TU WIEN Informatics

Visualisierung und Analyse der 3D-Konnektivität auf zellulärer Ebene

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medizinische Informatik

eingereicht von

Benjamin Beinder, BSc.
Matrikelnummer 11808602

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Assistant Prof. Dr. Renata Raidou
Mitwirkung: Eric Mörth, PhD
Associate Prof. Dr. Nils Gehlenborg

Wien, 23. Jänner 2025

Benjamin Beinder | Renata Raidou

TU WIEN Informatics

Visualization and Analysis of 3D-Connectivity at Cell Resolution

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Medical Informatics

by

Benjamin Beinder, BSc.
Registration Number 11808602

to the Faculty of Informatics

at the TU Wien

Advisor: Assistant Prof. Dr. Renata Raidou
Assistance: Eric Mörth, PhD
Associate Prof. Dr. Nils Gehlenborg

Vienna, January 23, 2025

Benjamin Beinder | Renata Raidou

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Benjamin Beinder, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 23. Jänner 2025

Benjamin Beinder

Danksagung

Zuallererst möchte ich mich bei meiner Familie bedanken, denn sie waren immer für mich da und all dies wäre ohne sie gar nicht erst möglich gewesen. Mein Dank geht auch an meine Mitbewohner, welche die Zeit in Wien eine Unvergessliche gemacht haben, wodurch ich unbeschwert studieren konnte. Und auch wenn ich mir oft schwertue sie anzunehmen, will ich mich besonders bei meiner Freundin für ihre stetige Unterstützung bedanken.

Mein besonderer Dank geht an meine Betreuer Eric Mörth und Renata Raidou, die mich in dieser Arbeit mit konstruktivem Feedback, der Organisation der Datensätze und wertvollem Input begleitet haben.

Abschließend möchte ich mich bei Clarence Yapp, Peter Sorger, Liam McLaughlin, Sanjay Jain und Ashley Kiemen für die Bereitstellung der in dieser Arbeit verwendeten Datensätze bedanken.

Acknowledgements

Above all I want to thank my family who supported me in my studies, who were always there for me, and without whom none of this would be possible. I also want to thank my roommate4s who made the time spent in Vienna an unforgettable experience, allowing me to continue studying light heatedly. And of course I want to thank my girlfriend, who always supports me even though I am the worst at accepting it.

My special thanks go to Eric Mörth and Renata Raidou for advising me through this thesis with constructive feedback, the organization of the required datasets and their own valuable input.

Finally, I want to thank Clarence Yapp, Peter Sorger, Liam McLaughlin, Sanjay Jain and Ashley Kiemen for providing the datasets that were used during this thesis.

Kurzfassung

Der Fortschritt von Abbildungstechnologien ermöglicht die digitale Kartierung des menschlichen Körpers in sich ständig verbessernden Auflösungen. Mehrere Forschungsprogramme haben Abbildungen von verschiedenen funktionalen Gewebeeinheiten und auch Abbildungen von einzelnen Zellen erstellt, um den menschlichen Körper und seine Krankheitszustände gemeinsam digital zu kartieren. Neuartige Beiträge zu diesen Forschungsprogrammen eröffnen eine Vielzahl an Möglichkeiten, die funktionelle Komplexität der Organe sowie deren Voranschreiten von Krankheiten im Detail zu erforschen. Die Analyse der Konnektivität zwischen funktionalen Gewebeeinheiten oder einzelnen Zellen ist dabei ein wesentlicher Teil dieser Forschung, da angenommen wird, dass diese Elemente über verschiedenste Wege kommunizieren.

Diese Arbeit präsentiert zwei unabhängige Anwendungen, welche Forscher bei der Analyse von Konnektivität unterstützen. Erstens, eine installierbare Python Anwendung, mit welcher Oberflächen-Polygonnetze und eine Abstraktion der Konnektivität in Form eines Graphen generiert werden können. Hierzu wird zusätzlich eine Methode vorgeschlagen, welche die tatsächliche Konnektivität von unvollständigen röhrenförmigen Polygonnetzen empfiehlt. Zweitens, eine interaktive web-basierte Visualisierungsanwendung, welche die Konnektivitätsanaylse unterstützt, indem sie eine dreidimensionale Ansicht mit zwei Graph-Ansichten der Konnektivitätsabstraktion synchronisiert. Die Anwendbarkeit der Programme wird mit drei unterschiedlichen Datensätzen getestet und vorgeführt. Hierbei wird die Konnektivität zwischen Glomeruli und Nerven in der Niere, die Konnektivität von intraepithelialen Läsionen in Gängen der Pancreas mit dem umliegenden Nervensystem, sowie die Konnektivität zwischen den einzelnen Zellen eines Melanoms untesucht.

Abstract

With the advancement of imaging technologies, the mapping of the human body at continually increasing resolution becomes possible. Multiple research programs have achieved the imaging of various functional tissue units at cell resolution and even the imaging of single-cells in a combined effort to map out the human body and its disease states. Novel contributions to these programs open up a multitude of avenues to explore the functional complexity and disease progression of organs in even more detail. A vital part of this exploration is the analysis of connectivity between the functional tissue units or single-cells, as they are believed to communicate through diverse channels.

This thesis presents two independent tools that empower researchers to improve their connectivity analysis workflows. Firstly, an installable Python tool that generates surface meshes and a connectivity abstraction in the form of a network from segmented volumetric data. In the course of this, a method for suggesting the connectivity of incomplete tubular meshes is proposed. Secondly, an interactive web-based visualization tool that improves connectivity analysis by synchronizing a three-dimensional spatial view with two network views that incorporate the connectivity abstraction. The applicability of both tools is tested and showcased with data from three distinct fields by analyzing the connectivity between glomeruli and nerves in the kidney, the connectivity of intraepitheliel lesions in pancreatic ducts with the surrounding nervous system, and the connectivity between single-cells of a melanoma.

Contents

CHAPTER 1

Introduction

1.1 Motivation and Problem Definition

New technologies that allow the mapping and characterization of cell types at single-cell resolution have given rise to the creation of the National Institutes of Health (NIH) Human Biomolecular Atlas Program (HuBMAP) [HuB19], which aims to create a comprehensive three-dimensional (3D) molecular and cellular atlas of the human body. Research into human diseases has made similar strides. The Kidney Precision Medicine Program (KPMP) [dBAA+21], which tries to understand chronic kidney disease and acute kidney injury and the Human Tumor Atlas Network (HTAN) [RRRO+20], which intends to construct 3D atlases of human cancers, have both created interesting datasets at cell resolution. Novel contributions to these programs open up a multitude of avenues to explore the functional complexity of the various organs in the human body.

Research in the domain of biology and pathology is often tightly linked to visualization, since it allows for an in-depth analysis of the data acquired with novel imaging techniques. For example, analyzing tissue data at cellular resolution with a 3D visualization gives insight into the spatial development of precancerous lesions and pancreatic cancer [KBG+22]. Similarly, visualizing larger biological structures at cell resolution can greatly improve their connectivity analysis [MZS+24]. For instance, inspecting the neurovascular connectivity of nephrons in the human kidney reveals that glomeruli can be organized into communities based on interconnected nerve networks. Glomeruli on the border of those communities might act as control centers that synchronize responses to the disruption of fluid homeostasis.

Current visualization tools that can handle the huge datasets produced by cellular imaging techniques lack analysis functionality, are very expensive, too complicated to use or extremely specialized in a certain field. This leads to biologists and pathologists having to spend their resources in a field, in which they are no experts and therefore to subpar visualization results, which might also affect the quality of their analysis.

1.2 Aim of the Work

The aim of this thesis is the design and implementation of a visualization tool for the exploration of connectivity between cells or FTUs. The starting point for this work is 3D imaging data that has already been segmented into different cell types or functional tissue units (FTUs). The providers of the data are mainly interested in analyzing individual connected components of distinct types, as they believe them to be communicating with or through each other. For cells, this communication is assumed to happen based on their proximity, while FTUs might communicate through other FTUs like nerves or vasculature. Since the imaging and segmentation process has flaws attributable to technology or hardware limitations, quality assurance emerges as another step that needs to be developed on a case-to-case basis.

3D visualizations can improve the spatial analysis of the structures found in this data. However, due to densely packed biological tissues, such visualizations often suffer from a lot of occlusion, which can be especially problematic in connectivity analysis, where the exact tracing of complex tubular structures is often required. This can be mitigated by utilizing abstraction views that reveal the biological hierarchy or structure in the data.

Consequently, the following research questions are formulated:

RQ1. How can the connectivity between cells or FTUs at cell resolution be effectively abstracted for analysis?

RQ2. How can this abstraction be incorporated into a visualization tool that enhances research workflows and supports domain-specific analysis?

1.3 Contribution

The *contribution* of this thesis is threefold. The first contribution is a Python pipeline that empowers researchers to create surface meshes and a connectivity graph creation from segmented volumetric data with only basic knowledge of computer science and limited hardware. The second contribution is a quality assurance implementation for the provided kidney dataset. The third contribution is a generalized exploratory visualization tool for the results of the Python pipeline that can be used as a starting point for further analysis of the volumetric data.

As a proof of concept, the pipeline, quality assurance and visualization are tested by conducting a case study that focuses on the neuronal interconnectivity of glomeruli in multiple kidney datasets which have been provided by Sanjay Jain Lab [Lab].

[C1] Mesh and connectivity network creation

In the Python pipeline, all the connected components of each segmented type in the volume data are found and labeled. Those components are then used to create meshes

with isosurface extraction. In order to keep as close as possible to the original data, downsampling is avoided up to this point if it is computationally feasible. The meshes are then simplified, while preserving their shape as much as possible, so they can be stored without needing access to excessive storage options. Depending on the dataset, either specific or generic quality assurance steps are employed. Afterward, the connectivity between the components is calculated with a Euclidean distance matrix and is transformed into a graph. To tackle the scalability problem, this graph can then be clustered based on interconnectivity.

[C2] Quality assurance and connectivity suggestion for kidney data

The quality of the data is improved by enabling the target user to split falsely merged entities and improve the continuity of missing mesh sections by utilizing a framework that suggests connectivity of tubular meshes like nerves and vasculature based on their orientation.

[C3] Exploratory visualization of the connectivity network and meshes

To achieve this, three individual views are combined in one interactive visualization: A 3D view that depicts the actual structure and spatial relationships of the components. A general overview of the whole connectivity network abstraction that helps the user to orient themselves, and a detail view that allows the user to inspect the connectivity of individual components. The three views are interactively linked so that hovering over a region of interest in one view highlights the semantically related components in all the other views while providing basic information through labels and tooltips.

1.4 Outline

This thesis is structured in the following way: Chapter 2 offers an overview of biological imaging techniques that enable the capturing of the data used in this thesis. It continues with an introduction of the organs that were the main focus of the case studies and ends with a description of the datasets that were used for the development as well as the case studies. Chapter 3 discusses related work in the fields of biology, pathology, visualization and connectivity abstraction. The design process of the application is discussed in Chapter 4 by comparing different approaches in regard to their advantages and disadvantages. Chapter 5 concerns itself with the actual implementation details of the pipeline and visualization tool. The results and case studies are presented and critically reviewed in Chapter 6. In Chapter 7, the thesis is summarized, limitations are addressed, and future work is proposed.

CHAPTER 2

Clinical Background

This chapter introduces the biology that is relevant to this thesis based on the provided datasets, followed by the imaging techniques used to create said datasets and finally the datasets themselves.

2.1 Relevant Biology

This section briefly describes the localization and function of the biological systems in the human body that host the entities of interest, as well as the biology surrounding those entities. These systems include the kidney, the pancreas and the skin.

2.1.1 Glomeruli in the Kidneys

The bean-shaped pair of kidneys found in the human body are organs that belong to the upper urinary tract and are positioned high on the posterior abdominal wall. The right kidney is located slightly lower than the left kidney, most likely due to the mass of the liver that lies above it [Mah19a]. The vasculature, nerves, lymphatics and ureter all enter the kidney through the renal hilum and spread out from there (see Figure 2.1). The main regions are the inner medulla and the outer cortex, which are connected by straight tubules and collecting ducts. The network of tubules in the medulla creates multiple pyramidal shapes that have their base opened to the cortex and point towards the hilum where the tubules end into papillae which in turn drain via collecting ducts into the ureter. Between those pyramids and comprised of vessels that enter and exit the cortex are the renal columns. A kidney is usually divided into six to eight lobes that consist of a medullar pyramid enclosed by renal columns [SPL24].

The cortex is the starting point for the functional unit of the kidney, which is called the nephron. Studies have shown that a human kidney has approximately an average of 1 million nephrons, but their number has a very high variability [BDDD$^+$11]. Figure 2.2

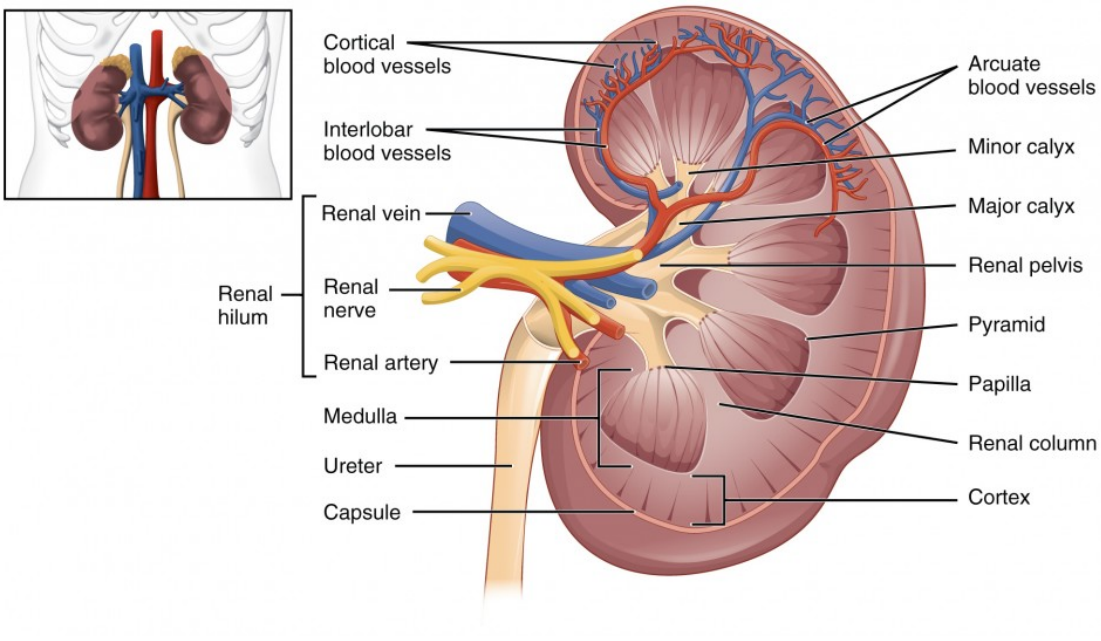

Figure 2.1: The labeled anatomy of a kidney (adapted from LibreTexts [Lib])

shows the anatomy of a single nephron. From the vasculature of the cortex, an afferent arteriole supplies blood to the glomerulus. The glomerulus is a network of capillaries that is surrounded by the glomerular capsule, which combined forms a renal corpuscle. The basic function of the renal corpuscle is to filter metabolic waste products from the blood. From the glomerulus, the blood is drained along the efferent arteriole, where it branches out to create the blood supply for the renal tubules. The tubules begin from the renal corpuscle in the cortex, descend into the medulla, ascend back to the originating corpuscle and finally drain urine via a collecting duct into the ureter [SPL24].

In a healthy human, about twenty percent of the cardiac output is filtered through the kidney's glomeruli every minute [Mah19a]. Besides the excretion of metabolic waste products, the kidney governs electrolyte regulation and the acid-base balance. In combination with the adrenal gland, it also has a major effect on the intravascular volume through the renin-angiotensin-aldosterone system. This in turn leads to the kidney having substantial control over blood pressure [SPL24].

The vasculature and tubules of the kidney are both innervated by sympathetic (efferent) nerves and sensory (afferent) nerves. Activity in the efferent nerves decreases the glomerular filtration rate and, together with the efferent nerves, they form a feedback loop that allows the brain to partially regulate the filtration in the kidney. The innervation stretches to the capillary network that makes up the glomerulus, but its functional significance is yet unknown [OTV21].

