [7].

Employing the DIADEM Metric, each of the reconstructions underwent an accuracy test by matching every GS reconstruction to one of the trees in the algorithms' reconstruction. The ground truth reconstructions are composed by a single tree au contraire to the output reconstructions which have a tendency to be fragmented, containing two or more trees. So as to achieve a proper assessment, considering that the Neocortical Layer 1 Axon (NL1AX, X being either 1 or 2 depending on the subset the image stack belongs to) dataset's reconstructions were a combination of multiple GS reconstructions and the output reconstructions were all combined into one file, whether in one or multiple trees, a script was implemented so as to compare each tree to their respective GS, available in Appendix A.

The script utilises command 3.1, provided by the DIADEM Metric in order to compare the reconstructions, which include three obligatory parameters. The first parameter, G, requires the input of a GS reconstruction in SWC format. Parameter T, indicates and requires the input of the automated reconstruction under evaluation, also in SWC format. The final obligatory parameter, D, configures the settings on which dataset the files are belonging to, which sets the distance and the path length error thresholds.

java -jar DiademMetric.jar -G [InitialRecontructionFile]
-T [GoldStandardReconstructionFile] -D [dataset_number] (3.1)

Once a result was obtained for each GS reconstruction belonging to that image stack, the final accuracy score for that reconstruction was computed by calculating the average of all the partial values. The accuracy for this dataset was computed additionally with image stacks combined into a mosaic by, likewise, comparing each tree of the output file to every ground truth reconstruction and computing the average of the partial scores.

Similar behaviour was applied to the OPF and CF datasets by considering negligible small short trees, a consequence of the noise in the image stacks during the reconstruction stage.

3.3 Efficiency

The lack of efficiency is the seedbed of the bottleneck in this field. This aspect of the algorithms is, therefore, an essential metric to be assessed. Unequivocally, the complexity and size of the reconstructed neuron affect the performance and resources required by the program to execute.

3.3.1 Execution time

The execution time, often referred to as CPU time, is a term used to describe the total amount of time a process executes [34].

With the aim of studying the efficiency of the reconstruction of the image stacks in the time dimension the tool Process Explorer [35] was utilised given it provides the detailed time for which a process has been executing. Particularly, the CPU User Time which refers to the CPU time consumed by the user-mode code of the process such as specific calculations, computations or processing tasks, the time the CPU has spent executing user-level operations. This measure allows the obtention of the time the reconstructions have taken, not considering the time spent in operating system activities or managing system resources.

3.3.2 Memory usage

Memory usage refers to the quantity of system memory (RAM) used by a program. The utilisation of more RAM than the system is capable of providing can compel a significant negative impact on its overall program performance. Frequent memory allocation and deallocation can be inefficient, memory fragmentation (scattered and unused memory) can encumber the allocation of large blocks of memory and swapping a portion of the resources to the disk storage can be much more time-consuming than accessing it in memory [36].

The analysis of this metric was performed through the implementation of a script, available in Appendix B, which polls every 125 milliseconds the value of memory the process is making use of, and exports it as a percentage to a CSV file accompanied by the timestamp the value was polled at. Knowing the timestamp at which the execution of the reconstruction starts and ends, the average memory employed by the reconstruction is computed.

3.3.3 CPU usage

The Central Processing Unit, usually referred to by its acronym, CPU, is the component of a computer responsible for performing the majority of the processing operations [36]. CPU usage is the measure that reflects the total percentage of resources being utilised by the execution of a program and the processing of its data.

This parameter was assessed through the execution of a script, the same utilised for the computation of memory usage available in Appendix B, which exports to a CSV file the value of CPU consumed by the execution of the reconstruction polled every 125 milliseconds, accompanied by the timestamp of the polling. Similarly, knowing the timestamp at which the automated reconstruction starts and ends, the average of all the CPU usage values obtained during the process is computed.

3.4 Consistency

Reliability and uniformity in the result reconstructions provided by the algorithms is a valuable quality. The ability to yield outputs with minimal deviation is a crucial behaviour for enhancing the trustworthiness and usability of the algorithms by heightening the confidence in the solidity, predictability and replicability of the obtained outcomes.

To evaluate this property of the software, an image stack was randomly selected from every dataset to be reconstructed a total of five times. Random selection granted representativeness by ensuring every image stack had an equal chance of being included, and an unbiased evaluation for a more objective assessment. Each output was saved to undergo a comparison with the initial reconstruction, produced during the efficiency assessment, by running a script that examined whether the files were identical to the initial reconstruction. Upon discovery of a file with different values, a new comparison was elaborated making use of the DIADEM Metric which provided a score for how similar the two reconstructions were executing the same command used for efficiency, command 3.1, by replacing the GS with the initial automated reconstruction.

3.5 Ease of use

It is relevant to acknowledge that the qualitative features of a software program are as relevant as the quantitative aspects. The software underwent an analysis on its usability, in particular, from a standpoint of evaluation of beginners in the field of neuromorphology and generating reconstructions.

The criteria of evaluation were sorted into three categories:

- **General.**
  Assessment of the features related to the interface and easy interaction and manipulation of the image stacks and reconstructions, such as the possibility of undoing and redoing recent actions or the presence and configuration of hotkeys and macros. Additionally, it was also taken into consideration the documentation and user manuals offered by the developers of the software on the functioning of the tool.

- **Reconstructions.**
  Evaluation of the features regarding the reconstruction and its files. The parameters assessed were the ability of 2D and 3D visualisation of the reconstructions, the possibility of editing the reconstruction, particularly, translating its coordinates along the axis, and whether the representation is colour coded so as to facilitate the comprehension of the type of nodes displayed.

- **Image stacks.**
  Evaluation of the capabilities of the program related exclusively to the original neuron image, namely 2D and 3D visualisation of the image stack and processing of the image such as geometric transformations.

Chapter 4

Results

The following chapter presents the result obtained from the execution of multiple reconstructions with NeuTube Vaa3D and NCTracer. Divided into four sections, the first one describes the results accumulated from the experiments related to the accuracy of the reconstructions. The second part concerns the findings related to the efficiency of the algorithms. The third section describes the results regarding the consistency of the outcomes and the final and fourth part addresses the ease of use of the different tools.

This chapter uses previously defined abbreviations to refer to the different image stacks belonging to each dataset. The digit after each name identifies the image stack inside every dataset.

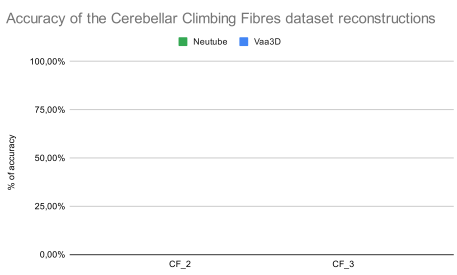
4.1 Accuracy

After performing an evaluation reconstructions' accuracy according to each software, all results were collected and presented in figure 4.1. The accuracy is expressed as a percentage, being 100% the most accurate reconstruction possible. It is important to highlight the fact that, to facilitate the result visualisation, the scale of figure 4.1.c differs from the usual by ranging from 0 to 10 instead of 0 to 100.

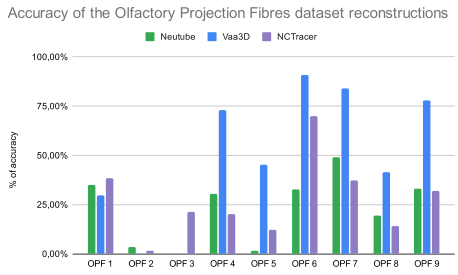
NCTracer is not included in figure 4.1.a in consideration of the fact the accuracy score is not 0% but null, being that it failed to execute any reconstructions due to the exorbitant sizes of the image stacks' file, resulting in a NullPointerException.

To study the effect the preprocessing of the image stacks had on the reconstructions, figure 4.2 summarises the OPF reconstructions' accuracy scores, elaborated with NeuTube with and without preprocessing.

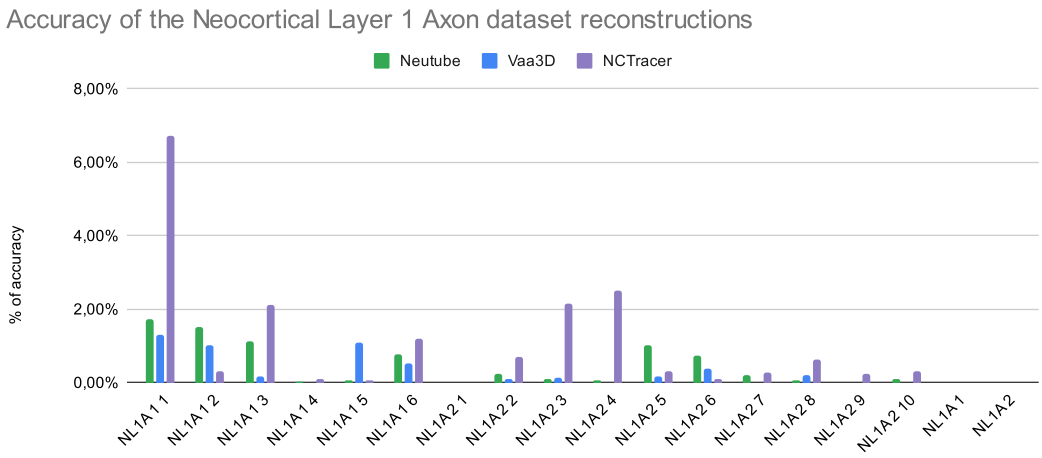
The corresponding numerical data supporting the results in these figures can be found in Appendix C.



(a)



(b)



(c)

Figure 4.1: Accuracy scores of the reconstructions for (a) CF dataset, (b) NL1AX dataset, and (c) OPF dataset. On the X-Axis, the image stack under evaluation. On the Y-Axis, the reconstruction's percentage of accuracy according to the DIADEM Metric.

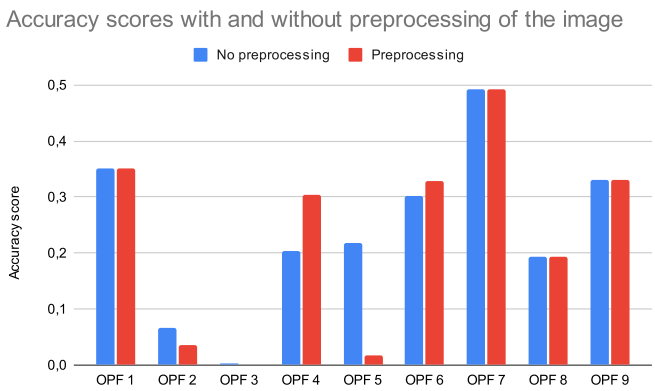


Figure 4.2: OPF reconstructions' accuracy scores using Neutube's preprocessing option

Below, figure 4.3, displays the GS reconstruction to the left and the OPF 1, the first reconstruction of the image stack, to the right. This automated reconstruction of OPF 1 is generated with NCTracer, which obtained the highest score in the accuracy assessment. However, some errors are to be found in the reconstruction, mainly by including extra descendants or branches being longer than the GS. The most notable differences are highlighted using circles.

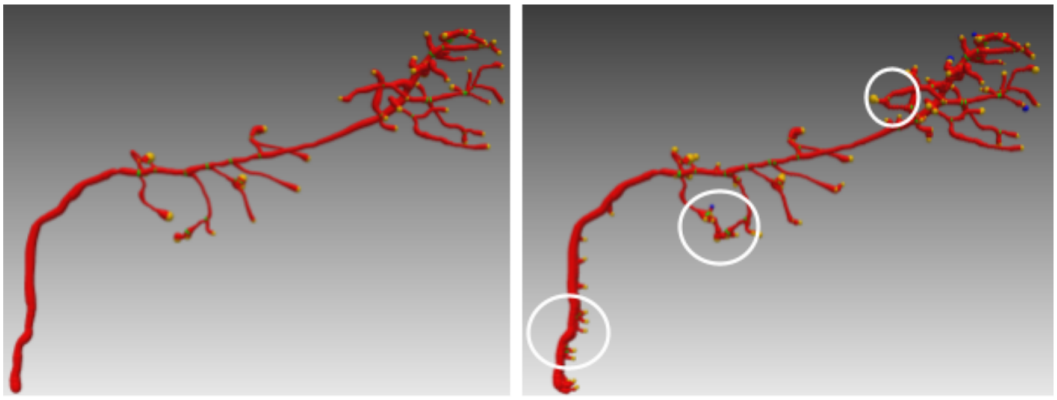
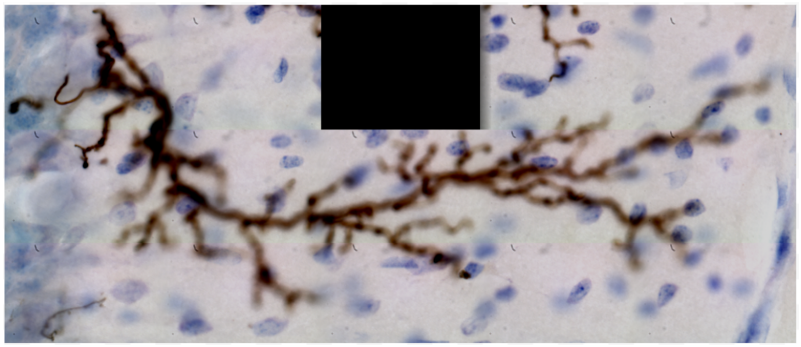
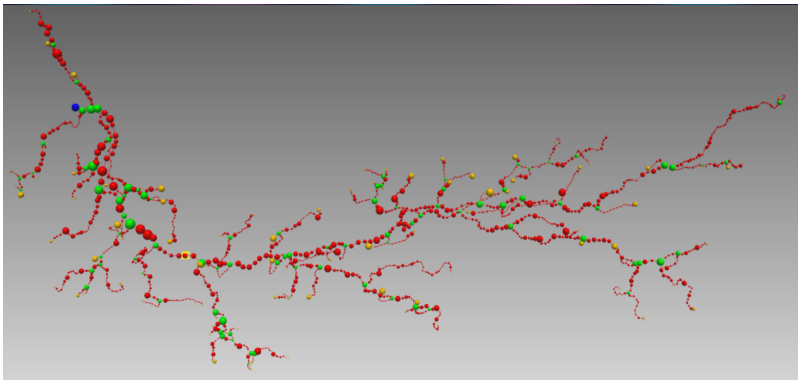


Figure 4.3: OPF 1 reconstruction. GS (left) and automated reconstruction by NCTracer (right).

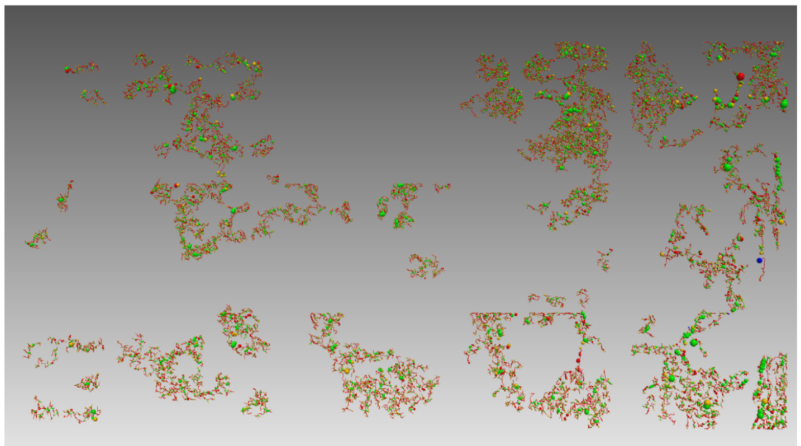
Figure 4.4, displays the reconstruction of CF 3, which obtained an accuracy score of 0 for the three algorithms. The main reasoning behind this is the elevated noise on the original image stack.