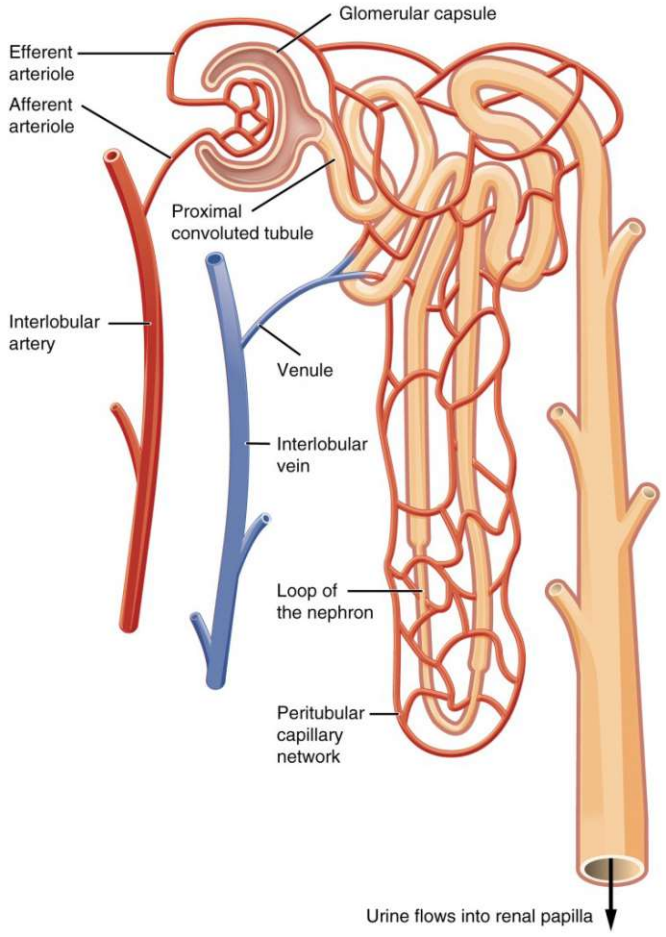

Figure 2.2: The labeled anatomy of the nephron with its connecting vasculature and tubules. (adapted from LibreTexts [Lib])

2.1.2 Intraepithelial Neoplasia in the Pancreas

The pancreas is an elongated exocrine and endocrine gland that is positioned behind the lower portion of the stomach. From left to right, it can be divided into four sections: The head that is nestled in the curve of the duodenum, the neck is a thin combining section, the body is the middle part which is in front of the superior mesenteric vessels and the tail that is the thin tip close to the spleen [Lon14]. Blood for the head and neck is supplied and drained by the pancreaticodudodenal vessels, and for the body and tail by branches of the splenic vessels. Interlobular ducts converge at the main duct of Wirsung and the accessory duct of Santorini, both of which open into the duodenum. Even smaller Intralobular ducts link the acinar tubules to the interlobular ducts. The acini are clusters of cells that make up the exocrine function of the pancreas by excreting digestive enzymes directly into the ductal system. The exocrine function of the pancreas stems from the islets of langerhans, which release glucagon, insulin and hormones into

the circulatory system [Mah19b].

One of the most lethal cancerous tumors a human can have, is an invasive adenocarcinoma of the pancreas [STAHAR18]. In an effort to reduce the mortality of this tumor, researchers are investigating the effect of detecting and treating non-invasive panrceatic neoplasia. A neoplasia in the ductal system of the pancreas is called a pancreatic intraepithelial neoplasia (PanIN) and has found to be a precursor for ductal adenocarcinoma. PanIN are classified into four grades: PanIN-1A, -1B, -2 and -3. Further research has shown that harmless PanIN lesions can progress into deadly invasive cancers [HBFM08]. Investigating this precursor in 3D, as opposed to 2D, can give more insight into what happens during the progression of the lesion [KDB+23].

2.1.3 Melanoma of the Skin

The skin is the largest organ of the human body and covers most of its surface. It is made up of three layers from outside to inside: the epidermis containing primarily keratinocytes and dendritic cells, the dermis composed of collagen and the subcutaneous tissue which contains conglomerations of lipocytes. The thickness of those layers varies depending on the area of the body. The tissue in the dermis connects the epidermis to the subcutaneous tissue and also contains sweat glands, hair follicles with oil glands, vasculature, lymphatic vessels and apocrine glands for sweat and scent release [KKG11].

Melanocytes are melanin producing cells that can be found in the epidermis. When such a melanocyte undergoes a malignant transformation, the result is a skin cancer called a cutaneous melanoma. About 0.7% of all cancer deaths worldwide are caused by cutaneous melanoma, with sunburns caused by ultraviolet radiation from sun exposure being a high risk factor for the transformation [SVAB+18]. Research has shown that cell to cell interactions have an effect on the emergence of cancers like cutaneous melanoma [YNZ+24]. Analyzing these interactions in the spatial context of connectivity could give more insights into the transformation process.

2.2 3D Reconstruction Techniques

This section introduces the 3D reconstruction techniques that were applied to acquire the used volume data described in Section 2.3. The approach for the kidney data was volumetric imaging with light sheet fluorescence microscopy (LSFM) of samples that have undergone a tissue-clearing process. This allows whole-body imaging without the need for cutting the tissue in sections but is limited by the penetration depth of the required antibody staining and densely fibrotic tissues. The technique for the pancreas data was the method of serially sectioning an embedded sample, hematoxylin and eosin (H&E) staining it and combining the digitized slices back together with image registration. This enables the labeling of more organs and can be used on fibrous tissue, but the z-continuity is a lot worse, which can lead to distorted 3D reconstructions of tubular tissues like vasculature or nerves. The images for the cell data were obtained by using thick section

tissue based cyclic immunofluorescence (t-CyCIF) with laser-scanning confocal microscopy (LSCM). To get the volume data, those images were then stitched back together utilizing a registration method. This technique makes highly multiplexed imaging at cell resolution possible, but is limited in the thickness of the sample.

2.2.1 Volumetric Imaging with Tissue Clearing

When it comes to visualizing thick biological tissues with volumetric light microscopy, an essential part is the transparency of the specimen that is investigated. Most biological samples from humans or animals are inherently opaque and must therefore be turned transparent by creating a uniform refractive index before any kind of microscopy that utilizes light can be employed effectively. The method behind this is often called 'tissue clearing' and several approaches have been developed. These approaches can generally be categorized in the following families [VPR21]: Organic solvent-based methods like 3DISCO [EBJ+12], hydrogel-based methods like CLARITY [CWK+13] and aqueous methods like CUBIC [STP+14]. Figure 2.3 shows an adult mouse that was perfused with a CUBIC cocktail for 14 days compared to a mouse that was perfused with a phosphate-buffered saline solution to maintain a constant pH.

After tissue clearing comes volumetric imaging in the form of microscopy. The method used for the kidney dataset was LSFM. LSFM was developed in an effort to create an adequate four-dimensional (x, y, z and time) microscope and uses plane-wise illumination with wide-field fluorescence detection to optically section a volumetric sample. There is a great variety of light-sheet microscopes and the best results are achieved with custom solutions [PH17]. Figure 2.4 shows the 3D reconstruction from an LSFM image of a kidney sample that includes glomeruli, nerves and collecting ducts. This image of the week from the HuBMAP website also inspired this thesis.

2.2.2 Registration of H&E Stained Serial Sections

This technique uses serially sectioned slides that are cut from a sample with a microtome. It is closely linked to digital pathology, where preserved and processed sections are digitized with a slide scanner and stored in a biobank. Those samples are already often H&E stained, which in combination with modern segmentation algorithms allows the identification of multiple tissues. Deep learning algorithms to label the tissue types can be especially effective [KBG+22]. Due to the mechanical nature of the cutting process, the tissue of sequential sections can get rotated, translated, folded, split and stretched. To correct this during 3D reconstruction, a registration step that maximizes the cross-correlation between sections is employed [KDB+23].

2.2.3 Stitching of Thick Section t-CyCIF

This approach is based on cyclic immunofluorescence [LFSCS16] and uses an imaging technique called t-CyCIF that was adapted for thick tissue sections and is able to create highly multiplexed immunofluorescence images at cell resolution with an optical

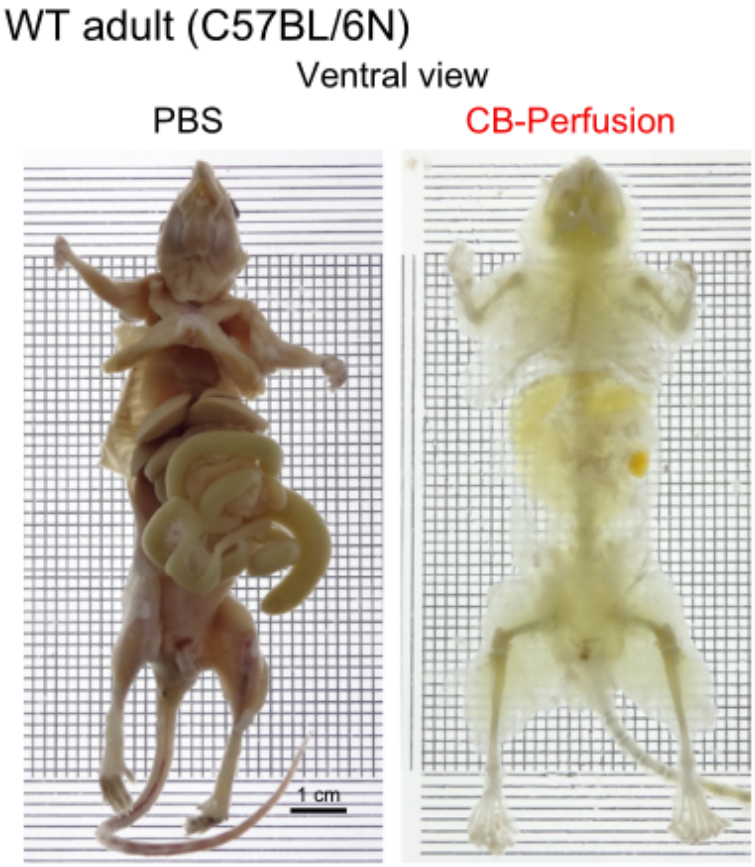

Figure 2.3: The whole body of an adult mouse which has undergone 14 days of perfusion with phosphate-buffered saline solution (PBS, left) compared to CUBIC-1 (CB-perfusion, right) [TKS+14].

microscope. This is achieved by cyclic bleaching, staining and imaging of pre-stained tissue slides. For the imaging, laser-scanning confocal microscopy (LSCM) is used.

LSCM passes the transmitted as well as emitted light through a pinhole, which increases the optical resolution that can be achieved. Since most of the fluorescent light is blocked by the pinhole, the signal intensity is decreased, which has to be compensated with a more sensitive detector and longer exposure times. The sample is scanned point by point while also adjusting the focal point [BOS+18].

To create a 3D reconstruction, the images acquired in the cycles are stitched together in a 3D registration process that can also include illumination correction. The result is a whole-slide image containing all the captured channels.

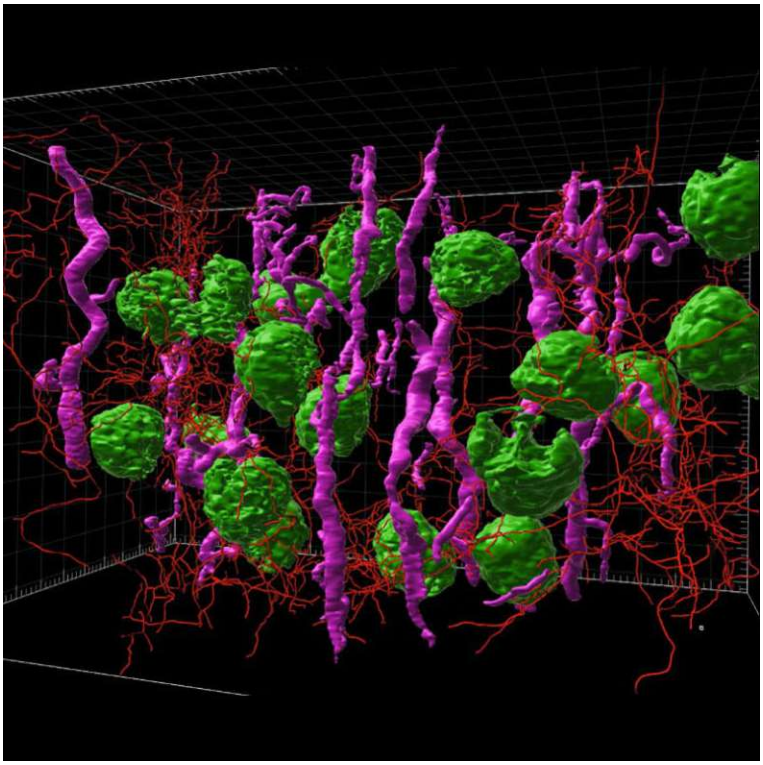

Figure 2.4: A 3D reconstruction from light sheet fluorescence microscopy images of glomeruli (green), nerves (red) and collecting ducts (pink) published as an image of the week on the HuBMAP consortium website [Pra23].

2.3 Used Data

For the development and design of this work, multiple volume datasets of various FTUs and cells were used. They were created by different 3D reconstruction techniques. The main focus was on the kidney datasets since they used volumetric imaging with LSFM which allowed the clean reconstruction and tracking of the tubular nerve connections between glomeruli.

2.3.1 Kidney Datasets

A total of 10 kidney datasets containing binary masks of FTUs representing nerves, vasculature, collecting ducts and glomeruli were used in the course of this thesis. The datasets were provided by Sanjay Jain Lab [Lab] and were created in the course of a paper on 3D nephron connectivity by McLaughlin et al. [MZS+24]. The samples for the datasets were obtained from deceased donors or donors that had nephrectomy surgery. The size of the dissected tissue was about $2cm \times 3-5mm \times 1-3mm$ and usually contained both the cortex and medulla. Tissue clearing was done with the CLARITY SHIELD Active clearing protocol [PSC+19]. Antibody staining was used on the glomeruli,

nerves, collecting ducts, tubules and nuclei for immunofluorescence. The images were then captured with the light-sheet microscope ZEISS Llightsheet 7 [Mica]. The whole sample was imaged with a 5× objective and after further inspection, regions of interest were imaged at 20× zoom. Z-stacks gained from the ZEISS Lightsheet 7 were stitched together with either the microscopy software ZEN Blue 3.3 [Micc] by ZEISS or the stitching software Stitchy [Bio] by Translucence Biosystems, depending on the file size of the entire volume. To make segmentation efficient, the X and Y dimensions of the volumes were downsampled 4 times as $1/16^{th}$ of the resolution. Finally, the binary masks were created by applying Cellpose 2 [PS22], Cellpose 3 [SP24] or the ImageJ [Col07] extension Labkit [ADS+22].

2.3.2 Pancreas Dataset

During development, one pancreas dataset containing volume data of various labeled FTUs, including nerves, smooth muscle, normal ductal epithelium and PanIN, was investigated. The dataset was provided by Johns Hopkins University Institute for Nanobiotechnology [JHU] and was created in the course of a paper on 3D reconstruction of pancreatic tissue at cellular resolution by Ashley Kiemen et al. [KBG+22]. The used sample contained pancreatic parenchyma with precancerous lesions, was fixed with formalin and embedded with paraffin. After sectioning, the sample was stained with H&E and digitized at 20× magnification with a resolution of 0.5 µm × 0.5 µm × 4 µm ($XxYxZ$) per section. The images were then aligned in a two-step registration process, with whole-field rigid-body registration followed by an elastic registration method [KLV+18]. To differentiate the FTUs a cell detection algorithm was applied that utilizes color deconvolution, normalization and an algorithm for particle tracking [WHC+12]. In the end, the FTUs were labeled with DeepLab semantic segmentation and a pre-trained ResNet50 network [CZP+18].

2.3.3 Cell Dataset

To test the Python pipeline capabilities regarding datasets with a large number of individual components, two melanoma datasets containing up to 43,000 individually segmented cells were used to reconstruct the 3D meshes and create a connectivity graph. The dataset was created in the course of a paper on 3D analysis of immune states in human tissue by Yapp et al. [YNZ+24], after which they made the data freely available on the NCI Human Tumor Atlas Network Portal [Nat]. The used samples were 35μm thick sections of melanoma which were prepared for image acquisition with thick section tissue based cyclic immunofluorescence (t-CyCIF). The imaging was done with the LSCM ZEISS LSM 980 with Airyscan 2 [Micb]. Regions of interest were captured at 40x zoom with 0.14 microns per pixel in the X and Y dimensions and 0.28 microns per pixel in the Z dimension with about 170 optical planes. The images were stitched together in a registration step that utilizes ZEN 3.7 [Micc] and MATLAB [Mat]. Segmentation of the individual cells was performed with Cellpose [SWMP21] and a custom preprocessing step.

CHAPTER 3

Related Work

3.1 Visualization at Cell Resolution

To understand the trillions of cells that comprise the human body, a consortium of currently 42 contributing sites are creating HuBMAP [HuB19], a global atlas of the human body at the cellular level. Furthermore, the HuBMAP intends to develop tools that help visualize and analyze cells with a focus on deoxyribonucleic acid (DNA), ribonucleic acid (RNA), proteins, phenotypes and functional tissue units in order to understand the processes involved in homeostasis aging and disease [JPS+23]. Keller et al. [KGM+24] created Vitessce, which is an open-source framework that uses Viv [MGP+22] to create interactive web-based visualizations for the exploration of multimodal and spatially-resolved single-cell experiments. It is serverless, can be embedded into Jupyter Notebooks or RStudio, supports a wide variety of formats, and offers a plugin system for external data loaders and views.