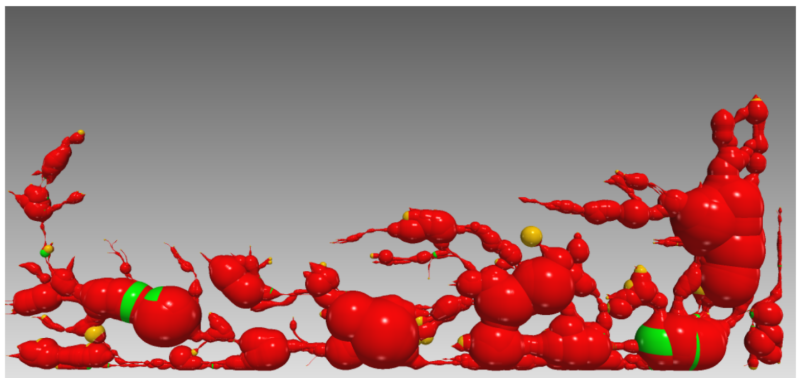
(a)



(b)



(c)



(d)

Figure 4.4: Image Stack of CF 3 (a) and its respective reconstructions: GS (b), NeuTube (c) and Vaa3D (d).

4.2 Efficiency

The evaluation of the efficiency of the software is based on three different assessments: execution time figure 4.5, CPU usage, figure 4.6, and Memory usage, figure 4.7. Table 4.1 presents the image stacks' file sizes in MegaBytes, essential for the assessment of the use of resources.

In figures 4.5 to 4.7, NCTracer fails to have a value for the execution time, CPU or memory usage of the OPF due to the fact that the execution was unsuccessful after a NullPointerException. No values were added for figure 4.5, however, it is worth noting that, for figures 4.6 and 4.7 the absence of a value does not imply a low use of resources but a failure in the execution.

Image Stack	Size (MB)	Image Stack	Size (MB)
CF_2	710	NL1A2 8	4.08
CF_3	611	NL1A2 9	5.16
NL1A1 1	8.77	NL1A2 10	3.81
NL1A1 2	4.75	NL1A1 Fused	109
NL1A1 3	6.22	NL1A2 Fused	262
NL1A1 4	8.2	OPF 1	45
NL1A1 5	8.77	OPF 2	0.488
NL1A1 6	7.45	OPF 3	5.17
NL1A2 1	7.69	OPF 4	16.7
NL1A2 2	5.6	OPF 5	19.00
NL1A2 3	4.76	OPF 6	25.20
NL1A2 4	4.74	OPF 7	17.70
NL1A2 5	4.48	OPF 8	21.20
NL1A2 6	5.08	OPF 9	23.00
NL1A2 7	3.2		

Table 4.1: Image stacks and their respective sizes.

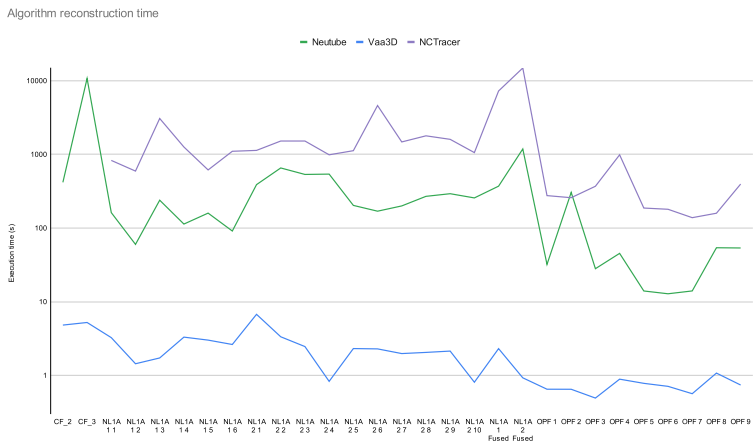


Figure 4.5: Algorithms' execution time on each evaluated dataset. On the X-Axis, the image stack assessed. On the Y-axis, the time in seconds needed to complete the execution of the reconstructions, plotted on a logarithmic scale.

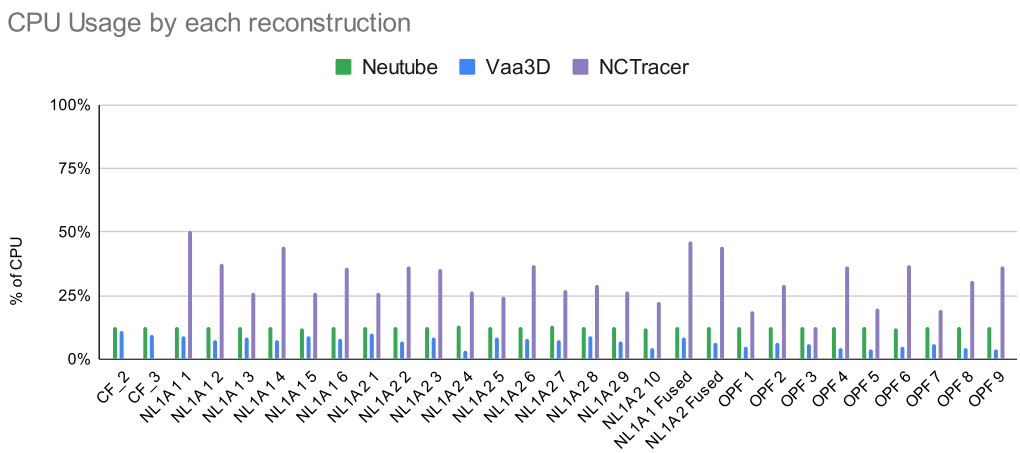


Figure 4.6: Algorithms' CPU usage on each evaluated dataset. On the X-Axis, the image stack assessed. On the Y-axis, the average percentage of CPU utilised for the execution of the reconstructions.

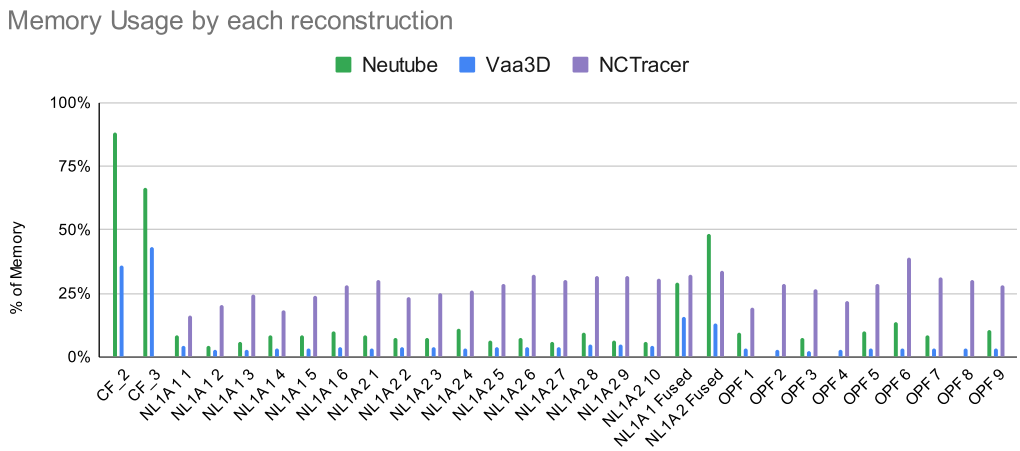


Figure 4.7: Algorithms' memory usage on each evaluated dataset. On the X-Axis, the image stack assessed. On the Y-axis, the average percentage of Physical memory used during the reconstruction process.