The authors of the datasets used in this thesis have all employed their own strategy for visualizing the data themselves. McLaughlin et al. [MZS+24] created a neurovascular nephron connectivity network from LSFM images of the kidney (see Section 2.3.1) and analyzed it over the human lifespan. By investigating their 3D reconstruction, they were able to categorize parts of their network into various motifs [MSOI+02] and identify control centers they call “Mother Gloms“ (Figure 3.1).

Kiemen et al. [KBG+22] have developed a quantitative 3D reconstruction of large tissues at cellular resolution (CODA), a method to reconstruct large tissues at subcellular resolution from serially sectioned H&E tissue sections (see Section 2.3.2). Figure 3.2 shows part of their visualization, which indicates that multiple anatomically distinct PanIN can grow in small or large ducts.

Yapp et al. [YNZ+24] have analyzed high-resolution images from thick section t-CyCIF LSCM (see Section 2.3.3) in 3D and created visualizations of intra- and extracellular

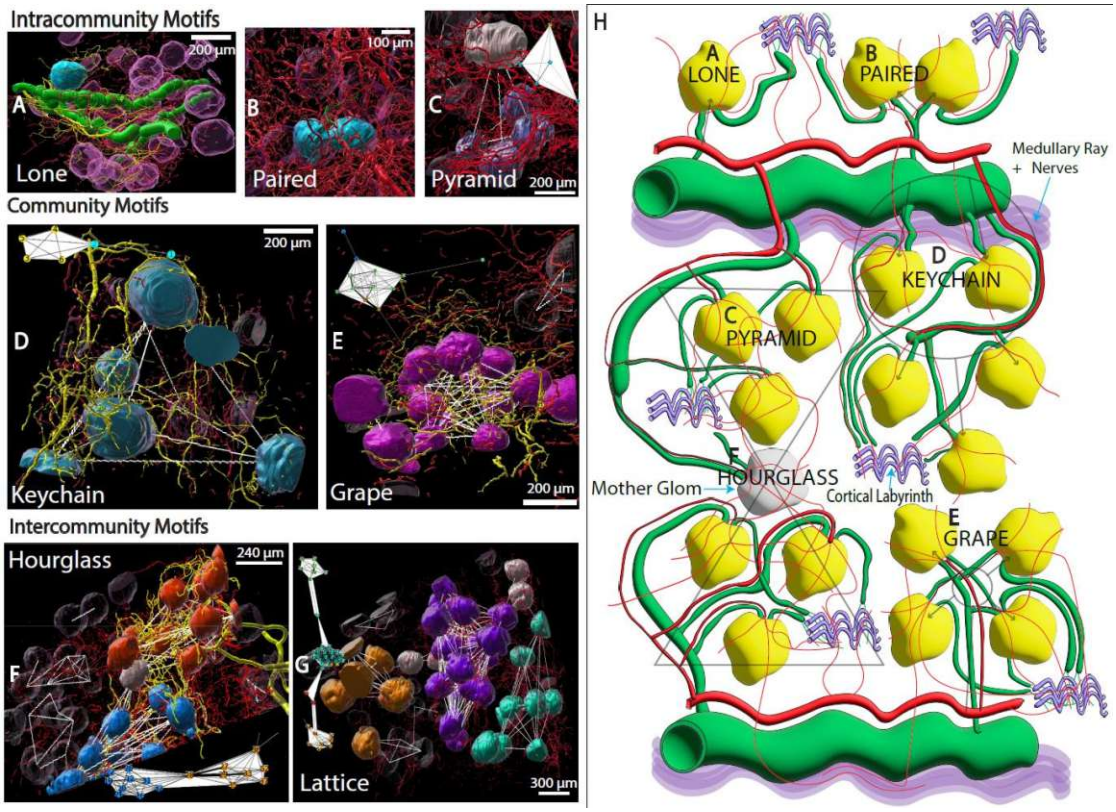

Figure 3.1: 3D reconstructions of neuro-glomerular network motif type examples (A-G) together with a cartoon-like abstraction of the motif types in relation to other nephron structures (H) [MZS+24].

morphology. The focus of their work was on analyzing cell interaction in early tumor formation. For this work, they categorized cell-cell interaction into three types and created an abstract visualization of cell interactions. The three types—Direct, Aposition, and Lymphonet—are displayed as line variations in Figure 3.3.

Besides these visualizations, there has been a multitude of other attempts to visualize cells and FTUs of different organs and from a wide range of data. Ghose et al. [GJM+23] reconstruct skin in 3D from micro-computed tomography images at cell resolution, analyze spatial relationships and compare it to 2D analysis. They investigate cell density within a specific distance, the distance of immune cells to endothelial cells (vasculature), and the distance of UV-damaged keratinocytes to the skin surface.

ClusterMap developed by He et al. [HTH+21], is a framework for clustering RNAs into subcellular structures, cell bodies, and tissue regions in 2D and 3D. To support this, they employ density peak clustering on mRNA transcriptomic imaging spot data. The clusters are then typed by gene expressions of their spots and used to construct a 2D or 3D visualization. Cell graphs created with triangulation are used to identify neighboring

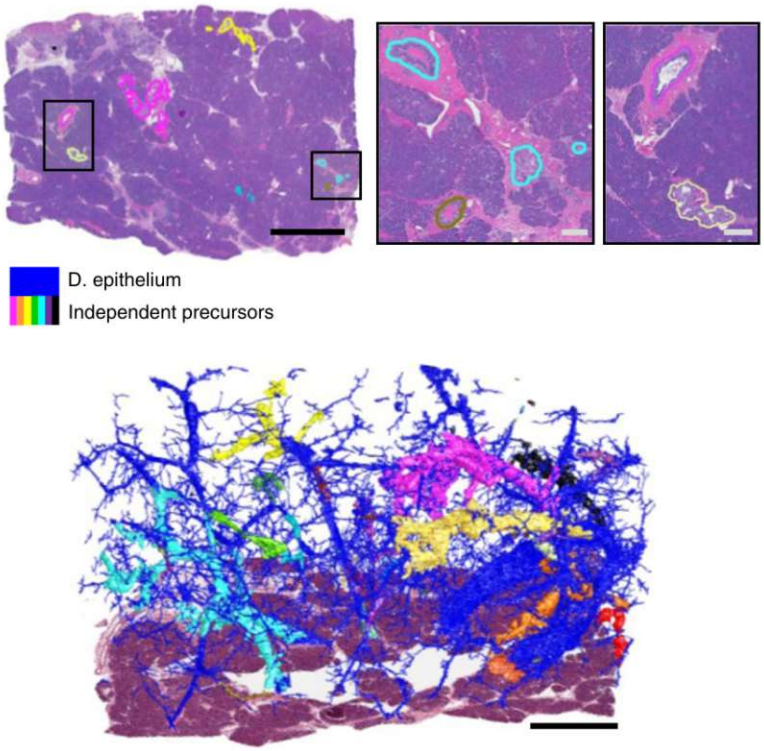

Figure 3.2: Colored and labeled PanIN in H&E stained serial sections (top) and a 3D reconstruction together with the ductal epithelium of the pancreas (bottom) [KBG+22].

cells.

Kuett et al. [KCÖ+22] used sections from ultramicrotomy and performed imaging mass cytometry to create 3D volume renderings of human breast cancer samples. They analyzed them and found that measurements of spatial neighborhoods can differ between 3D and 2D, highlighting the importance of 3D analysis in tumor localization.

There have also been visualization tools that solely focus on 2D exploration. With Scope2Screen, Jessup et al. [JKW+21] present a Focus+Context method for the pathological exploration of multivariate tissue imaging data. Focus+Context is an interaction technique where multiple levels of details are shown by displaying the focus in the context [Car99]. They employ various techniques for channel exploration like a multi-channel or split screen lens while also augmenting features through single-cell histograms, radial single-cell plots, or a classification result legend. Furthermore, they implement a visual annotation method that allows to save views of regions of interest for reference and later exploration. Warchol et al. [WKN+22] build upon this framework and create Vicinity, which is an analysis tool with coordinated views for cell interaction in multiplexed tissue imaging data. Neighborhood composition and spatial arrangements of cells can be queried and analyzed with respect to their statistical significance. Neighborhoods are quantified

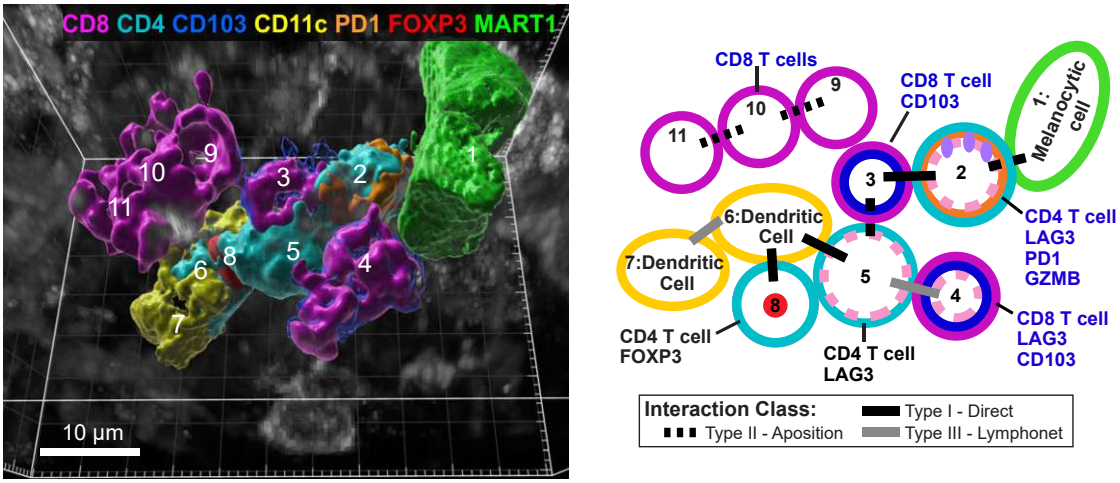

Figure 3.3: A 3D surface rendering (left) of 11 cells from a melanoma close to the dermal-epidermal junction, with a schematic abstraction of cell interconnectivity and their types (right) [YNZ+24].

based on the occurrence of specific cell types in a defined radius that forms a vector that can be compared between all the cells in the data.

While many visualizations focus on spatial relationships, some employ abstractions of functional components. Anchang et al. [AHB+16] describe the use of SPADE [QSB+11], a density-based algorithm used for the visualization of single-cell cytometry data and extraction of hierarchy between similar cells. They compare SPADE to other recent approaches and provide a guide that combines the benefits of SPADE with t-distribution stochastic neighborhood embedding.

Research of neurons in the brain at cellular resolution has already led to 3D reconstruction methods and explorative visualizations. Early works include V3D by Peng et al. [PRL+10], an efficient system that, through the help of the plugin V3D-Neuron enables the 3D reconstruction of neuronal structures from high-resolution brain images. With VIMO, Troidl et al. [TWC+23] have developed an analysis tool for the connectome that uses motif drawing to create queries and reduce the complexity of the massive network. It visualizes neuronal connections in 3D at a cell level and employs simplification to reduce clutter and connectivity highlighting with exploded views to further improve comprehension of 3D structures.

3.2 Connectivity Abstraction

When inspecting biological connectivity at a cellular level, the sheer number of connections often calls for an abstraction to make the analysis feasible. Special care has to be taken when creating a visual abstraction since in the process of abstracting, data is emitted for the sake of comprehension [VCI20]. Careful analysis of the use case can help understand

which data to emit and which to keep. The brain with its estimated 100 trillion neuronal connections is a prime example of this [Zim11]. Figure 3.4 shows a visualization tool by Jianu et al. [JDL11] that abstracts and filters a tractogram created with diffusion MRI. They try to simplify the inherently dense and complex structures that are tractograms while keeping the anatomical context. The abstraction and the original tractogram are displayed in connected views.

Figure 3.4: An interactive stand-alone visualization tool of a tractogram with planar 2D neural maps (left) and an explorative web interface (right) [JDL11].

Yeh et al. [YSD+19] show how assigning streamlines of tractograms created from diffusion MRI to network nodes in the connectome with the purpose of brain parcellation can be highly consequential. Depending on the mechanism employed, the result varies significantly.

Graph theory is a powerful tool with which connectivity between any kind of subject can be represented and analyzed in the form of a network. Bullmore et al. [BS09] describe how graph theory can be used for the analysis of structural and functional systems in complex brain networks. They elaborate on different network measures that can be employed like node degree, centrality and robustness, path length, efficiency, and modularity. They conclude that anatomical networks (not just brains) are sparsely connected, while the physical connections between nodes are close to minimal. Nelson et al. [NB21] explain graph theory in general and go into detail about graph theory metrics like node degree, paths, path characteristics, centrality, graph motifs, clustering coefficient, and communities which can be useful for any network analysis process.

Since FTUs are communicating with each other through biochemical processes, measures like centrality that are employed for social networks might as well be relevant for biological networks. While investigating the concept of centrality Freeman et al. [F+02] found that there are three conceptual foundations for centrality measures. The *Degree* is used as an index for communication activity, *Betweenness* as an index for potential control, and

Closeness as an index of independence or efficiency. They assign three measures to each foundation, where two are for the individual nodes and one is for the overall graph. The overall graph measures of the three foundations agree for the extreme cases where the centrality is either 0 or 1. Care has to be taken when deciding which concept of centrality should be employed.

Entourage, developed by Lex et al. [LPK+13], is a visualization tool that employs graph theory and abstraction to visualize interconnectivity between biological pathways. It tries to further the analysis of crosstalk between multiple pathways by showing contextual subsets of a selected node in multiple linked views. Those contextual subsets do not show their whole biological pathway but only the relevant nodes based on mapped experimental data. By utilizing stubs for both nodes and multiple windows they show connectivity between nodes without cluttering the view.

Keller et al. [KEC06] tried to uncover whether matrices or node-link diagrams are better for visualizing graphs that represent connectivity. They investigated the effect that size, density, and directionality of connectivity models have on the comprehension of a matrix visualization and concluded that both size and density affect the response time and error rates of users. For node-link diagrams, the prior knowledge of the subject has a big impact on how well a user can read information. The choice of the best representation seems to rely heavily on the detailed properties of the connectivity model and the tasks that have to be carried out.

With Graffinity, Kerzner et al. [KLS+17] have developed a novel visualization framework that combines both matrices and a node-link diagram to represent connectivity in large graphs (see Figure 3.5). They use queries, a connectivity matrix with juxtaposed node properties, a node table with aggregation of paths crossing the node, a path list, and a node-link diagram in multiple linked views. The large data is handled by only visualizing the queried information.

In the biological context of vasculature, nerves, and other tubular FTUs an extraction process of the general connectivity from the volume or mesh data is necessary. One approach to achieve this for tubular meshes is centerline extraction. Yahya-Zoubir et al. [YZHSO16] developed an automatic algorithm that extracts the centerline from a general 3D tubular mesh. The algorithm defines a moving section and a reference section at the ends of the tubular geometry respectively. It then advances the moving section by traveling along the mesh to the adjacent vertices and creates a center point for every section. Another method for vasculature meshes by Wei et al. [WWL+18] tries to abstract the vasculature into segments of cylindrical shape. The mesh is segmented into branches with K-means fuzzy clustering [Dun73]. They then extract the centerline for each segment by iteratively finding an optimal cut plane and the center point of the points in this plane. To connect the centerline at the branching points they first smooth the centerline, followed by thinning and re-centering.

Another more general approach is creating a curve skeleton of a 3D geometry. Au et al. [ATC+08] developed a curve skeleton extraction method by iterative contraction of

the mesh. This is accomplished through Laplacian smoothing with global positional constraints and subsequent connectivity surgery. Abdollahzadeh et al. [AST21] use a cylindrical shape decomposition algorithm for the segmentation of tubular voxel objects like axons and vasculature networks. The algorithm finds a curve skeleton by employing a cost function that takes the distance field to the object's surface into account. The curve skeleton is partitioned into skeleton branches and a skeleton graph is created. The object is then decomposed into its branches by cutting away the intersections. In the last step, the missing connections are reconstructed with generalized cylinders. Tagliasacchi et al. [TZCO09] have created a method that extracts the curve skeleton from an incomplete point cloud and also allows surface reconstruction of missing data. For this, they utilize the rotational symmetry axis of the oriented point set to create the curve skeleton.

3.3 Network Visualization

There is a wide variety of visualization techniques for networks. While there is not one definitive solution for all use cases, some attempts to create guidelines and studies into applicability have been made. In the context of biological networks, Marai et al. [MPB+19] have created ten simple rules that should be considered when creating network visualizations for communication. They suggest that one should determine the purpose of the visualization, consider alternative layouts, and be aware of spatial implications. Colors, labels, markings, and channels should be used appropriately and with consideration of the

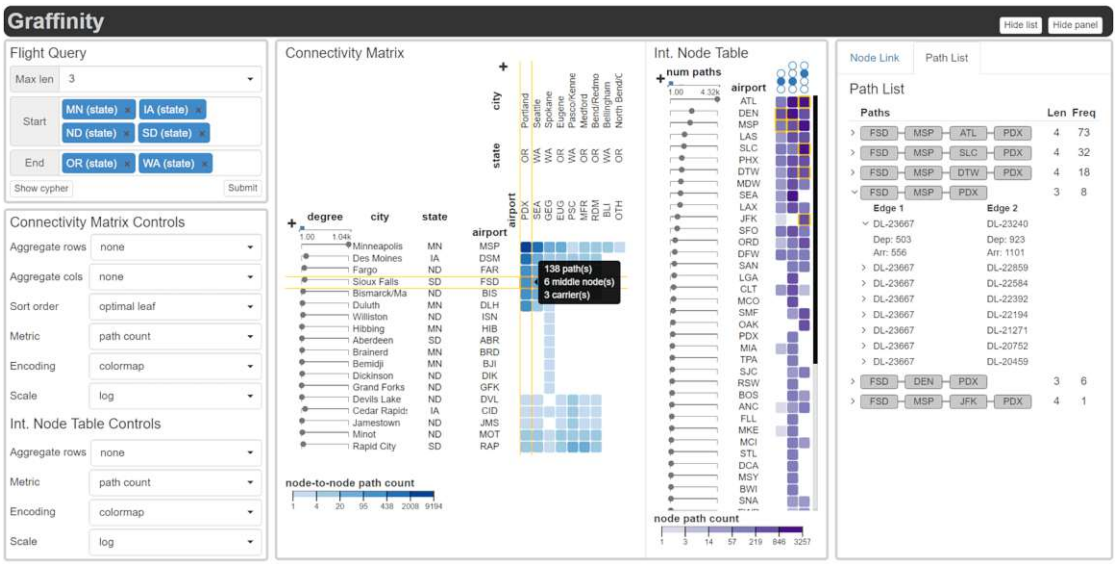

Figure 3.5: The connectivity visualization tool Graffinity shows the result from a specific query on a flight dataset. It combines a connectivity matrix with a node table and interactively highlights information from the user's selection through a path list and node-link representation [KLS+17].

viewer's possible individual traits and impairments. The right level of detail combined with separated layers and views can help the overall comprehension.

Node-link diagrams are a broadly used technique when it comes to network visualization. Saket et al. [SSKB14] tried to evaluate different flavors of diagrams that incorporate nodes and links. Their study investigates the accuracy and speed of node-based tasks, network-based tasks, and group-based tasks on node, node-link, and node-link-group diagrams. It concludes that node-link-group diagrams are 8% more accurate than node and node-link diagrams on all tasks and that tasks were performed 15% faster when using node-link and node-link-group diagrams in comparison to node diagrams. These results contradict other studies in similar but subtly different settings.

Nobre et al. [NMSL19] discuss state-of-the-art visualization approaches for multivariate networks including node-link layouts, tabular layouts, tree layouts, and also view operations. Table 3.1 shows a table they compiled from analyzing 210 papers that were published by the visualization research community. It summarizes the applicability of various layouts and view operations for visualizing different network types, sizes, topological structures as well as node and edge attributes. Their research clearly shows that visualizing large networks (> 1000 nodes) is a challenge for any technique if it stands on its own and is not complemented with additional views and interactive methods.

A sequence of edges that connects a sequence of vertices in a graph is called a path. Such paths are an important part of analyzing connectivity in a communication network, for example when calculating the betweenness centrality [F+02]. In a crowdsourced study, Nobre et al. [NWHL20] investigated the advantages and disadvantages of node-link diagrams with on-node encoding and adjacency matrices with juxtaposed tables. They formulated 9 hypotheses, which they tested by comparing the differences of both approaches on 16 different tasks. In the course of this, they confirmed their hypotheses that node-link diagrams perform more accurately and faster for all tasks related to paths.

Partl et al. [PGS+16] developed Pathfinder, a visualization solution for paths in large graphs. It employs queries to handle large data and only shows the parts the user is interested in. The queried data is displayed in a ranked path list that is augmented with the node's attributes and allows sorting by multiple constraints and aggregations that the user can devise based on their need. The topology of the queried nodes and their paths is displayed in a node-link diagram and statistics are displayed in an additional view.

3.4 Summary

For the brain, there already exist general tools like tractogram visualization applications [JDL11] and also more fine-grained tools that allow the connectivity analysis at the cell level like *Vimo* [TWC+23]. For other organs of the human body, new imaging and 3D reconstruction techniques allowed the creation of visualizations of FTUs like the glomeruli in the kidney [MZS+24] and the PanIN in the pancreas [KDB+23] as well as a

		Size			Type				Node Attributes				Edge Attributes				Topolog. Structures				
		Small <100 nodes	Medium (<1,000)	Large (>1,000 nodes)	Complex (sparse)	Complex (dense)	Layered/K-Partite	Trees	Few (<5)	Several (≥5)	Homog. (1 type)	Hetero. (>1 type)	Few (<3)	Several (≥3)	Homog. (1 type)	Hetero. (>1 type)	Single node/edge	Neighbors	Paths	Clusters	Entire/sub network
Node-Link Layouts	On-node/edge encoding	3	2	1	3	1	3	3	2	1	3	2	2	1	3	1	3	3	2	2	2
	Attr.-driven faceting	3	1	1	3	1	3	1	3	1	3	3	2	1	2	1	3	2	1	1	1
	Attr.-driven positioning	3	1	1	3	1	1	1	3	1	3	1	2	1	2	1	3	2	1	1	2
Tabular Layouts	Adjacency matrix	3	1	1	2	3	2	1	2	3	3	2	3	2	3	2	3	3	1	3	2
	Quilts	3	1	1	3	1	3	3	3	3	3	3	3	3	3	2	3	3	2	2	2
	BioFabric	3	1	1	3	3	2	1	3	3	3	3	3	3	3	3	3	1	1	1	2
Implicit	Inner nodes & leaves	3	2	1	0	0	0	3	3	1	3	1	0	0	0	0	3	3	3	0	3
	Leaves	3	2	2	0	0	0	3	3	1	3	1	0	0	0	0	3	2	1	0	3
View Operations	Juxtaposed	3	2	1	3	1	3	3	3	3	3	3	3	3	3	3	2	1	1	2	2
	Integrated	3	2	1	3	1	3	3	3	3	3	3	2	2	3	3	3	3	3	1	2
	Overloaded	3	2	1	3	1	3	3	3	1	3	1	1	1	1	1	3	3	2	3	2

Table 3.1: Nobre et al. [NMSL19] developed a scoring system for the performance of visualization techniques on different network tasks and types (**0**: no support; **1**: poor support; **2**: limited support that might require interaction; **3**: very good support)

visualization of individual melanoma cells [YNZ+24]. Like in the brain, there is a need for understanding the inherent connectivity between the individual components of those visualizations, but currently, there is no generalized tool that would enable researchers to analyze this.

CHAPTER 4

Application Design

The previous chapters presented the motivation, background and state-of-the-art that are relevant to this thesis. This chapter discusses the design process for the resulting application in detail. It explains various approaches in a comparative fashion, while laying out successful as well as unsuccessful discoveries. In the course of this, it links different aspects of the application to the contributions in Chapter 1.

4.1 Application Overview

This section provides a brief overview of the components that comprise the network generation pipeline and visualization application. The following sections go into more detail about the individual aspects that are shown here and explain the design process and why they were chosen.

Figure 4.1 shows the simplified functional overview of the final network generation pipeline. The starting point and input for the pipeline is biological volumetric data. This data can be either a binary mask depicting a singular functional tissue unit (FTU)/cell type or a labeled dataset. The labeling can either represent a segmentation of different FTUs/cell types or a segmentation of components (e.g. single cells). If the input data contains different FTUs or cell types, the mesh creation steps of the pipeline need to be run for every FTU or cell type that is of interest.

The first step of the pipeline is to label the components that are connected three-dimensionally in the volumetric data. This step can be skipped if the input already is a segmentation of components. Multiple statistics like the bounding box, centroid and voxel count are calculated for every component and they are then sorted by size. According to user input, the 3D meshes of the k largest components are generated from the highest feasible volumetric data resolution. In order to keep the file size manageable, the meshes are simplified while keeping their 3D shape intact. Depending on the dataset,

quality assurance steps have to be taken before continuing. The user can then specify connection metrics to calculate between various FTUs or cell types. The simplest metric to use here is the Euclidean distance [Sza15]. From the specified metrics, the pipeline generates a connectivity network with optional clustering and exports it separately from the 3D mesh files of the individual components. To make handling these large datasets feasible, the network generation utilizes parallel computing and also lays the foundation for distributed computing.

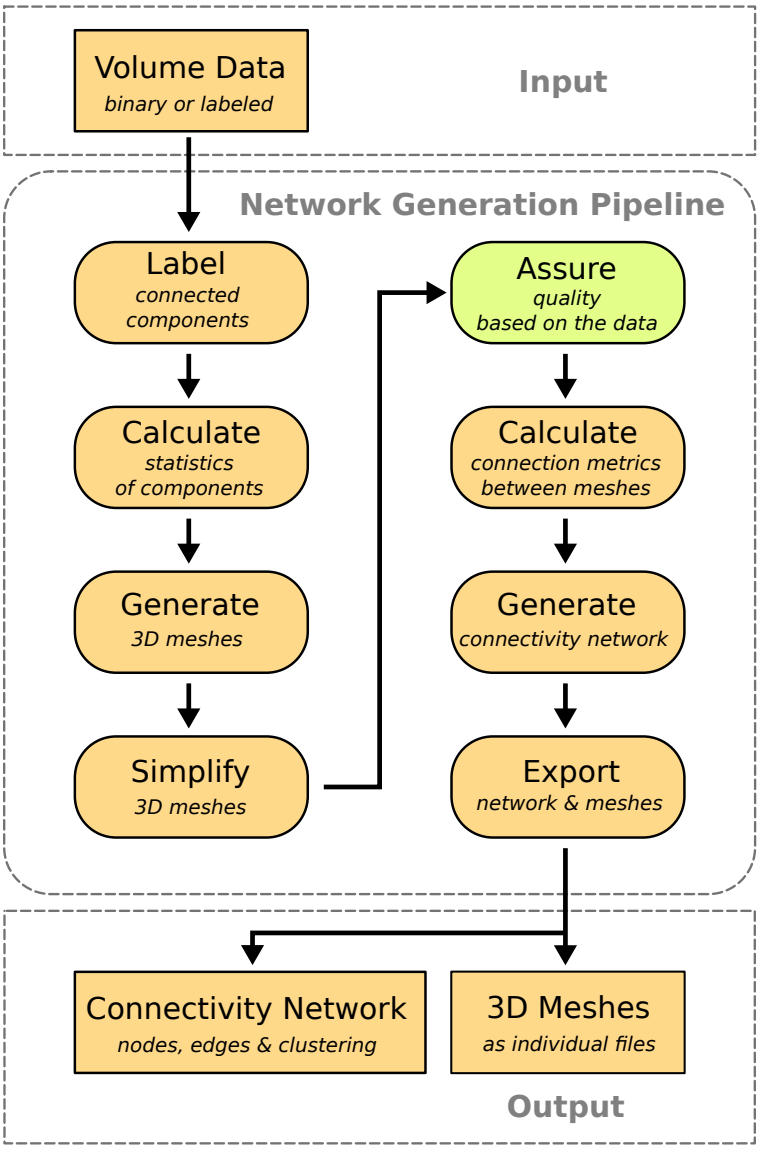

Figure 4.1: A simplified functional overview of the implemented network generation that can be used to create 3D meshes with their associated connectivity network.

The web application is an interactive prototype that visualizes the connectivity between cells and FTUs using the output generated from the network generation pipeline. Figure 4.2 shows the proposed layout of the explorative user interface. The interface consists of three adjacent views. On the left side, there is a 3D view that takes up half of the screen space and shows the 3D meshes. This view offers orbit controls to navigate the 3D environment. On the right side, the screen space is once more split in half, with a network view at the top and an ego graph view at the bottom. The network view visualizes the connectivity network as a hypergraph with expandable clusters. The ego graph is shown when a node has been selected, either in the 3D view or the network view, and creates a radial ego graph showing hop distances between connected cells or FTUs.

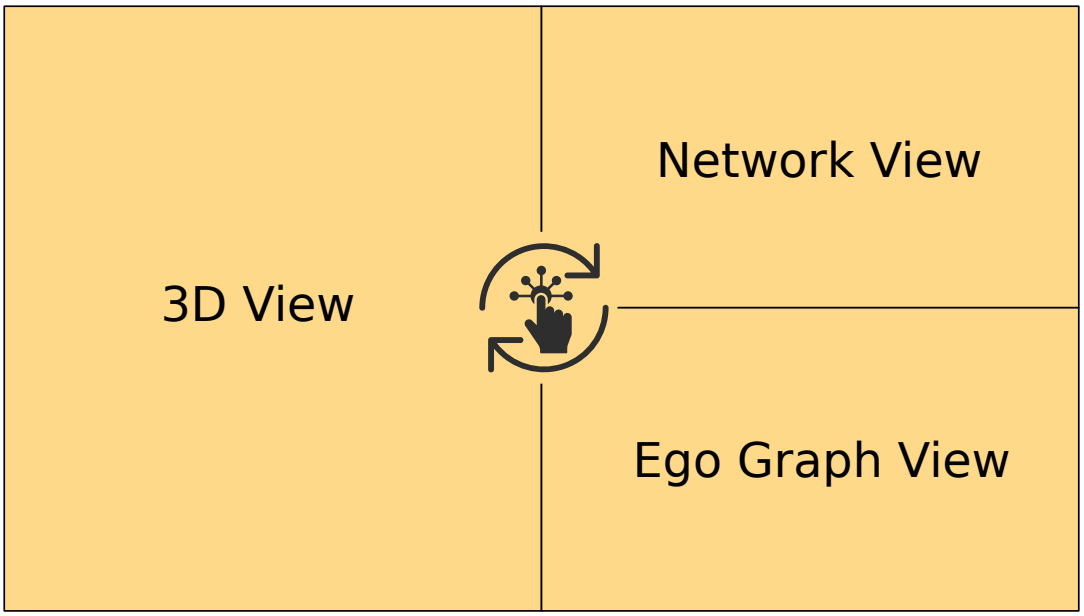

Figure 4.2: The schematic layout of the web application interface. Interaction with any of the three views, 3D View, Network View and Ego Graph View is synchronized between all views.

4.2 Mesh Generation

The aspects of the mesh creation process are detailed in this section while discussing different approaches, performance improvements gained from parallelization and quality assurance depending on the input dataset.

4.2.1 Connected Components

In order to create individual meshes of biologically separate FTUs or cells, the volumetric data has to be split into spatially connected components. This is achieved by labeling

the individual pixels of the volumetric data according to their connectivity based on a set strategy. The step of finding the connected components creates a symbolic representation in the form of labels, which has the same 3D shape as the input data. If the input data is already a symbolic representation in the form of labels from a segmentation process (as opposed to binary data), the step of finding the connected components has to be undergone for each label in the input data.

Rosenfeld et al. [Ros76] describe a basic strategy for finding connected components in a binary image with the size $N \times M$, that is used widely by modern algorithms. When denoting a single pixel in the binary image as $a(x, y)$ with $0 \leq x \leq N-1$ and $0 \leq y \leq M-1$, the connectivity of a pixel $a(x, y)$ can be inspected based on its neighbors. The so-called *4-neighbors* of a pixel $a(x, y)$ are the pixels $a(x-1, y)$, $a(x+1, y)$, $a(x, y-1)$ and $a(x, y+1)$. When adding the pixels $a(x+1, y+1)$, $a(x+1, y-1)$, $a(x-1, y-1)$ and $a(x-1, y+1)$ to the *4-neighbors* they are called the *8-neighbors*. There can exist a *4-connection* or *8-connection* between two pixels v and k if there is a path of pixels $p_1, p_2, ..., p_n$ with $v = p_1$ and $k = p_n$ in which sequential pixels p_i and $p_(i+1)$ with $0 \leq i \leq n-1$ are respectively *4-neighbors* or *8-neighbors*. A maximal set consisting of pixels where each pixel has a *4-connection* or *8-connection* to every other pixel is called a *4-connected* or *8-connected* component.

This principle for 2D binary images can be expanded into 3D volumetric data. For three dimensions there are three distinct connection criteria: 6-neighbors, 18-neighbors and 26-neighbors that correspond to face-, edge- and vertex-connectivity respectively [HQN05]. When considering a voxel v with position (x, y, z), each neighbor $v^{'}$ with position $(x^{'}, y^{'}, z^{'})$ that satisfies Equation (4.1) is connected by the voxel face and a 6-neighbor of v.

$$|x - x^{'}| + |y - y^{'}| + |z - z^{'}| = 1 \tag{4.1}$$

Each neighbor $v^{'}$ that satisfies Equation (4.2) is connected by the voxel face or edge and a 18-neighbor of v.

$$\begin{aligned} 1 \leq |x - x^{'}| + |y - y^{'}| + |z - z^{'}| \leq 2 \\ max(|x - x^{'}|, |y - y^{'}|, |z - z^{'}|) = 1 \end{aligned} \tag{4.2}$$

Each neighbor $v^{'}$ that satisfies Equation (4.3) is connected by the voxel face, edge or corner vertex and a 26-neighbor of v.

$$\begin{aligned} 1 \leq |x - x^{'}| + |y - y^{'}| + |z - z^{'}| \leq 3 \\ max(|x - x^{'}|, |y - y^{'}|, |z - z^{'}|) = 1 \end{aligned} \tag{4.3}$$

Furthermore, He et al. [HRG+17] claim that any connected component labeling algorithm (CCL) can easily be extended not just to three dimensions but also n-dimensions. They broadly divide CCLs into two categories: algorithms based on label-propagation and algorithms based on label-equivalent-resolving. The former algorithms search for unlabeled pixels in the image and propagate the label to pixels that are connected to the original

pixel. These include algorithms that propagate along the contour of a component and algorithms that utilize recursion combined with raster scans. Label-equivalent-resolving algorithms all use multiple raster scans. In the first scan, every pixel is assigned a provisional label, which is then integrated into a unique label through further scans by the detection of equivalent labels. Among the label-equivalent-resolving algorithms, the two-scan algorithms are the most efficient.