4.3 Consistency

To investigate the uniformity and reliability of the results, three image stacks belonging to different datasets were repeatedly reconstructed and studied. The selected neurons were CF 2, NL1A2 7 and OPF 6. Each image stack was reconstructed five times under the same circumstances: before every reconstruction, the image underwent a preprocessing step and, for NCTracer, the input parameters such as the number of steps, remained invariant.

Table 4.2 describes an outline of the consistency experiments: every cell contains the consistency score of the reconstruction upon comparison with the initial reconstruction elaborated during the accuracy analysis. The value can fluctuate between 0 and 1, being the latter the maximum possible grade.

NCTracer failed to execute any reconstructions of the CF 2 stack due to a NullPointerException originated by the elevated size of the file.

Predominantly, the consistency score is 1 for most reconstructions. However, it is worth noting that for those whose value is below the optimal, after an accuracy evaluation using its respective GS, the accuracy score for the reconstruction remained the same as the one obtained with the initial one used for comparison in this section.

Iteration	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
CF 2	1	1	1	1	1	1	1	1	1	1					
NL1A2 7	1	1	1	1	1	1	1	1	1	1	1	0.778	1	1	1
OPF 6	1	1	1	1	1	1	1	1	1	1	1	1	1	0.218	0.218

Table 4.2: Consistency score for the three image stacks reconstructions by each algorithm.

4.4 Ease of use

The evaluation outcomes pertaining to the ease of use of the tools are outlined in table 4.3. These findings contribute to an assessment of the user-friendliness, usability, and intuitiveness of each tool.

					Reconstruction					Image Stack		
Software	**Undo**	**Hot Keys**	**Macros**	**Documentation**	**3D visualisation**	**2D visualisation**	**Editing**	**Translation**	**Colour Coded**	**Processing**	**3D visualisation**	**2D visualisation**
NeuTube	U	Y	N	Y	Y	Y*	Y	Y	Y	Y	Y	Y
Vaa3D	U	Y	N	Y**	Y	N	N	Y	N	Y	Y	Y
NCTracer	N	N	Y	Y	Y	Y	Y	N	N	Y	Y	Y

Undo: U = Unlimited, L = Limited, N = No, Y = Yes.
* 2D Visualisation is only available after the reconstruction, there is no possibility to load from a file a reconstruction on top of an image stack.
** The software's help button does not lead to a manual anymore, but to a 404 error. On the download page for the software [11] there are three documentation options with, as stated, different levels of depth. However, two of them, the User Manual and the GoogleDocs, are unavailable.

Table 4.3: Ease of use analysis for the software

Chapter 5

Discussion

5.1 Results analysis

In this report three different open–source software for neural morphology reconstruction were evaluated. This section will provide a short summary and insight of the most pertinent results in relation to the problem statement and research questions. The results do not indicate a clear superiority of any of the software but rather that each software possesses both positive and negative attributes.

Regarding the focal point of the study, accuracy, NCTracer had the best reconstruction for the majority count of image stacks, with thirteen best reconstructions while the second best, Vaa3D, had seven, not taking into consideration the ties, as seen in figure 4.1. However, if the average score for all the reconstructions was computed, Vaa3D had the highest accuracy. Interestingly, it succeeded in making the most accurate reconstructions of the three, as in score not in amount. The accuracy of the reconstructions could, however, be somewhat polarised as the algorithm occasionally was unable to provide a proper one, where the yielded reconstruction was inaccurate to the point of the totality of the reconstructed neuron being represented by one or multiple massive nodes. NeuTube, on the other hand, was the most stable with the lowest deviation in respect to the average of the three.

Figure 4.3, on the right, displays a reconstruction elaborated by NCTracer, although the other two tools provided a really similar one, where there can be observed erroneous branches and path lengths that deteriorate the accuracy score of the reconstructions. Figure 4.4, on the other hand, presents a neuron that obtained a score of 0 for all the reconstructions as a consequence of the elevated noise the background of the image stack presents (figure 4.4.a.).

On another topic, regarding preprocessing. In figure 4.2, it can be observed how the accuracy score of the reconstructions with and without preprocessing barely

varies, sometimes being superior and sometimes inferior. Given that it only could be tested using Neutube, we cannot confirm that the effect is minimal in accuracy. Since preprocessing was directly included inside the reconstruction's process for the remaining software, what we can confirm is that image stacks that do not undergo preprocessing and enhancing, generate reconstructions with more trees. This surplus is a product of the neuron's reconstruction being more fragmented, but also from the picture's noise not being removed.

It is, however, worthy of note how consistency did not emerge as a decisive factor since all software performed with excellence. Although table 4.2 displayed lower scores for NCTracer on the reconstruction of OPF and NL1A given the fact that the files produced were not completely identical, unlike the rest, the accuracy score produced upon comparison with the respective GS was the same to the one obtained with the initial reconstruction.

NeuTube displayed once again stability in comparison to the other two software in regards to the use of resources, particularly CPU since, independently of the size of the image stack, the average percentage of CPU used was 11.79% with a standard deviation of 0.4687, a low dispersion from the average. Alternatively, the memory usage is affected by the size of the image stack, not only for NeuTube, but also for NCTracer and Vaa3D. On the other hand, Vaa3D had a significantly lower execution time than the other two algorithms; that much lower that it became necessary to use a logarithmic scale on the Y-Axis in figure 4.5 so as to be able to appreciate the magnitude in comparison to NeuTube and NCTracer.

As beginners of this field of science and expertise, we were able to properly judge the set up difficulty, ease of use and steepness of the learning curve for each of the software. All three software counted with easy and aided installation executable for both Windows and iOS, except NCTracer which was only available for Windows operating system. However, NCTracer used Java(TM) and the presence of any other JAVA versions installed on the device caused the program to not allow the loading of the image stacks. Regarding ease of use, NCTracer was the most intuitive to use, followed by NeuTube. Vaa3D failed to provide active feedback to the user through the interface, but displayed it through the terminal. Although it was through understandable messages, it made the application less intuitive and interactive. Furthermore, Vaa3D's automatic tracing algorithm was unavailable for iOS, which limits its accessibility. NCTracer counted with well-designed and simple menus with the features located where they were expected to be. Additionally, it provided an elevated sense of control to the user due to the fact that all parameters from the algorithm were configurable and can be set by the user. NCTracer was the most robust application of the three, provided that NeuTube failed to do more than two reconstructions one after another without reloading the whole program, and, particularly for the image stacks

with bigger sizes, would close unexpectedly amid program execution. Similarly, and although Vaa3D was more stable, it had a tendency to freeze quite often.

On the contrary, the program that allowed for the easier manipulation of reconstructions, both in 3D view and by command line was NeuTube. In view of NL1A forming a mosaic, the GS reconstructions were translated with respect to the coordinate at the origin, which is where these are placed by default, being able to translate them was a must in order to obtain a proper accuracy score. NeuTube fulfilled all the expectations regarding these and, although Vaa3D also counted with the feature of translating the coordinates of the nodes in the SWC file, it was less intuitive and therefore harder to use than NeuTube.

Moreover, NeuTube provided the smoothest process of reconstruction out of the three. Vaa3D generated all the reconstructions flipped along the Y-Axis and therefore required a previous preprocessing of the image stack so as to obtain a loyal reconstruction in regard to the standard. On their part, NCTracer exported the reconstructions into the SWC file using commas to denote the decimal point of the numbers. These made, not only the DIADEM Metric crash, but also caused an error message in its own software when trying to import one of its own reconstructions to visualise it, as the format the SWC file was imported in was not considered valid.

These results are, to some extent, aligned with the study conducted by Feng, Zhao and Kim [16]. Their conclusions showcase the outperformance of neuTube in front of the other two software evaluated. Although semi-manual reconstructions were the feature assessed, our results from automated reconstructions are not distant from their conclusions. While it did not outperform the other tools, it did behave and execute remarkable reconstructions, being the most stable of the three.