Table 4.1: Five state-of-the-art algorithms with the additional memory space they require for an image of size N (adapted from [HRG+17]).

Year	Algorithm	Authors	Memory Space
2004	Contour Tracing Labeling (CTL)	Chang et al. [CCL04]	0
2007	Hybrid Object Labeling (HOL)	Herrero [MH07]	0
2010	Improved Run-based Connected-component Labeling (IRCL)	He et al. [HCS10]	$3 \times N^2/2$
2015	Improved Block based Connected-component Labeling (IBCL)	Chang et al. [CCY15]	$3 \times N^2/4$
2015	Improved Configuration-Transition-based Connected-component Labeling (ICTCL)	Zhao et al. [ZHYC15]	N^2

Table 4.1 shows five state-of-the-art algorithms. CTL and HOL are label-propagation-based algorithms and therefore need no additional memory when computing. IRCL, IBCL and ICTCL are all two-scan equivalent-label-based algorithms that need additional memory space at about the image size N^2. All five algorithms have the same time complexity of $O(N^2)$ but when it comes to execution times ICTCL and IRCL outperform the other algorithms on different image sizes, component density and realistic images. IRCL performs better on low-density images and on medical, natural and textural images, while ICTCL performs better on most other densities as well as artificial patterns and shapes. However, performance differences between the two are very minor [HRG+17].

3D adaptions for both IRCL and ICTCL as well as similar algorithms and already available libraries were considered for the development of the network generation pipeline in this thesis. In addition to finding the connected components, the calculation of various basic statistics proves helpful for further analysis, selective mesh creation and quality assurance steps. Specifically, the voxel counts, bounding boxes and centroids of the components are calculated. However, any number of additional component-specific information, like the morphological or phenotypical features of cells, might be of interest. This leaves room for improving the incorporation of user-specific data and could be considered in future work, as discussed in Chapter 7.

4.2.2 Isosurface Extraction

The pancreas dataset provided by Kiemen et al. [KBG+22] is a labeled pancreas volume that has a resolution of $408 \times 12177 \times 15921$ voxels. This comes to a total of about 80 GB of raw data. Processing or visualizing volumes of this magnitude is either not possible or very slow for most personal computers (PC). In order to cut this down to a manageable size without having to sacrifice connectivity information due to downsampling, the isosurface of the highest resolution volumetric data, which is still computationally feasible, is extracted and then simplified in a subsequent step.

Isosurface extraction is the process of converting volumetric data into a triangle mesh. Marching cubes by Lorensen et al. [LC98] is a very prominent algorithm for isosurface extraction. It is a grid-based method that creates patches of surfaces for every cell in the grid separately. Which surfaces it generates are determined by the signs of the cell's corners, Since a cell has 8 corners there is a total of $2^8 = 256$ possible surface configurations. By utilizing rotation, inversion or reflection, a lookup table of 15 configurations suffices to create all possible surface configurations. If the input data is not binary, the algorithm determines the position of the generated surface patch vertices by linear interpolation of the scalar values that make up the corners of the cell. The marching tetrahedra algorithm proposed by Doi et al. [DK91] solves ambiguity issues that are introduced by some configurations of marching cubes. It does this by slicing the cell into six irregular tetrahedra, creating shared edges with all surrounding cubes and preventing cracks due to slightly varying intersection points. Another approach by Scopigno et al. [Sco94] deals with the ambiguity problem by a small adjustment of the lookup table.

An alternative isosurface extraction method is dual contouring. Ju et al. [JLSW02] created this method by utilizing adaptive octrees. The algorithm creates the surfaces inside the voxels of the volumetric data, as opposed to the marching cubes algorithm which uses the voxels as corners. However, this technique can lead to non-manifold meshes, which necessitates an extra step to fix those geometries. Boissonat et al. [BO05] have shown that Delaunay triangulation can also be used to extract 3D isosurfaces that are made up solely of well-shaped triangles and are topologically and geometrically faithful to the input surface.

Because of the high-resolution imaging data and lack of downsampling, the connectivity of tubular structures like nerves or vasculature was kept mostly intact, which led to huge components that wound through the entire volume. This in turn sparked the development of a parallelized process. Due to the widespread availability of Marching cubes implementations and the excellent potential for parallelization as well as its good performance on grid-like data, an adapted version of it was designed for the network generation pipeline. By running the isosurface extraction in parallel, the central processing unit (CPU) of a PC can be utilized in full, while also keeping the random-access memory (RAM) requirements low. Figure 4.3 shows a simplified visual representation of the developed process. First, the component volume is divided into multiple chunks. The chunk size can be specified by the user to adapt to different datasets and their own RAM

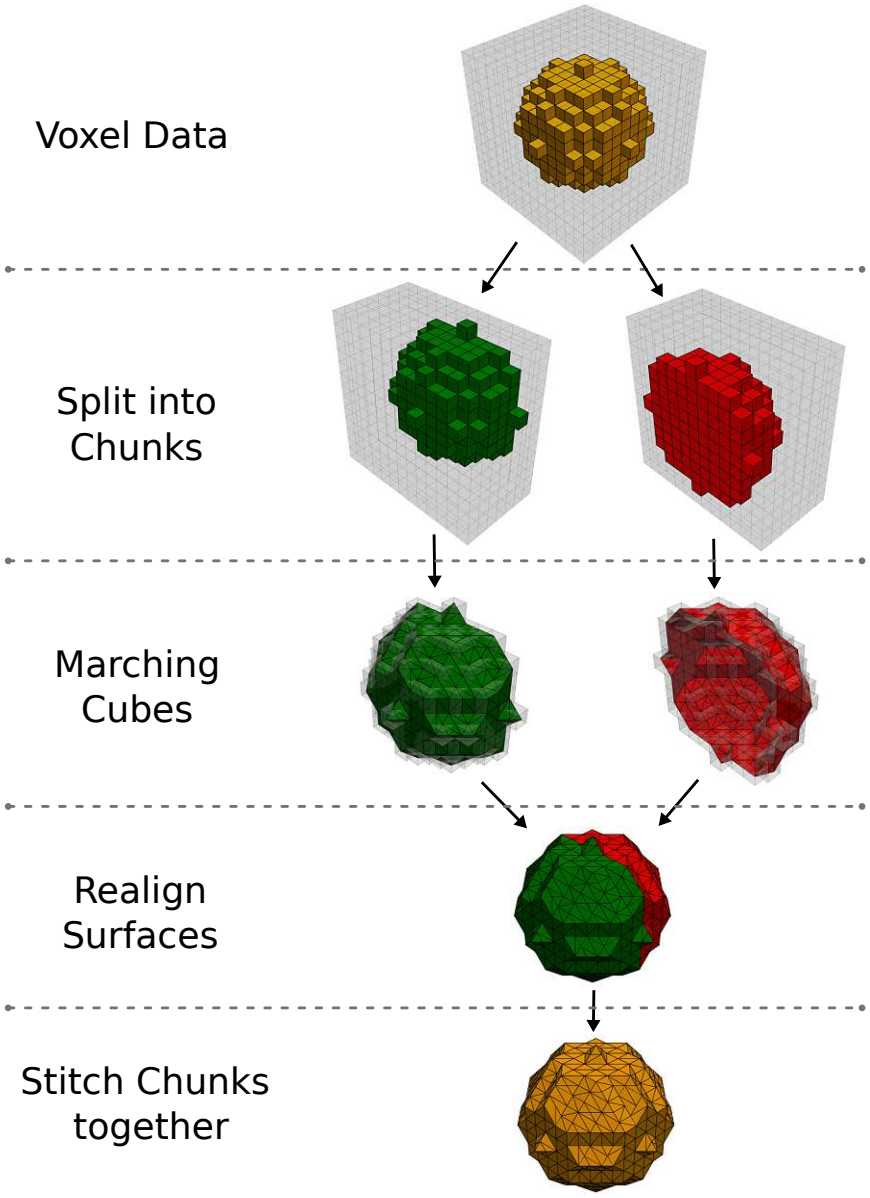

Figure 4.3: The parallelization strategy for the isosurface extraction of large components. The chunk size and consequently the number of chunks is user-specific.

capacities. The isosurface extraction is then performed on several chunks in parallel until all surfaces are generated. Surface meshes of two neighboring chunks are aligned so that the vertices on their bounding boxes overlap. Finally, the meshes are stitched back together by deleting duplicate vertices and updating their normals.

4.2.3 Simplification

By using isosurface extraction on the highest possible resolution of the volumetric data that is computationally feasible, a 3D mesh with a massive amount of vertices is created. Extracting the biggest component of smooth muscle from the downsampled (by a factor of 4) pancreas dataset by Kiemen et al. [KBG+22] already results in a mesh that has more than 1 million vertices. This kind of mesh not only takes up a lot of memory space but slows down any consequential computation and is very hard to visualize in a non-static and interactive tool. Therefore, the triangles in the mesh need to be simplified. Since connectivity is important, this needs to be done in a way that is minimally destructive of the component's shape.

The most general form of mesh simplification tries to reduce the number of vertices and edges. In order to keep the shape, a simplification method needs to adapt its reduction strategy to the local geometrical properties so that it does not remove vertices from regions with high detail or curvature. On the other hand, vertices in regions that are flat should be decimated more strongly. This is achieved by assigning every vertex and/or edge a value that measures the error that their removal would introduce to the shape of the original mesh. After removing a vertex or edge, a remeshing step repairs the triangles in the local region. Mesh simplification is usually done until either a certain number of faces or vertices, a certain percentage of reduction of faces or vertices or an error metric is reached [PB13].

State-of-the-art algorithms for mesh simplification include a method by Garland et al. [GH97] that utilizes quadric error metrics. As an error metric, they use the squared distance to the planes that meet at every vertex. To calculate this error for every vertex, they create a fundamental error quadric K_p for every plane p that meets the vertex. K_p is then summed to a single 4×4 matrix Q that is used to calculate the error at vertex $v = [v_x \ v_y \ v_z \ 1]^T$ in its quadratic form, as shown in Equation (4.4).

$$\Delta(v) = v^T Q v \tag{4.4}$$

After calculating Q matrices for all vertices, they select pairs of (v_1, v_2) that either form an edge or have a Euclidean distance that is less than a threshold parameter t. For each pair, the contraction target $\bar{v}$ with the lowest error $\Delta(\bar{v})$ is computed with the approximation of $\bar{Q} = Q_1 + Q_2$ as the sum of the Q matrices of both vertices. Figure 4.4 visually shows the process of edge contraction and the determination of the contraction target $\bar{v}$. The pairs are then put in a heap by using the lowest error of the contraction target as the key. Finally, the mesh is simplified by iteratively removing and contracting the pairs with the lowest error from the heap. When contracting a pair, the error of the contraction targets of the remaining pairs involving vertices from the current pair, have to be updated. Another approach to mesh simplification is *progressive meshes*, introduced by Hoppe et al. [HOP96]. Viewing *progressive meshes* can work in real-time by adjusting the level of detail (LOD) according to the viewer's perspective. This is achieved by creating a decimation algorithm that employs an edge contraction method (similar to the one in Figure 4.4), but which can also be reversed. The reverse operation

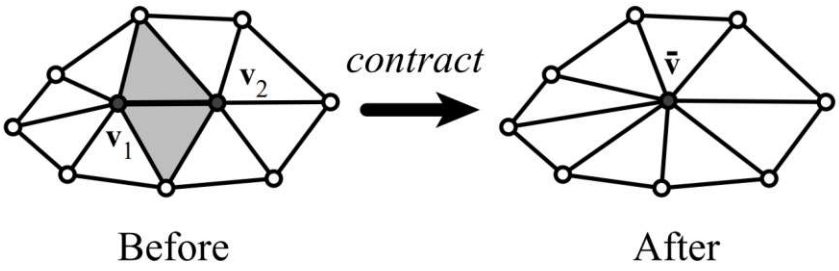

Figure 4.4: The process of edge contraction by merging two vertices and removing duplicate edges [GH97].

is called vertex splitting and allows a smooth transition between a high and low vertex and edge count. The process of creating a progressive mesh leads to a simple base mesh and a series of refinement operations. Even though this is smaller than saving the original mesh, it still takes up a considerable amount of memory.

Lindstrom et al. [LT98] created a memory-efficient polygonal simplification algorithm that is quite similar to the algorithm in [GH97]. They also employ edge contraction and try to place the contraction target in such a way that they minimize the change in volume as well as the boundary of the original mesh.

When comparing different approaches, the method by Garland et al. [GH97] is the fastest but the one by Lindstrom et al. [LT98] has the lowest mean geometric error while *progressive meshes* leaves the most flexibility. Taking this into consideration, an algorithm that utilizes quadric error metrics was employed for initial computation and visualization attempts, because performance was the biggest concern. It is important to note though that an approach like *progressive meshes* that can smoothly transition between LODs is worth considering when creating visualizations of even larger datasets.

4.2.4 Quality Assurance

This section focuses solely on the quality assurance of the meshes that were generated by the network generation pipeline. For quality assurance steps that were taken to improve the connectivity between components, see Section 4.3.2. Since the imaging techniques are limited in resolution or contain mechanical processes (described in Section 2.2) and subsequent segmentation processes still include downsampling of the original data (as described in Section 2.3.1), there is a high chance that generated meshes are merged, warped, rotated or skewed. Therefore, it is of vital importance that the resulting meshes are inspected for every individual dataset and the specific analysis task. As this is a process that is handled on a case-to-case basis and can not be easily generalized, the network generation pipeline is designed to make the generated meshes available for quality assurance.

For the case study of the kidney dataset by McLaughlin et al. [MZS+24] an exemplary

process was designed. On detailed inspection, it became evident that multiple glomeruli that looked like distinct entities to the eye were merged into conglomerate meshes. This might have been a result of the downsampling that was applied before segmentation, or simply due to a lack of resolution in the imaging process. To counteract this, an algorithm was designed that splits the merged meshes into single meshes. For every mesh, the number of glomeruli is estimated by dividing the diagonal of the 3D bounding box by the mean size of the bounding box of all non-merged meshes. This step requires human-in-the-loop interaction to find the specified mean size. Figure 4.5 shows an overview of the splitting process of a conglomerate mesh with three glomeruli. After the number of glomeruli is estimated, it is used to cluster a point cloud consisting of the mesh vertices into distinct clusters. For clustering, the k-Means algorithm by Lloyd [Llo82] is employed. In order to find the cutting planes, the cluster centers are all connected by

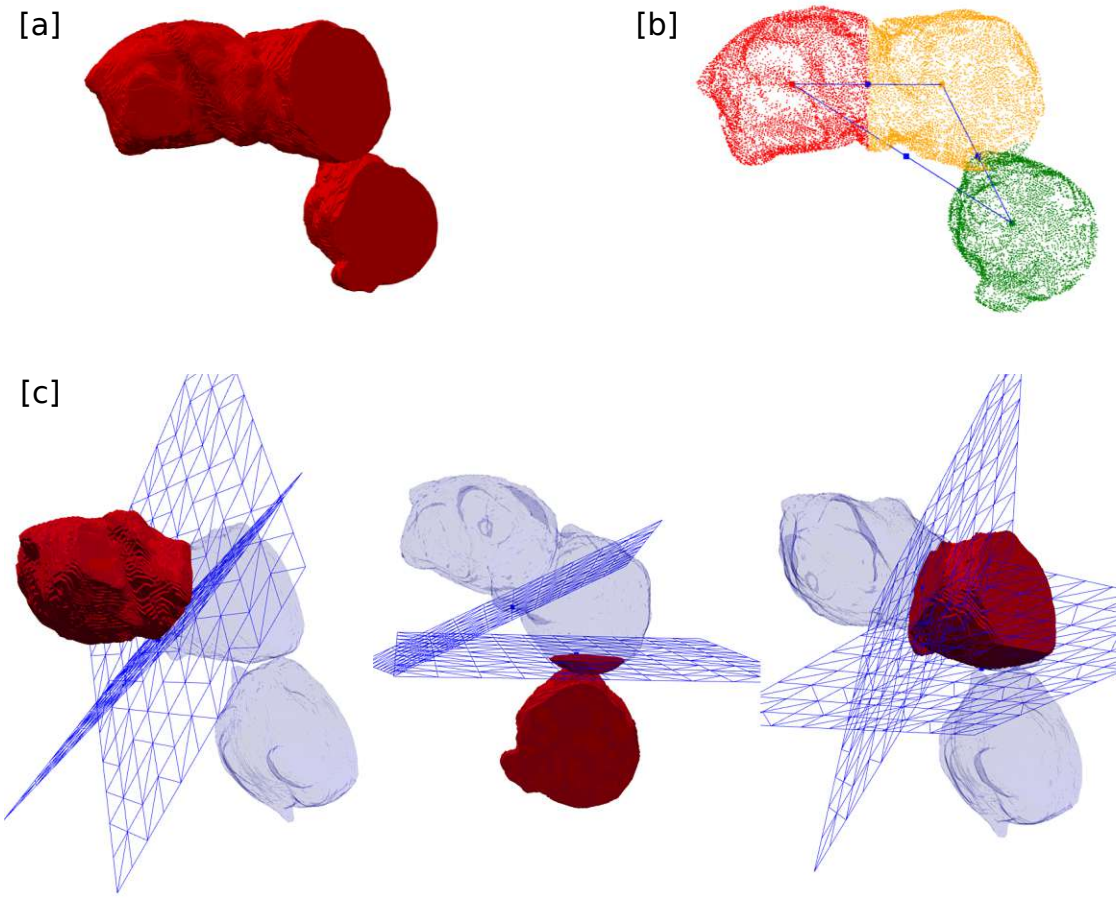

Figure 4.5: The splitting of merged glomeruli meshes that were created from a dataset provided by McLaughlin et al. [MZS+24] in the course of quality assurance. **[a]** A merged conglomerate mesh consisting of three glomeruli, with two on the sample border. **[b]** The clustered vertices (red, yellow, green) with the connected cluster centers. **[c]** The cutting planes (blue) with the resulting glomeruli meshes.

a theoretical line (as seen in Figure 4.5b). This line makes up the normal vector that defines the orientation of the cutting plane in 3D. The position of the plane is selected by finding the vertex between the cluster centers that has the minimal orthogonal distance to the line. The vertex is then projected onto the line to get the actual position of the cutting plane. By using the cutting planes, the mesh is separated into single meshes and the cut is repaired with the help of Attene's [Att10] MeshFix algorithm (see Figure 4.5c). Lastly, the glomeruli that were on the border of the sample and consequently incomplete were removed, since neuronal connectivity that happened outside of the sample could not be faithfully represented.

4.3 Network Generation

This chapter focuses on the connectivity network generation from the meshes resulting from the network generation pipeline described in Section 4.2. Additionally, a quality assurance technique for the connectivity of tubular meshes like vasculature or nerves is proposed.

4.3.1 Metric System

The goal for the connectivity network generation was to devise a system that is as generalized as possible and allows for different connectivity metrics to play a role when considering if two components are connected. Initially, the user is able to specify $\{1, 2, 3, ..., n\}$ input configurations where each points to a set of previously generated meshes corresponding to FTUs or cells. Additionally, the user determines $\{1, 2, 3, ..., m\}$ metric configurations that specify which metric is calculated between the meshes from the input configurations. For all m metric configurations, the system calculates a metric matrix between all possible combinations of input configurations n. This leads to a maximum of k metric matrices that are calculated, as shown in Equation (4.5). Since some metrics might not be relevant for the meshes of all input configurations, the metrics that should be calculated can be specified as well.

$$k = m * \frac{n}{2} * (n - 1) + n \tag{4.5}$$

Since the input for the mesh generation are segmentation masks of biological imaging data, there is no inherent information about which pixels and consequently which FTUs or cells form a biological connection of any kind. To determine connectivity, one therefore either needs additional connectivity data or biological knowledge from which to derive it. In the case of McLaughlin et al. [MZS+24] they used their biological knowledge of the glomeruli to specify a distance of 10μm in space at which they believed a glomerulus to be connected to a nerve. Using this as a starting point, an algorithm that calculates the Euclidean distance between meshes was designed. The algorithm utilizes a multidimensional binary search tree, called a k-d tree, that was proposed by Bentley et al. [Ben75]. It builds a k-d tree of the vertices of one mesh and finds the distance to the nearest neighbor for all vertices of the other mesh by querying the tree. It then

proceeds by finding the shortest distance between two vertices of the meshes, with a total time complexity of $O(n\ log\ n)$. To increase performance even further, the algorithm is designed to run in parallel. However, one cell dataset provided by Yapp et al. [YNZ+24] has 43,000 individual components, which would result in a Euclidean distance matrix of 43.000×43.000 distances. Calculating this on the average PC is not feasible. Therefore, an adapted algorithm that only calculates distances based on the surroundings of a component was devised. The adaption finds a local neighborhood for every component by extending the bounding box by a distance, d and only calculates the distances for the components that are fully or partially in this neighborhood. Figure 4.6 shows the local neighborhood of a single cell from the 43,000 cell dataset. The distance d can be specified through the metric configuration. Depending on the specified neighborhood size, the algorithm can cut down the number of calculations by over 99.9%, but using this method is only reasonable when biological connectivity beyond the neighborhood is impossible or not of interest.

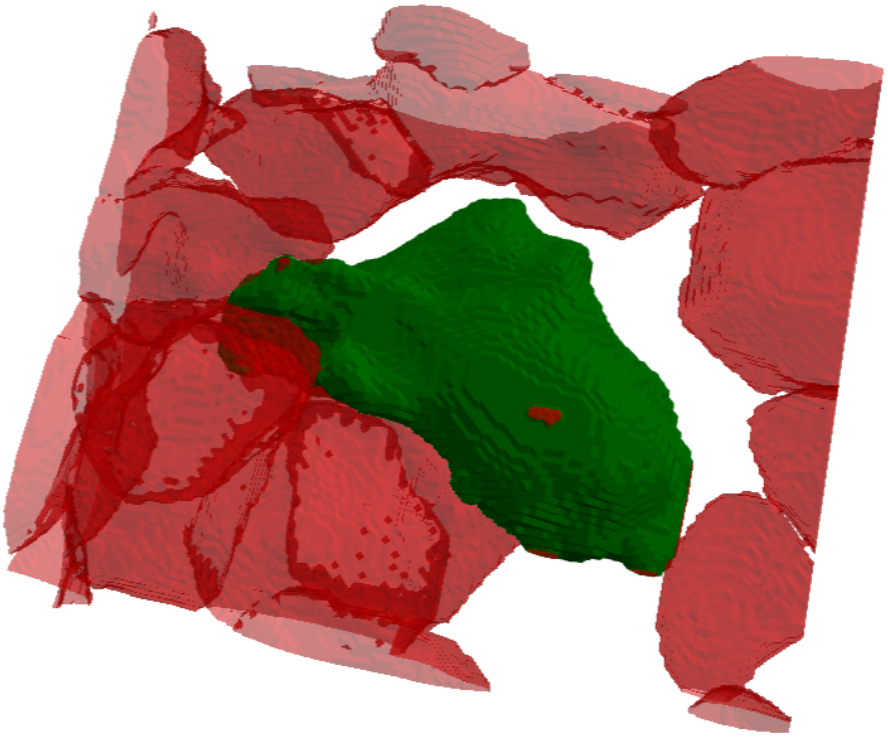

Figure 4.6: The local neighborhood (red) of a single cell (green) from a dataset containing 43,000 cells provided by Yapp et al. [YNZ+24] ($d = 20$ voxels).

To give users the flexibility to choose their own connectivity constraints, the metric system can also be extended with custom metrics by designing and implementing a compatible algorithm.

4.3.2 Tubular Mesh Connectivity Suggestion

While looking into the quality assurance of the meshes generated from the used datasets, a potential disconnect of tubular meshes was discovered. Small patches of the mesh appeared to be missing from nerves that looked continuous to the human eye. This was mainly visible in 3D reconstructions of imaging data that has suboptimal z-continuity due to mechanical capturing processes, like the data provided by Kiemen et al. [KBG+22]. Figure 4.7a shows one such possible disconnect.

Even though some of the missing parts can be explained by the image acquisition and segmentation technique, some can also be explained by biological entities that were either not captured or not visualized. Discerning which case it is requires deeper biological knowledge in the specific field. With this in mind, a system was designed that is able to suggest the connectivity of tubular meshes based on their orientation. This system can be parametrized by the user to fit the use case. It lays the groundwork for a more sophisticated tool that should allow for connectivity quality assurance with a human in the loop. The system uses three metrics to suggest connectivity. First, it uses the Euclidean distance metric d to find components that are in proximity p of each other. For every pair of close components like the one in Figure 4.7a, it calculates two additional metrics in the following way.

The indices of the closest vertices are used to find their position. This is possible since the algorithm described in Section 4.3.1 is able to store the indices of the closest vertices when it calculates the Euclidean distance. For both meshes, a local point cloud is created that consists of the closest vertex v_c to the other mesh and all vertices v_i that satisfy $|v_c - v_i| \leq r$ with r being the local point cloud radius. Figure 4.7b shows the local point clouds of two close components. In order to find the direction of the local point clouds, principal component analysis (PCA) [Pea01] is applied for both of them. The first principal component of the point cloud is the direction in which the vertices have the most variance. This means that it should represent the general direction in which the mesh is pointing at the closest points. Figure 4.7c shows the three principal components for each mesh, that correspond to the eigenvectors of the point cloud's covariance matrix. The eigenvectors are scaled based on their eigenvalues, leading to the first principal component being depicted with a larger arrow. From this, the second metric is derived by calculating the angle ϕ between the eigenvectors with the largest eigenvalue (the first principal components). Additionally, the mean position v_x and v_y of both point clouds is calculated, with x denoting the first and y denoting the second component. From them, the bridging vector $v_b = v_y - v_x$ is defined. The third metric is the angle θ between v_b and the eigenvector with the largest eigenvalue from the first component. Figure 4.8 shows the principle of this calculation in 2D with e_x and e_y as the eigenvectors with the largest eigenvalue for the first and second components x and y.

The user can then specify four variables depending on their data: the proximity p, the local point cloud radius r, the angle between directions a_1 and the angle between the direction and bridging vector a_2. The system then proceeds to suggest connectivity

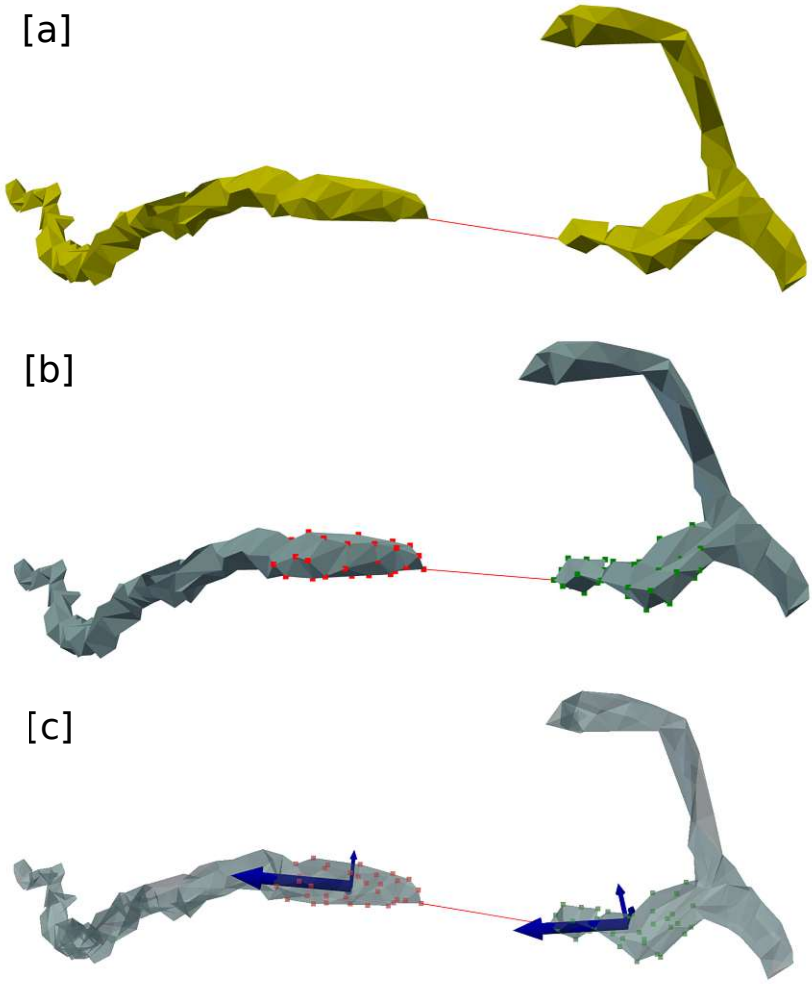

Figure 4.7: **[a]** Two nerve mesh components reconstructed from a 5x kidney dataset provided by McLaughlin et al. [MZS+24]. The red line connects the closest vertices of both meshes and indicates a possible connection that might have been missed during reconstruction. **[b]** The red and green squares indicate the vertices that are in range r of the closest vertices of the respective mesh ($r = 20$ voxels). **[c]** The blue arrows show the eigenvectors of the PCA of the closest vertices for both meshes.

between a pair of components if all three metrics meet the condition in Equation (4.6). The distance d ensures that the components are next to each other. The angle between directions ϕ ensures that the closest sections of the meshes point in the same direction, and the angle between direction and bridging vector θ ensures that the closest sections of the mesh are on the same plane.

$$d \leq p, \quad \phi \leq a_1, \quad \theta \leq a_2 \tag{4.6}$$

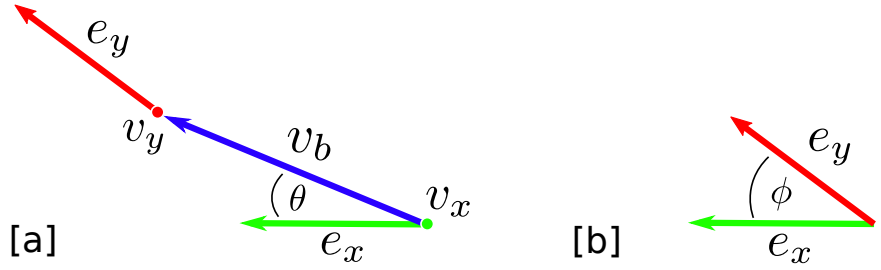

Figure 4.8: The angle θ between e_x and v_b **[a]** and the angle ϕ between e_x and e_y **[b]** make up two of the three metrics used for the incomplete tubular mesh suggestion. The vectors e_x and e_y are the eigenvectors with the largest eigenvalue of localized point clouds at the nearest vertices of two meshes x and y. v_b is the vector between the mean positions v_x and v_y of both localized point clouds.

4.3.3 Graph Creation

By utilizing the metric system described in Section 4.3.1 the connectivity between individual components can be identified based on the Euclidean distance or any other custom implemented metric. The connectivity information of all the combined input configurations has to be represented by a suitable data structure that allows easy transfer of the information while offering additional support for further analysis.

An undirected graph is a triple $G(V, E, \gamma)$, where $V(G)$ is a vertex set, $E(G)$ is an edge set and γ is a relation associating two vertices, called endpoints, with one edge. A loop in this sense is an edge where both endpoints are the same vertex [W+01]. This can be further simplified by defining a simple undirected graph that does not allow any loops, reducing it to an ordered pair $G(V, E)$ with $E \subseteq \{\{x, y\} \mid x, y \in V \text{ and } x \neq y\}$. The simple undirected graph was chosen as the representation for the connectivity since it is one of the most basic structures, with graph theory [W+01] offering a plethora of analysis options. The idea is to give researchers the possibility to calculate metrics like node degree, centrality, robustness and clustering coefficient as well as find structures through abstractions like motifs [NB21, F+02, BS09].

To keep it as general as possible, the network generation algorithm was designed to use the components from the metric system's input configurations as vertices V and any potential connection derived from the connectivity metrics as edges E. If the tubular connectivity suggestion algorithm from Section 4.3.2 is utilized, the network generation merges the tubular meshes that should be connected into combined components, so that they only make up one vertex in the graph. From this, two different connectivity patterns for the resulting graph can emerge, depending on the analysis use case. Figure 4.9a shows intermediary-connectivity, which is used for entities that are connected through another entity. For example, the connection of two glomeruli through a nerve. Figure 4.9b shows metric-connectivity, which is the connectivity of multiple entities with each other due to specific metrics. For example, the connection between individual cells based on membrane to membrane interaction with proximity as the metric.

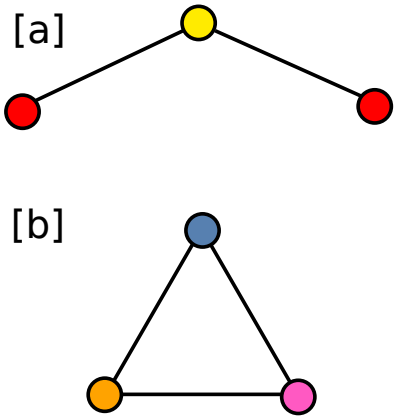

Figure 4.9: **[a]** Intermediary-connectivity of two red entities through a yellow entity. **[b]** Metric-connectivity between three entities.