NCTracer provided two possible algorithms for the reconstructions: Voxel coding and Fast Marching. Initially, the intent was to employ for reconstructions was Voxel Coding after having read the documentation offered on the method, however, it failed to finish the reconstructions for most of the image stacks of the datasets after taking over 10 hours to do so.

5.2 Limitations

The selection of software to evaluate was a crucial determinant. As was briefly mentioned in chapter 1.2, the selected software had a common denominator being that they were all open-source software. This selection was mainly based on budgetary limitations as most tools exist under a commercial licence and due to their excessive pricing, restricting this study to a cost-free software. Furthermore, most open-source software were over a decade old and no longer under development by its developers, as was evident by faulty or complicated installation procedures, or outdated and

inactive web pages.

Moreover, compiling a proper dataset was challenging, Although over the last decade, the amount of reconstructions available on online databases has increased significantly, the obtention of the original neuron's image stack from the database was mostly impossible. This led to a rather small and undiverse dataset. Such was the restriction, that we were unable to collect neurons other than axons, therefore this study might be affected by the incapability of finding a full neuron, with all parts, to reconstruct.

However, the biggest limitation encountered during the conduction of this research was our lack of expertise in this field. We are nothing more than beginners regarding neuromorphology, our eye is not critical enough to determine whether a reconstruction is fitting or not. Additionally, although we tried to find the most suitable values to use for the input parameters for the algorithms, we are far from being able to discern if they are the most appropriate for the characteristics of every image stack.

5.3 Future research

The study conducted for this research paper and the resulting outcomes are rather restricted, mainly by the dataset compiled and the specifications of the computer executing the experiments. In future studies, a larger dataset should be compiled, both in size and, particularly, diversity of neuron types, given that, overall, the accuracy score of the reconstructions remains moderately similar for all image stacks inside a dataset, but can greatly differentiate between datasets. Additionally, on the same line of thought, the reconstruction of neurons containing other structures than axons would be valuable for this field.

Another recommendation concerns the substitution or modification of the DIADEM Metric by removing the weighted evaluation of the nodes. The DIADEM Metric assigns a higher weight to nodes closer to the roots of the reconstructed trees since the evaluation is done from the root downwards. In practice, a missing or wrongly located node near the root or the branch end are the same type of error with equal impact when it comes to manually addressing the issue. This amendment of the metric would aid in providing a more precise accuracy score for reconstructions.

In order to more accurately measure each algorithm's execution time, CPU usage and memory usage, future studies should be conducted with the source code in lieu of the software. Doing so will enable the conductors of the study to modify the source code to better fit their needs in benchmarking rather than inventing their own scripts and using system monitoring tools provided by the OS to accomplish this.

Finally, as mentioned in section 5.2, a limitation has been that no commercial

software could be included in the comparison of the study. It would be particularly interesting to conduct a similar study regarding the evaluation of open-source software for neural morphology reconstruction including a commercially distributed tool.

Chapter 6

Conclusions

The aim of this study was to discover the field of neuron morphology reconstruction's current state of development by comparing the numerous advantages and disadvantages the software offers. The results of this study unveil that all software produced somewhat similar outcomes, varying depending on the dataset sample being reconstructed. It also highlighted that each software had their own areas they excel in and shortcomings to be improved upon.

These results can aid in generating a more standardised, consistent and cohesive community in the field or more collaboration and exchange of ideas. By analysing and comparing the advantages and disadvantages of these software, it can assist in the development of more efficient and reliable software or improve upon already existing software.

The focus on solely analysing open-source software instead of those requiring a commercial licence was a result of budgetary restrictions, which limited the field's accessibility. It is, however, worth noting that upon further research and the opinion of experts, that other commercial software such as Neurolucida provide equal or better accuracy and efficiency as well as documentation and formation for the researchers. Most software under consideration was either discontinued or inaccessible anymore.

This study might aid researchers or universities that study neuromorphology, work with these software and/or suffer from budgetary restrictions. This can make projects more economically viable and sustainable. As a result, this project can help facilitate better understanding for the field and enable those interested to partake and boost the acceleration in technological advancement for neuromorphology and neuroscience as a whole. In doing so, the gained knowledge might one day be applied in other scientific fields such as generating better models for artificial intelligence or in medical diagnostics for the identification and treatment of neurodegenerative

diseases, psychological conditions and physical trauma; injuries just like the ones described in the Edwin Smith surgical papyrus, but were incapable of treating.

Bibliography

[1] Stanley Finger. *The Dawn of Neurology*. Oxford University Press, USA, 2000, pp. 7–19. URL: https://books.google.se/books?id=3OWU1wnOy84C (visited on 02/22/2023).

[2] *Neuroscience*. Oxford Advanced Learner's Dictionary. URL: https://www.oxfordlearnersdictionaries.com/definition/english/neuroscience (visited on 02/22/2023).

[3] Larry R. Squire et al. *Fundamental Neuroscience*. 4th. Academic Press, USA, 2013, p. 3. URL: https://books.google.se/books?id=AEmEn-_hD9IC (visited on 02/23/2023).

[4] *NCI Dictionary of Cancer terms*. National Cancer Institute. URL: https://www.cancer.gov/publications/dictionaries/cancer-terms/def/neuron (visited on 05/15/2023).

[5] Aaron Woodruff. *What is a neuron? - Queensland Brain Institute - University of Queensland*. Queensland Brain Institute. URL: https://qbi.uq.edu.au/brain/brain-anatomy/what-neuron (visited on 02/26/2023).

[6] Duncan E. Donohue and Giorgio A. Ascoli. "Automated reconstruction of neuronal morphology: An overview". In: *Brain Research Reviews* 67.1-2 (Nov. 2010), pp. 94–102. URL: https://www.sciencedirect.com/science/article/pii/S0165017310001293 (visited on 03/29/2023).

[7] *DIADEM Challenge*. URL: https://diadem.janelia.org/index.html (visited on 02/23/2023).

[8] *Neuromorpho.Org*. URL: https://neuromorpho.org (visited on 02/23/2023).

[9] Hanchuan Peng et al. "BigNeuron: Large-Scale 3D Neuron Reconstruction from Optical Microscopy Images". In: *Neuron* (2015). URL: https://doi.org/10.1016/j.neuron.2015.06.036 (visited on 03/26/2023).

[10] *NeuTube*. URL: https://neutracing.com/ (visited on 03/29/2023).

[11] Hanchuan Peng. *Vaa3D* -. URL: http://home.penglab.com/proj/vaa3d/home/index.html (visited on 03/22/2023).

[12] *Neural Circuit Tracer*. Neurogeometry Lab. URL: https://neurogeometry.sites.northeastern.edu/neural-circuit-tracer/ (visited on 04/06/2023).

[13] Marcel Oberlaender. "Neuronal morphology and its significance". In: *The Neocortex*. Ed. by Wolf Singer. 2019, pp. 125–139. URL: https://pure.mpg.de/pubman/faces/ViewItemOverviewPage.jsp?itemId=item_3186507_3 (visited on 02/13/2023).

[14] Xiaobai Li and Hualou Liang. "Digital Reconstructions of Neuronal Morphology: Three Decades of Research Trends". In: *Frontiers* 23 (Mar. 2012). URL: https://www.frontiersin.org/articles/10.3389/fnins.2012.00049/full (visited on 02/26/2023).

[15] Damien M. O'Halloran. "Module for SWC neuron morphology file validation and correction enabled for high throughput batch processing". In: *PloS one* (Jan. 2020). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6977721/ (visited on 05/23/2023).

[16] Lei Feng, Tingwei Zhao, and John Kim. "NeuTube 1.0: A new design for efficient neuron reconstruction software based on the SWC format". In: *eNeuro* 2.1 (Jan. 2015). URL: https://www.eneuro.org/content/2/1/ENEURO.0049-14.2014 (visited on 04/04/2023).

[17] Hanchuan Peng et al. "Automatic reconstruction of 3D neuron structures using a graph-augmented deformable model". In: *Academic.oup.com* (June 2010). URL: https://academic.oup.com/bioinformatics/article/26/12/i38/285842 (visited on 04/04/2023).