However, the connectivity pattern is a theoretical concept that has no effect on the actual graph, but is very relevant for the analysis goals of the researchers that provided the data for this thesis. The concept is explained here to highlight that the analysis is still possible even though the graph is kept very basic and generalized.

4.3.4 Scalability

One of the kidney datasets provided by McLaughlin et al. [MZS+24] is at a lower resolution with a zoom of 5× and was used to identify the smaller region of interest that were captured at a zoom of 20×. The quality assurance of this dataset quickly showed that many glomeruli were merged into clusters of more than 20 glomeruli, which made it impossible for the algorithm described in Section 4.2.4 to split them into individual glomeruli. Even though the analysis of such a flawed reconstruction would not be very meaningful, the connectivity network was created to improve the scalability of the visualization, since this dataset holds more than thirty times the number of glomeruli compared to a 20× dataset. In the course of this, a graph clustering algorithm based on interconnectivity between glomeruli was designed.

The algorithm creates an interconnectivity matrix (Figure 4.10) by aggregating the connectivity of every glomeruli pair in the graph. A pair of glomeruli counts as connected if both glomeruli have an edge to the same nerve vertex. In addition to the connection count, the identifier (ID) of the nerves by which the glomeruli are connected are stored in the interconnectivity matrix. After the creation of the interconnectivity matrix, the algorithm iteratively merges the two glomeruli with the highest number of connections into a combined cluster until there are only glomeruli with one connection left. During this process, the combined count of connections is calculated by summing the connections of both components while utilizing the previously stored nerve ID to prevent duplicate connections.

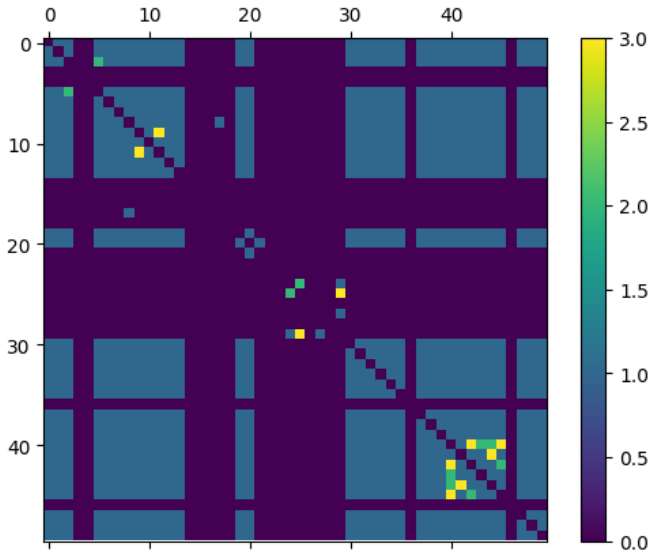

Figure 4.10: The interconnectivity matrix of the first 50 glomeruli in the graph created from the 5× dataset by McLaughlin et al. [MZS+24]. The x- and y-axis depict the glomeruli ID and the color represents the number of nerves that connect them.

In the end, glomeruli that have no connection as well as glomeruli that have only one connection are combined into a separate cluster, respectively. The result of the algorithm is a clustering of the glomeruli nodes and an interconnectivity matrix that complement the original connectivity graph.

4.4 Visualization

In this section, the design of 2D and 3D visualizations that show the connectivity graphs and meshes resulting from the processes described in Section 4.3 and Section 4.2, is discussed. It focuses mainly on the graphs and meshes generated from the kidney datasets provided by McLaughlin et al. [MZS+24], since their research goals had the most alignment with connectivity analysis. During experimental prototyping, three visualizations were designed. The first is a force-directed hypergraph that incorporates the clusters described in Section 4.3.4. The second is a radial ego graph that highlights hop distances. The third is a 3D surface rendering of the meshes. The interactivity of these visualizations will be discussed separately in Section 4.5.

4.4.1 Connectivity Overview

The goal of the first visualization was to create an overview that is either able to show the entire connectivity graph at once or, if this is not feasible, make it traversable through interaction or implement some kind of query interface to get a local overview.

Due to the novel datasets and connectivity graph generation, multiple visualization and representation angles were considered during experimentation.

Under the assumption that glomeruli communicate with each other via nerves, it would make sense to adapt the connectivity graph in such a way that the glomeruli are nodes and the nerves are edges. However, the initial analysis showed that there are single nerves that connect many glomeruli, which leads to a graph that has a large number of edges. Additionally, this would remove any nerve that only connects to one glomerulus from the visualization. Based on the findings by Nobre et al. [NWHL20] that highlight the strength of node-link diagrams on path related tasks which translates to connectivity, the experimentation started there. For the positioning of the nodes, a force-directed placement [FR91] was employed to reduce edge crossings and highlight interconnected clusters. For the initial approach, an attracting force for nodes that have an edge and a repulsive force between all nodes was chosen. Figure 4.11 shows a force-directed node-link representation of a 20× kidney dataset with 28 glomeruli that contains a nerve which is connected to most of the glomeruli.

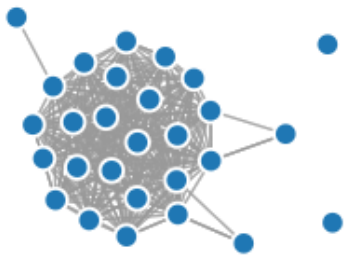

Figure 4.11: A force-directed graph of a 20× kidney dataset with the glomeruli as nodes and nerves as edges.

The visual clutter that is produced by the edges, shows that there is need for some kind of edge bundling [ZXYQ13]. Since most edges would represent the same nerve, simply going back to having the glomeruli and nerves as nodes yields a much more comprehensible visualization, as can be seen in Figure 4.12. This kind of visualization also preserves the nerves that only connect to one glomerulus. The nerve that is connected to the majority of glomeruli also becomes very evident when looking at this visualization. McLaughlin et al. [MZS+24] were mainly interested in the interconnectivity between glomeruli and therefore performed a step they coined *trunk removal*, in which they removed big parts of this main nerve, which will be referred to here as *trunk*. The aim of this thesis, however, is to keep the connectivity abstraction and visualization as close as possible to the original data.

In order to make the distinction between nodes easier, they were colored in a way that is common in anatomical illustrations. Therefore, the nodes depicting glomeruli, which

are a network of capillaries, were colored red while the nerves were colored yellow (see Figure 4.13). Since color is a very complicated topic that hinges on a wide variety of

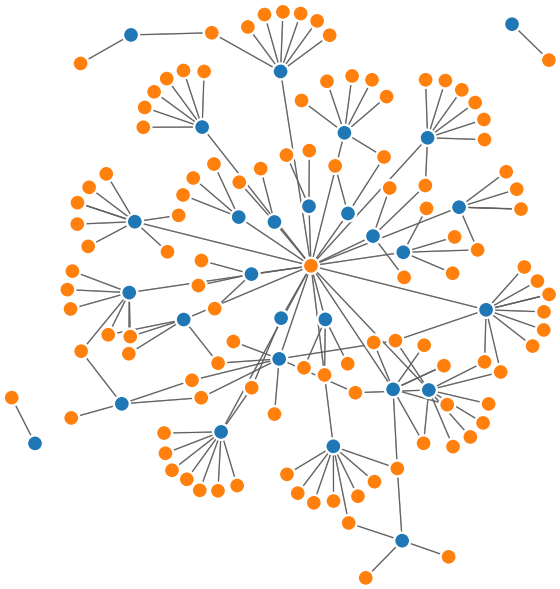

Figure 4.12: A force-directed graph of a 20× kidney dataset with glomeruli (blue) and nerves (orange) as nodes. The edges represent a proximity of less than 50 voxels.

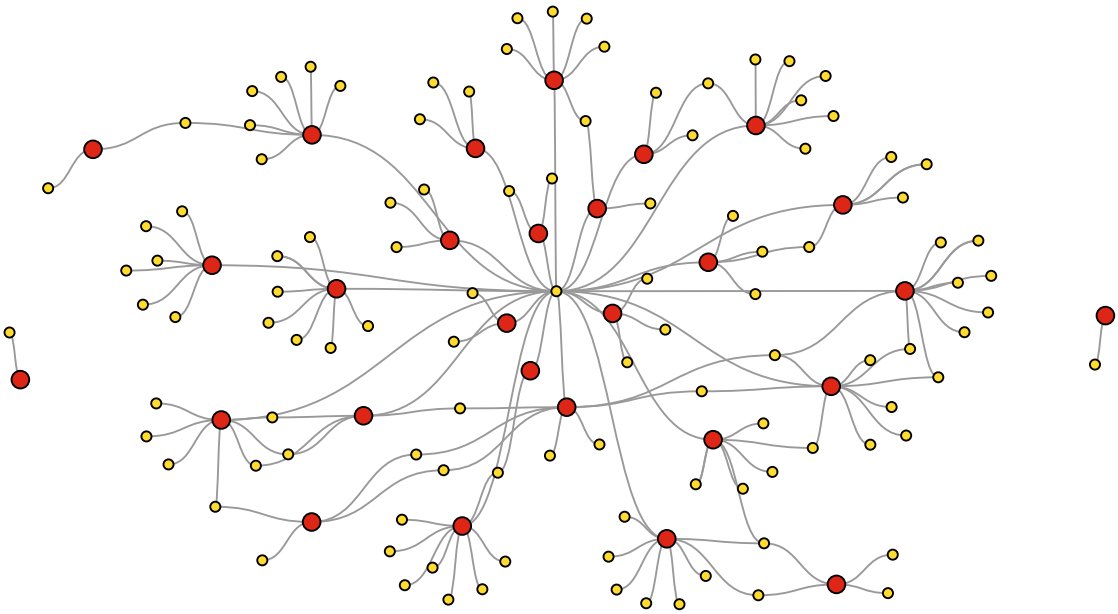

Figure 4.13: A force-directed graph of a 20× kidney dataset with glomeruli (red) and nerves (yellow) as nodes. The connecting nodes (nerve nodes) are smaller, the edges are Bézier curves, and the whole layout is expanded to fit the viewport.

different factors, including the dataset that is represented, the decision was made to make it configurable. For the kidney dataset, an exemplary scheme was chosen to show the impact that color has on comprehension and aesthetics. Figure 4.13 also shows an attempt at improving the path finding between glomeruli by changing the edges from straight lines to Bézier curves [Bez74], making them easier to follow in case of edge crossings. Furthermore, the utilized area of the viewport is maximized by introducing a boundary force and forces in the X and Y direction that stretch the graph depending on the width and height of the viewport. This makes it easier to see connections that were previously bunched up and cluttered. Section 4.3.3 introduced the concept of intermediary-connectivity, which is applicable for the kidney dataset, since it is assumed that glomeruli are communicating with each other via nerves. To highlight this concept, while also keeping the nerve nodes, their size is decreased in respect to the glomeruli nodes.

Pathari et al. [PNAM24] found that light-on-dark color schemes, commonly known as *Dark Mode*, have a positive effect on eye fatigue in bright ambient conditions. Figure 4.14 shows a graph with a dark background, which in addition to the eye fatigue aspect improves the contrast of the visualization due to the chosen color scheme for the nodes and edges. To make it clear which entity is the intermediary connecting entity, the edges are colored in the same color as the connecting entity. This way, the graph looks like glomeruli that are connected by nerves. Finally, a legend is added in the lower left corner to improve node comprehension and also offer a tool to switch which entity is the intermediary connecting entity. For more information on interactive behavior, see Section 4.5.

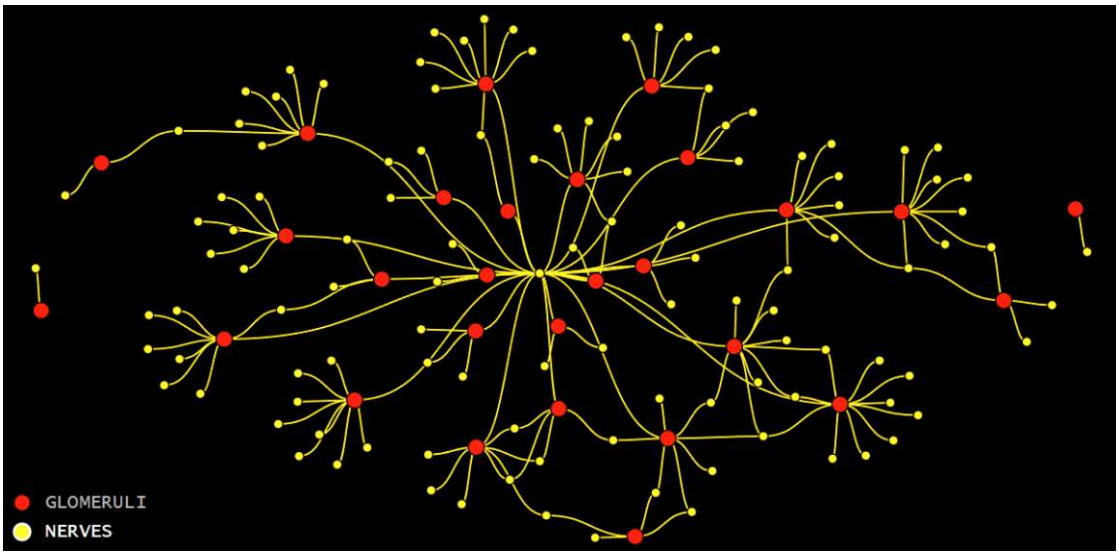

Figure 4.14: The graph shown in Figure 4.13, but with a dark background, a node legend, and edges that are colored the same as the intermediary connecting nodes to highlight glomeruli connectivity.

While the visualization seen in Figure 4.14 works well for smaller datasets, it is not very scalable and quickly becomes very cluttered with an increase in glomeruli and nerve entities. Figure 4.15 shows an extreme example of a 5× dataset with over 750 glomeruli.

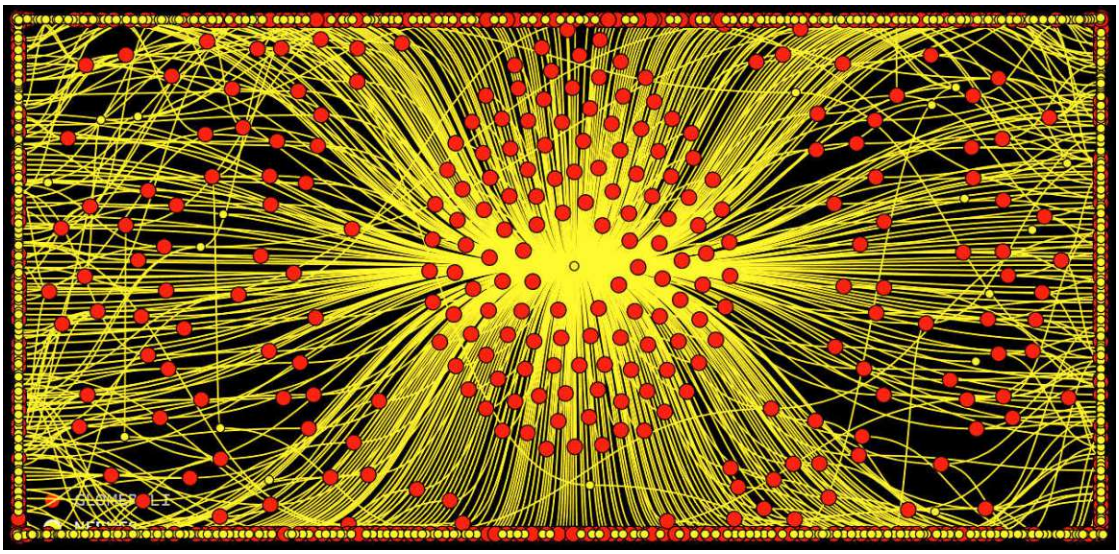

Figure 4.15: A visualization of the graph generated from the $5x$ kidney dataset, with $1,518$ nodes and $1,854$ edges that highlights the scalability problems.

Evidently, some sort of agglomeration is necessary to make out any kind of connectivity between the shown entities. As described in Section 4.3.4, the data of the 5× dataset, still needs either higher resolution or better quality assurance for any meaningful analysis. Even so, the current visualization was already pushed to its limits by some of the smaller 5× datasets and therefore a more scalable visualization was designed. For this, the clustering described in Section 4.3.4 is utilized to create the graph in Figure 4.16. Clusters of glomeruli and all the connections between the glomeruli in a cluster are represented by a red rectangle node. Glomeruli and nerves that are not in a cluster are visualized with the same nodes and edges just as before. If a nerve forms a connection between a glomerulus and any glomeruli of a cluster, it is also visualized. The same goes for nerves that connect different clusters, with the prime example for this being the big nerve that connects most of the glomeruli (see Figure 4.14). Since the clustering is based on glomeruli, the legend loses interactivity, as changing the intermediary connecting entity would call for a recalculation of the clusters, which is too slow to be done in real time by the designed algorithm. In order to make this clustering graph useable for analysis, it needs some kind of user input to make navigating the graph possible. The interactive mechanisms that allow this are described later in Section 4.5.