[18] Edsger W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numerische Mathematik* 1 (July 2022), pp. 269–271. URL: https://doi.org/10.1145/3544585.3544600 (visited on 04/04/2023).

[19] J. Kim and et al. "MGRASP enables mapping mammalian synaptic connectivity with light microscopy". In: *Nature News* (Dec. 2011). URL: https://www.nature.com/articles/nmeth.1784 (visited on 05/16/2023).

[20] Tingwei Zhao and et al. "Automated reconstruction of neuronal morphology based on local geometrical and global structural models - neuroinformatics". In: *SpringerLink* (Nov. 2011). URL: https://link.springer.com/article/10.1007/s12021-011-9120-3 (visited on 04/25/2023).

[21] R. Unnikrishnan, C. Pantofaru, and M. Hebert. "Toward objective evaluation of image segmentation algorithms". In: *IEEE Xplore* (Apr. 2007). URL: https://ieeexplore.ieee.org/abstract/document/4160946/ (visited on 04/07/2023).

[22] P. Chothani, V. Mehta, and A. Stepanyants. "Automated tracing of neurites from light microscopy stacks of images - neuroinformatics". In: *SpringerLink* (May 2011). URL: https://link.springer.com/article/10.1007/s12021-011-9121-2 (visited on 04/07/2023).

[23] James A. Sethian. *Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials sciences*. Cambridge University Press, 2010. URL: https://doi.org/10.1145/3544585.3544600 (visited on 04/07/2023).

[24] Ronak Gala et al. "Active learning of neuron morphology for accurate automated tracing of neurites". In: *Frontiers in neuroanatomy* (May 2014). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4032887/ (visited on 04/27/2023).

[25] Gleb T.K. GA. "The diadem metric: Comparing multiple reconstructions of the same neuron". In: *Neuroinformatics* (Sept. 2011). URL: https://pubmed.ncbi.nlm.nih.gov/21519813/ (visited on 05/15/2023).

[26] Kris M. Brown and et al. "The diadem data sets: Representative light microscopy images of neuronal morphology to advance automation of digital reconstructions". In: *Neuroinformatics* (Sept. 2011). URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4342109/ (visited on 05/05/2023).

[27] Izumi Sugihara, Hsiu-Sheng Wu, and Yoshihisa Shinoda. "Morphology of single olivocerebellar axons labeled with biotinylated dextran amine in the rat". In: *Wiley Online Library* (Oct. 1999). URL: https://onlinelibrary.wiley.com/doi/10.1002/(SICI)1096-9861(19991115)414:2%5C%3C131::AID-CNE1%5C%3E3.0.CO;2-F (visited on 05/05/2023).

[28] Angelo J. Canty and Valentina Di Paola. "Axonal reconstructions going live - neuroinformatics". In: *SpringerLink* (Apr. 2011). URL: https://link.springer.com/article/10.1007/s12021-011-9112-3 (visited on 05/06/2023).

[29] Gregory S.X.E. Jefferis and et al. "Comprehensive Maps of Drosophila Higher Olfactory Centers: Spatially Segregated Fruit and Pheromone Representation". In: *Cell* (Mar. 2007). URL: https://pubmed.ncbi.nlm.nih.gov/17382886/ (visited on 05/23/2023).

[30] Ammar A. Kabbani. *Projection fibers of the Brain: Radiology Reference Article*. Radiopaedia Blog RSS. Dec. 2019. URL: https://radiopaedia.org/articles/projection-fibres-of-the-brain (visited on 05/05/2023).

[31] Andre Fiala. "Olfaction and olfactory learning in Drosophila: Recent progress". In: *Current Opinion in Neurobiology* (2007). URL: https://www.sciencedirect.com/science/article/pii/S0959438807001274?via%5C%3Dihub (visited on 05/05/2023).

[32] E.C. Marin et al. "Representation of the Glomerular Olfactory Map in the Drosophila Brain". In: *Cell* (Apr. 2022). URL: https://pubmed.ncbi.nlm.nih.gov/12007410/ (visited on 05/23/2023).

[33] Tom Lee and Liqun Luo. "Mosaic analysis with a repressible cell marker for studies of gene function in neuronal morphogenesis". In: *Neuron* (). URL: https://pubmed.ncbi.nlm.nih.gov/10197526/ (visited on 05/06/2023).

[34] *High-performance embedded computing (second edition)*. Chapter 4 - Processes and Operating Systems. Morgan Kaufmann, 2014. (Visited on 05/17/2023).

[35] Mark Russinovich. *Process explorer - sysinternals*. Apr. 2023. URL: https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer (visited on 05/16/2023).

[36] Abraham Silberschatz, Peter B. Galvin, and Greg Gagne. *Operating system concepts*. 9.2.2 Memory Allocation, 9.3 Paging, 9.5 Swapping. Wiley, 2018. (Visited on 05/03/2023).

Appendix A

Script for measuring accuracy

Script_neocortical_1.ps

```
Param(
    [switch]$n,
    [switch]$m,
    [switch]$l,
    [switch]$f
)

function has_content(){
    Param(
        [String]$fileName
    )
    $file = "C:PATH\Diadem Metric\" + $fileName
    $content = Get-Content $file
    if ([string]::IsNullOrWhiteSpace($content))
        {
        return $false
    } else {
        return $true
    }
}

function TreeClean(){
    Param(
        [String]$file
    )
```

```
    $path = "C:PATH\Diadem Metric\" + $file

    $inputfile = [System.IO.File]::OpenText(
       $path)
    $outputFile = [System.IO.File]::CreateText
       ("C:PATH\Diadem Metric\testtmp.swc")

    $counter = 0

    while (($line = $inputfile.ReadLine()) -ne
       $null) {
        $lineArray = $line.Split(' ')
        if ($counter -ge 2){
            $outputFile.WriteLine($line)
        }
        else {
            if ($lineArray[-1] -eq "-1") {
                $counter += 1
                if ($counter -ge 2){
                    $outputFile.WriteLine($line
                       )
                }
            }
        }
    }

    $inputFile.Close()
    $outputFile.Close()

    $rmpath = "C:PATH\Diadem Metric\" + $file

    Remove-Item -Path $rmpath
    Rename-Item -Path "C:PATH\Diadem Metric\
       testtmp.swc" -NewName $file
}
```

```
$count = 1
$count2 = 1

$outputFile = "output.txt"
New-Item -ItemType File -Path $outputFile -
   Force

if ($n){
    Write-Output "Flag N: Neutube" | Out-File -
       FilePath $outputFile -Append
    if ($f){
        while ($count -lt 35){
            if ($count -lt 10) {
                $golden = "NC_0" + $count + ".
                   swc"
            }
            else {
                $golden = "NC_" + $count + ".
                   swc"
            }
            Write-Output $golden | Out-File -
               FilePath $outputFile -Append
            java -jar DiademMetric.jar -G
               $golden -T "Neutube_Fused_1.swc"
                -D 3 | Out-File -FilePath
               $outputFile -Append
            $count += 1
        }
    }
    else {
        while ($count2 -lt 7){
            $recons = "Neutube_Stack_0" +
               $count2 + ".swc"
            Write-Output $recons | Out-File -
               FilePath $outputFile -Append
            while ((Test-Path $recons) -and (
               has_content($recons))){
                $count = 1
                while ($count -lt 35) {
```

```
                        if ($count -lt 10) {
                            $golden = "NC_0" +
                               $count + ".swc"
                        }
                        else {
                            $golden = "NC_" +
                               $count + ".swc"
                        }

                        if ((Test-Path $golden) -
                           and (Test-Path $recons))
                            {
                            Write-Output $golden |
                               Out-File -FilePath
                               $outputFile -Append
                            java -jar DiademMetric.
                               jar -G $golden -T
                               $recons -D 3 | Out-
                               File -FilePath
                               $outputFile -Append
                        }
                        $count += 1
                    }
                    TreeClean($recons)
                }
                $count2 += 1
            }
        }
    }
    elseif ($m) {
        Write-Output "Flag M: NCTracer" | Out-File
           -FilePath $outputFile -Append
        if ($f){
            while (has_content("NCTracer_Fused_1.
               swc")){
                $count = 1
                while ($count -lt 35){
                    if ($count -lt 10) {
```

```
                $golden = "NC_0" + $count +
                    ".swc"
            }
            else {
                $golden = "NC_" + $count +
                   ".swc"
            }
            Write-Output $golden | Out-File
                -FilePath $outputFile -
               Append
            java -jar DiademMetric.jar -G
               $golden -T "NCTracer_Fused_1
               .swc" -D 3 | Out-File -
               FilePath $outputFile -Append
            $count += 1
        }
        TreeClean("NCTracer_Fused_1.swc")
    }
}
else {
    while ($count2 -lt 7){
        $recons = "NCTracer_Stack_0" +
           $count2 + ".swc"
        Write-Output $recons | Out-File -
           FilePath $outputFile -Append
        while ((Test-Path $recons) -and (
           has_content($recons))){
            $count = 1
            while ($count -lt 35) {
                if ($count -lt 10) {
                    $golden = "NC_0" +
                       $count + ".swc"
                }
                else {
                    $golden = "NC_" +
                       $count + ".swc"
                }
```

```
                    if ((Test-Path $golden) -
                       and (Test-Path $recons))
                        {
                        Write-Output $golden |
                           Out-File -FilePath
                           $outputFile -Append
                        java -jar DiademMetric.
                           jar -G $golden -T
                           $recons -D 3 | Out-
                           File -FilePath
                           $outputFile -Append
                    }
                    $count += 1
                }
                TreeClean($recons)
            }
            $count2 += 1
        }
    }
}
else {
    Write-Output "Vaa3D" | Out-File -FilePath
       $outputFile -Append
    if ($f){
        while (has_content("Vaa3D_Fused_1.swc")
           ){
            $count = 1
            while ($count -lt 35){
                if ($count -lt 10) {
                    $golden = "NC_0" + $count +
                        ".swc"
                }
                else {
                    $golden = "NC_" + $count +
                       ".swc"
                }
                Write-Output $golden | Out-File
                    -FilePath $outputFile -
                   Append
```

```
                java -jar DiademMetric.jar -G
                   $golden -T "Vaa3D_Fused_1.
                   swc" -D 3 | Out-File -
                   FilePath $outputFile -Append
                $count += 1
            }
            TreeClean("Vaa3D_Fused_1.swc")
        }
    }
    else {
        while ($count2 -lt 7){
            $recons = "Vaa3D_Stack_0" + $count2
                + ".swc"
            Write-Output $recons | Out-File -
               FilePath $outputFile -Append
            while ((Test-Path $recons) -and (
               has_content($recons))){
                $count = 1
                while ($count -lt 35) {
                    if ($count -lt 10) {
                        $golden = "NC_0" +
                           $count + ".swc"
                    }
                    else {
                        $golden = "NC_" +
                           $count + ".swc"
                    }

                    if ((Test-Path $golden) -
                       and (Test-Path $recons))
                        {
                        Write-Output $golden |
                           Out-File -FilePath
                           $outputFile -Append
                        java -jar DiademMetric.
                           jar -G $golden -T
                           $recons -D 3 | Out-
                           File -FilePath
                           $outputFile -Append
```

```
                        }
                        $count += 1
                    }
                    TreeClean($recons)
                }
                $count2 += 1
            }
        }
    }
```

Script_neocortical_2.ps

```
Param(
    [switch]$n,
    [switch]$m,
    [switch]$l,
    [switch]$f
)

function has_content(){
    Param(
        [String]$fileName
    )
    $file = "C:PATH\Diadem Metric\" + $fileName
    $content = Get-Content $file
    if ([string]::IsNullOrWhiteSpace($content))
        {
        return $false
    } else {
        return $true
    }
}

function TreeClean(){
    Param(
        [String]$file
    )

    $path = "C:PATH\Diadem Metric\" + $file

    $inputfile = [System.IO.File]::OpenText(
       $path)
    $outputFile = [System.IO.File]::CreateText
       ("C:PATH\Diadem Metric\testtmp.swc")

    $counter = 0

    while (($line = $inputfile.ReadLine()) -ne
       $null) {
```

```
            $lineArray = $line.Split(' ')
            if ($counter -ge 2){
                $outputFile.WriteLine($line)
            }
            else {
                if ($lineArray[-1] -eq "-1") {
                    $counter += 1
                    if ($counter -ge 2){
                        $outputFile.WriteLine($line
                            )
                    }
                }
            }
        }

        $inputFile.Close()
        $outputFile.Close()

        $rmpath = "C:PATH\Diadem Metric\" + $file

        Remove-Item -Path $rmpath
        Rename-Item -Path "C:PATH\Diadem Metric\
            testtmp.swc" -NewName $file
    }

    $letters = "A","B","C","D","E","F","G","H","I
        ","J","K","L","M","N","O","P","Q","R","S","T
        ","U"

    $count = 1
    $count2 = 1

    $outputFile = "output.txt"
    New-Item -ItemType File -Path $outputFile -
        Force

    if ($n){
        Write-Output "Flag N: Neutube" | Out-File -
            FilePath $outputFile -Append
```

```
if ($f){
    while (has_content("Neutube_Fused_2.swc
       ")){
        foreach ($letter in $letters) {
            $golden = "NC_" + $letter + ".
               swc"
            Write-Output $golden | Out-File
                -FilePath $outputFile -
               Append
            java -jar DiademMetric.jar -G
               $golden -T "Neutube_Fused_2.
               swc" -D 3 | Out-File -
               FilePath $outputFile -Append
        }
        TreeClean("Neutube_Fused_2.swc")
    }
}
else {
    while ($count2 -lt 11){
        $recons = "Neutube_Stack_0" +
           $count2 + ".swc"
        if ($count2 -eq 10) {
            $recons = "Neutube_Stack_" +
               $count2 + ".swc"
        }
        Write-Output $recons | Out-File -
           FilePath $outputFile -Append
        while ((Test-Path $recons) -and (
           has_content($recons))){
            foreach ($letter in $letters) {
                $golden = "NC_" + $letter +
                    ".swc"
                if ((Test-Path $golden) -
                   and (Test-Path $recons))
                    {
                    Write-Output $golden |
                       Out-File -FilePath
                       $outputFile -Append
                    java -jar DiademMetric.
```

```
                                jar -G $golden -T
                                $recons -D 3 | Out-
                                File -FilePath
                                $outputFile -Append
                            }
                        }
                        TreeClean($recons)
                    }
                    $count2 += 1
                }
            }
        }
        elseif ($m) {
            Write-Output "Flag M: NCTracer" | Out-File
               -FilePath $outputFile -Append
            if ($f){
                while (has_content("NCTracer_Fused_2.
                   swc")){
                    foreach ($letter in $letters) {
                        $golden = "NC_" + $letter + ".
                           swc"
                        Write-Output $golden | Out-File
                            -FilePath $outputFile -
                           Append
                        java -jar DiademMetric.jar -G
                           $golden -T "NCTracer_Fused_2
                           .swc" -D 3 | Out-File -
                           FilePath $outputFile -Append
                    }
                    TreeClean("NCTracer_Fused_2.swc")
                }
            }
            else {
                while ($count2 -lt 11){
                    $recons = "NCTracer_Stack_0" +
                       $count2 + ".swc"
                    if ($count2 -eq 10) {
                        $recons = "NCTracer_Stack_" +
                           $count2 + ".swc"
```

```
                }
                Write-Output $recons | Out-File -
                   FilePath $outputFile -Append
                while ((Test-Path $recons) -and (
                   has_content($recons))){
                    foreach ($letter in $letters) {
                        $golden = "NC_" + $letter +
                            ".swc"
                        if ((Test-Path $golden) -
                           and (Test-Path $recons))
                            {
                            Write-Output $golden |
                               Out-File -FilePath
                               $outputFile -Append
                            java -jar DiademMetric.
                               jar -G $golden -T
                               $recons -D 3 | Out-
                               File -FilePath
                               $outputFile -Append
                        }
                    }
                    TreeClean($recons)
                }
                $count2 += 1
            }
        }
    }
    else {
        Write-Output "Vaa3D" | Out-File -FilePath
           $outputFile -Append
        if ($f){
            while (has_content("Vaa3D_Fused_2.swc")
               ){
                foreach ($letter in $letters) {
                    $golden = "NC_" + $letter + ".
                       swc"
                    Write-Output $golden | Out-File
                        -FilePath $outputFile -
                       Append
```

```
                java -jar DiademMetric.jar -G
                    $golden -T "Vaa3D_Fused_2.
                    swc" -D 3 | Out-File -
                    FilePath $outputFile -Append
            }
            TreeClean("Vaa3D_Fused_2.swc")
        }
    }
    else {
        while ($count2 -lt 11){
            $recons = "Vaa3D_Stack_0" + $count2
                + ".swc"
            if ($count2 -eq 10) {
                $recons = "Vaa3D_Stack_" +
                    $count2 + ".swc"
            }
            Write-Output $recons | Out-File -
                FilePath $outputFile -Append
            while ((Test-Path $recons) -and (
                has_content($recons))){
                foreach ($letter in $letters) {
                    $golden = "NC_" + $letter +
                        ".swc"
                    if ((Test-Path $golden) -
                        and (Test-Path $recons))
                        {
                        Write-Output $golden |
                            Out-File -FilePath
                            $outputFile -Append
                        java -jar DiademMetric.
                            jar -G $golden -T
                            $recons -D 3 | Out-
                            File -FilePath
                            $outputFile -Append
                    }
                }
                TreeClean($recons)
            }
            $count2 += 1
```

```
            }
        }
    }
```

Appendix B

Script for measuring CPU and memory usage

```
$param1 = $args[0]
$Process = Get-Process -Name $param1
$TotalMemory = (Get-WmiObject -Class
   Win32_ComputerSystem).TotalPhysicalMemory
$numberProcessors = (Get-WmiObject -Class
   Win32_ComputerSystem).
   NumberOfLogicalProcessors
$path = "\Proceso($($Process.Name))\% de tiempo
    de procesador"

while ($true) {
    $Data = [PSCustomObject]@{
        Timestamp = Get-Date -Format "yyyy-MM-
           dd HH:mm:ss.fff"
        CPU = (Get-Counter -Counter $path).
           CounterSamples.CookedValue /
           $numberProcessors
        Memory = ($Process.WorkingSet /
           $TotalMemory) * 100
    }
    $Data | Export-Csv -Path "C:\Users\alici\
       OneDrive\Desktop\cpu_usage.csv" -Append
       -NoTypeInformation
    Start-Sleep -Milliseconds 1000
}
```

Appendix C

Software parameter measurements

		NeuTube			
Name	**Size (MBytes)**	**Time (s)**	**CPU-usage (%)**	**Memory (%)**	**Accuracy (%)**
CF_2	710	418,343	12,5	88,325	0
CF_3	611	10743,609	12,53	66,57	0
NL1A1 1	8,77	162,256	12,45	8,72	0,0173
NL1A1 2	4,75	60,156	12,8	4,35	0,01525
NL1A1 3	6,22	239,289	12,7	6,13	0,011167
NL1A1 4	8,2	113,5	12,65	8,239	0,00025
NL1A1 5	8,77	160	12,04	8,236	0,000706
NL1A1 6	7,45	91,312	12,45	10,068	0,00767
NL1A2 1	7,69	389,328	12,47	8,498	0
NL1A2 2	5,6	655,593	12,78	7,412	0,00233
NL1A2 3	4,76	535,343	12,41	7,444	0,001125
NL1A2 4	4,74	540,953	13,2	11,159	0,000714
NL1A2 5	4,48	203,406	12,56	6,5367	0,010125
NL1A2 6	5,08	169,75	12,49	7,589	0,0074
NL1A2 7	3,2	200,125	12,89	5,6567	0,002
NL1A2 8	4,08	269,986	12,61	9,3967	0,00056
NL1A2 9	5,16	293,156	12,53	6,41	0
NL1A2 10	3,81	256,984	12,3	5,933	0,001
NL1A1 Fused	109	371,093	12,5	29,08	0
NL1A2 Fused	262	1.187,188	12,48	48,109	0
OPF 1	45	32,171	12,37	9,41	0,351
OPF 2	0,488	305,359	12,5	10,32	0,036
OPF 3	5,17	28,125	12,5	7,65	0
OPF 4	16,7	45,420	12,5	8,848	0,305
OPF 5	19,00	14,015	12,48	9,829	0,017
OPF 6	25,20	12,843	12,34	13,462	0,328
OPF 7	17,70	14,036	12,38	8,7	0,492
OPF 8	21,20	54,140	12,45	8,561	0,193
OPF 9	23,00	53,750	12,48	10,29	0,331

Vaa3D				NCTracer			
Time (s)	CPU-Usage (%)	Memory (%)	Accuracy (%)	Time (s)	CPU-Usage (%)	Memory (%)	Accuracy (%)
4,827	10,85	35,891	0	0	0	0	0
5,234	9,62	43,199	0	0	0	0	0
3,259	8,79	4,15	0,01313	829,859	50,2	16,45	0,0673
1,441	7,46	2,53	0,010125	594,53	37,67	20,52	0,00325
1,727	8,67	2,56	0,00167	3078,656	26,18	24,568	0,021167
3,312	7,224	3,363	0	1262,282	44,3	18,306	0,000875
3,017	9,205	3,425	0,01082	616,313	26,01	23,856	0,0007058
2,631	7,883	3,687	0,0053	1104,233	35,86	28,357	0,0121
6,754	9,804	3,34	0	1135,109	25,93	30,186	0
3,348	6,771	3,625	0,001	1521,531	36,56	23,541	0,007
2,466	8,578	3,65	0,00125	1521,531	35,6	24,897	0,021625
0,831	3,328	3,2	0	989,219	26,46	25,96	0,025
2,316	8,254	3,9	0,001625	1124,938	24,62	28,775	0,00325
2,293	8,005	4,033	0,004	4614,953	36,91	32,45	0,001
1,981	7,41	3,9	0	1478,672	27,18	30,409	0,0028
2,054	8,747	4,617	0,00212	1790,047	29,4	31,761	0,0064
2,141	6,76	4,7	0	1603,859	26,7	31,709	0,00225
0,809	4,522	4,3	0	1059,875	22,56	30,953	0,003
2,312	8,468	15,9	0	7272,719	46,2	32,529	0
0,928	6,352	13,25	0	14917,375	44,3	33,678	0
0,65	4,625	3,23	0,298	276,188	18,98	19,427	0,383
0,649	6,31	2,65	0	258,734	29,25	28,45	0,015
0,493	5,86	2,33	0	369,94	12,4	26,692	0,215
0,89	4,07	2,7	0,729	988,313	36,45	21,741	0,201
0,781	3,85	3,13	0,453	187,281	19,95	28,56	0,122
0,711	4,94	3,4	0,91	180,172	36,96	39,0168	0,7
0,565	5,95	3,4	0,839	138,583	19,34	31,316	0,374
1,076	4,3	3,3	0,414	159,235	30,73	30,047	0,14
0,741	3,955	3,5	0,779	396,25	36,3	27,995	0,32

No preprocessing	Preprocessing
0.351	0.351
0.067	0.036
0.003	0
0.204	0.305
0.217	0.017
0.302	0.328
0.492	0.492
0.193	0.193
0.331	0.331