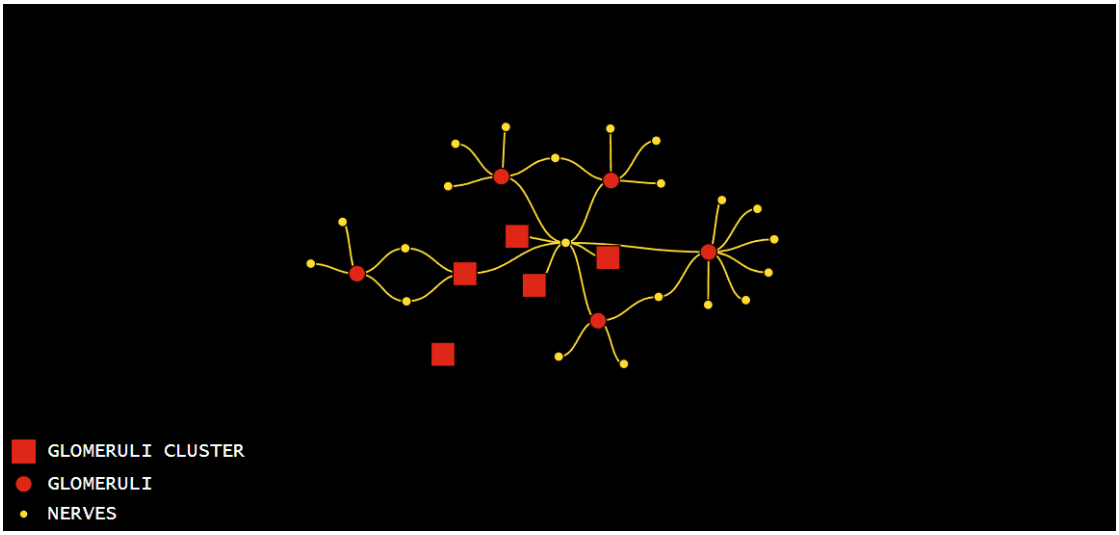

Figure 4.16: The graph shown in Figure 4.14 with visual clustering based on the glomeruli interconnectivity. Glomeruli clusters are depicted as red rectangles.

4.4.2 Detail View

The aim of the second visualization was to give a more detailed view on the connectivity of a single component in the connectivity graph. The user should be able to select the component of interest from either the overview visualization, described in Section 4.4.1, or the 3D visualization, described in Section 4.4.3. The view should make it easier to analyze local connections by further abstracting the overview through the removal of less relevant components.

One visualization technique that fits these parameters well, is the node-relative ego(centric) graph, which is often used for social networks but has also been used for visualizing neuronal connectivity [AABS+14]. An ego graph is a subgraph of a larger network that depicts nodes that are relative to a focal node, called the *ego* node. The neighbors of the *ego* node are named *alters*. Depending on the hop distance to the ego node, the neighbors are called 1-alters, 2-alters, up to k-alters [EPF+24]. Figure 4.17 shows the construction of a 2-alter ego graph by creating a layered node-link diagram (Figure 4.17b) from a regular node-link diagram (Figure 4.17a), in which the layers correspond to the hop distances. Nodes with a hop distance larger than 2 are emitted, due to it being a 2-alter ego graph. Furthermore, the graph does not show the edges between nodes that are on the same layer. Utilizing this approach, the visual complexity of a graph can be reduced interactively while also improving the connectivity analysis of individual components.

For the detail view, a custom radial ego graph was designed. The radial design was chosen to increase the maximum number of nodes that one layer can contain, as the kidney datasets contained glomeruli with many direct neighbors. This was mainly due to the *trunk* nerve described in Section 4.4.1. Figure 4.18 shows the initial attempt at creating

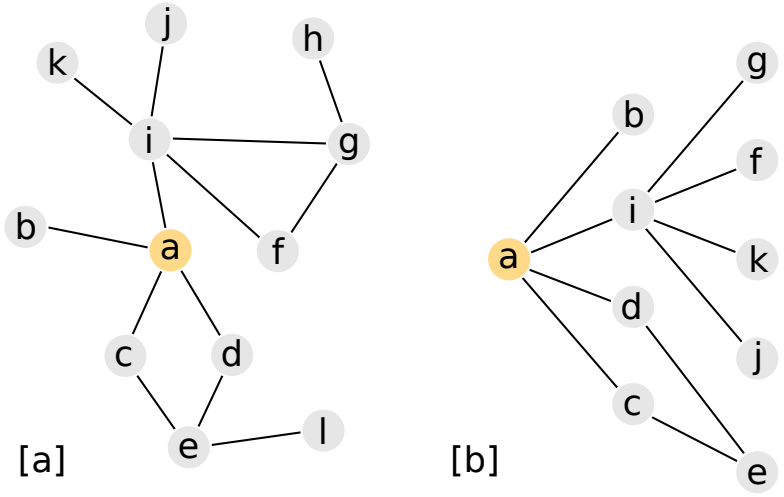

Figure 4.17: A regular node link diagram [a] and the corresponding layered 2-alter ego graph [b] for the orange node *a*.

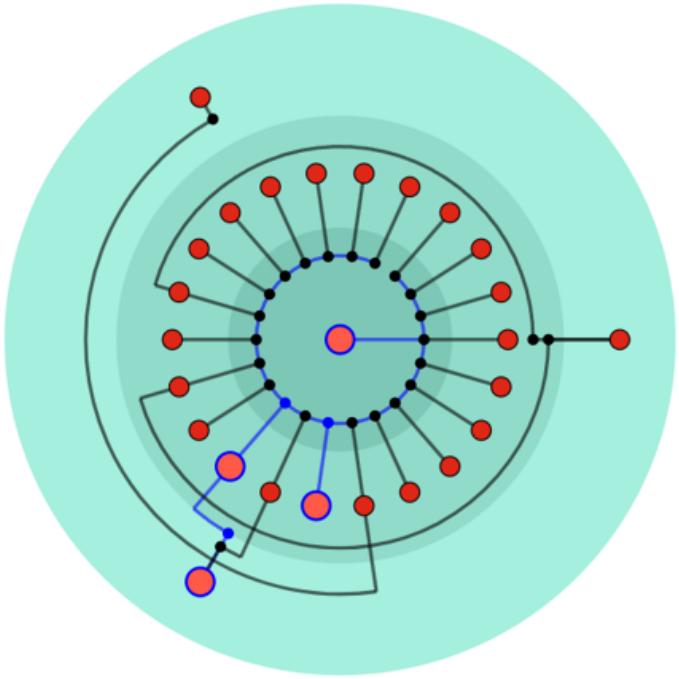

Figure 4.18: The first attempt at creating a radial ego graph as an additional detail view. Depicted are glomeruli (red) from a $20x$ kidney dataset, with the edges representing nerve bundles. The blue highlight shows a multi-selection of connected components and the cyan color gradient represents the layers of the graph.

this visualization with a $20x$ kidney dataset. The ego node is at the center, depicting a selected glomeruli node of the connectivity graph, and the alters are positioned in circular layers that are highlighted by a color gradient with increasing luminosity the farther the hop distance from the ego node gets. Glomeruli can be connected with each other via multiple nerves. Therefore, the edges represent either a single nerve or a bundling of all the nerves that connect the specific glomeruli. The nodes are positioned at a fixed angle depending on the number of nodes in the layer, but this could be further improved with an algorithm that minimizes edge length. However, creating such an algorithm is not trivial, due to the multi-layered interconnectivity of the nodes. The whole visualization is based on utilizing interactivity to select the ego node. This process as well as the multi-selection seen in Figure 4.18 is further described in Section 4.5. Since the ego graph is different for every node, it needs to be recreated in real time or precomputed.

Figure 4.19 shows an improved version of the radial ego graph. The light on dark color

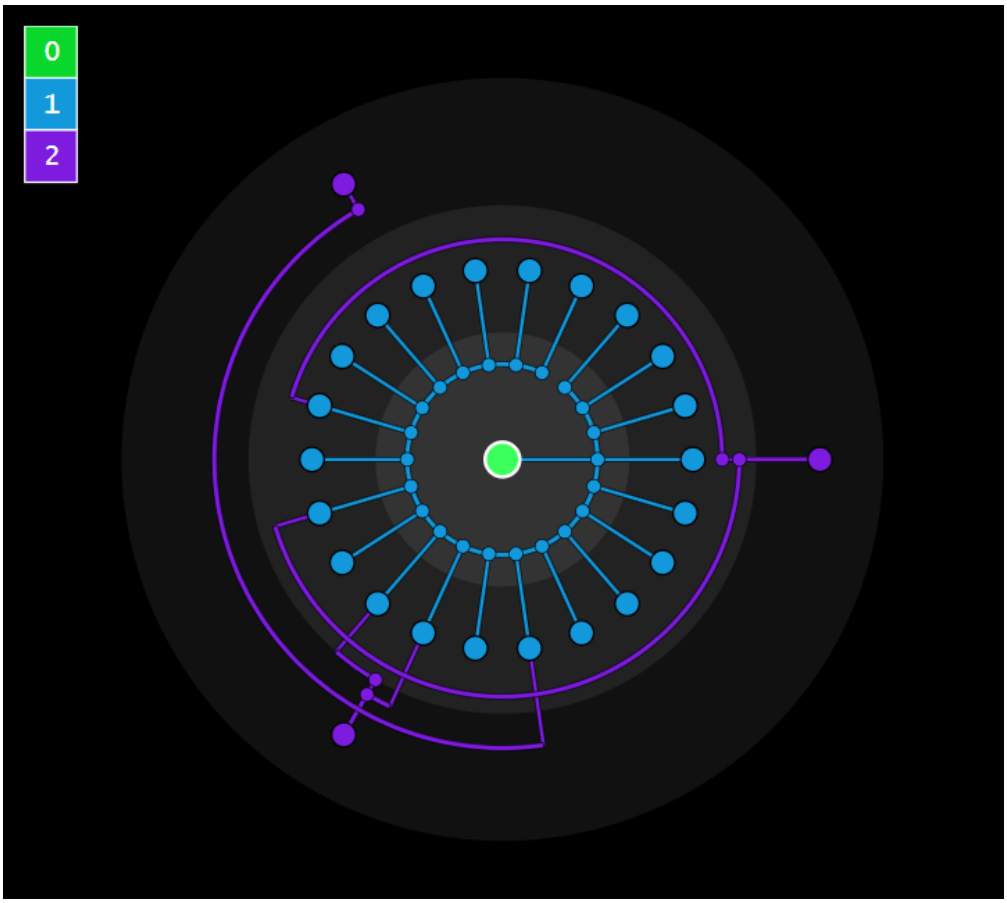

Figure 4.19: The same graph as in Figure 4.18, with a light on dark color scheme, including a color gradient for the ego node and alters. Each layer of nodes and its connecting edges have the same color, which represent the hop distance to the ego.

scheme used in Section 4.4.1 is applied here as well. Additionally, a custom color gradient is created depending on the number of visualized layers. Every layer of the ego graph is associated with one color from this gradient. The nodes of each layer and the edges connecting them to the layer below are colored according to this gradient. The gradient association is displayed to the user through a legend that contains the color and number of hop distances to the ego node. By coloring not only the background but also the nodes and edges, the distinction between components with different hop distances becomes a lot clearer, especially when this color gradient is synchronously propagated to the other visualizations, which will be shown Section 4.5.

4.4.3 3D View

The purpose of the third visualization was to show the actual volumetric data in 3D to allow for better spatial analysis of connectivity. Especially, when it comes to tortuous structures like nerves or vasculature, the added dimension helps in understanding their topology without the need for additional views.

Since the network generation pipeline creates simplified meshes, the 3D visualization was created by using surface rendering. Figure 4.20 shows a surface rendering of the

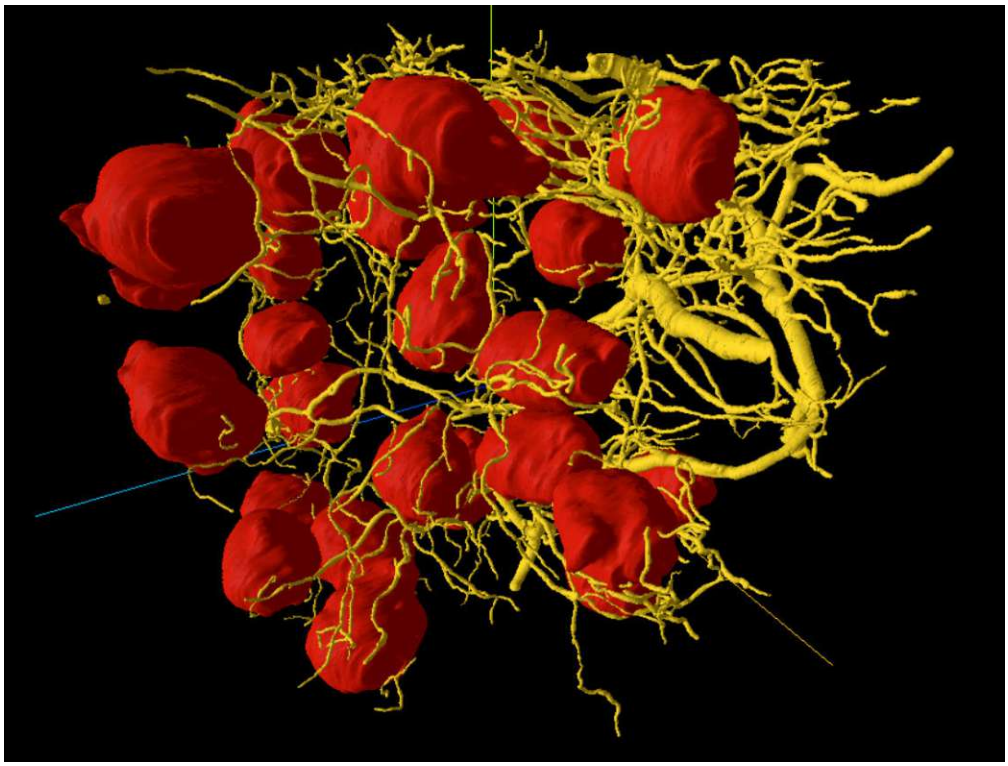

Figure 4.20: A surface rendering of glomerulus (red) and nerve (yellow) meshes generated from a $20x$ kidney dataset provided by McLaughlin et al. [MZS+24]

glomerulus and nerve meshes that were generated from a $20x$ kidney dataset. The color scheme was chosen to match the light on dark scheme discussed in Section 4.4.1 with the meshes being the same color as the nodes in the graph. Looking at this visualization, it becomes apparent that the meshes are rather densely packed, making it hard to distinguish meshes or even trace a nerve. In order to improve this, multiple interactive mechanisms, that integrate the other views into the analysis process, are designed and will be discussed in Section 4.5.

Another issue in 3D visualization is scalability. The 3D reconstruction of one dataset provided by Yapp et al. [YNZ+24] results in 43,000 meshes. Loading and rendering this many meshes at an useable vertex resolution is not feasible for most PCs. Therefore, a pipeline that generates meshes at various LODs by Ackerman [Ack] was investigated. This pipeline generates multi-resolution meshes in the *precomputed* [MSb] format, which is specific to the web viewer *neuroglancer* [MSa] that is based on the Web Graphics Library (WebGL) [Gro]. This web viewer is able to load specific meshes in the appropriate LOD, depending on the zoom and the area the user is looking at. Figure 4.21 shows the dataset with 43,000 cells loaded in *neuroglancer* at four different zoom levels. The meshes were generated as described in Section 4.2 after downsampling the volumetric data by a factor of one and without any mesh simplification. They were then transformed into the *precomputed* format with the aforementioned pipeline. The LOD increases with the zoom, making it possible to show the whole dataset in an interactive 3D view. However, even though the total disk size of the meshes can vastly be reduced by utilizing the *precomputed* binary format, *neuroglancer* still requires a lot of RAM when higher detailed meshes in multiple regions are investigated. While the rendering was fast enough to be considered interactive, it always delivered less than 5 frames per second on a machine with an AMD Ryzen 7 2700X with 16GB of RAM and a GTX 1070Ti as its dedicated graphics processing unit (GPU). This is still far removed from delivering a smooth analysis experience on datasets of this scale, but the concept is very promising.

Improving and integrating this kind of LOD visualization would go beyond the scope of this thesis, which is mainly focused on connectivity. It was therefore decided to keep the 3D view simple and fast by sacrificing scalability in order to highlight the concept of the connectivity analysis. Nevertheless, utilizing *neuroglancer* with its *precomputed* format is definitely worth considering for practical applications and future work, as demonstrated with the dataset provided by Yapp et al. [YNZ+24].

4.4.4 Positioning

The designed views are positioned according to the layout seen in Figure 4.2 by dividing the screen space in half and placing the 3D view on the left side, with the views that incorporate the abstraction on the right side. The right half is divided into top and bottom, with the connectivity overview designed in Section 4.4.1 at the top and the detail view designed in Section 4.4.2 at the bottom. The views are juxtaposed in a way that optimizes the used screen space to deliver as much information as possible in one window.

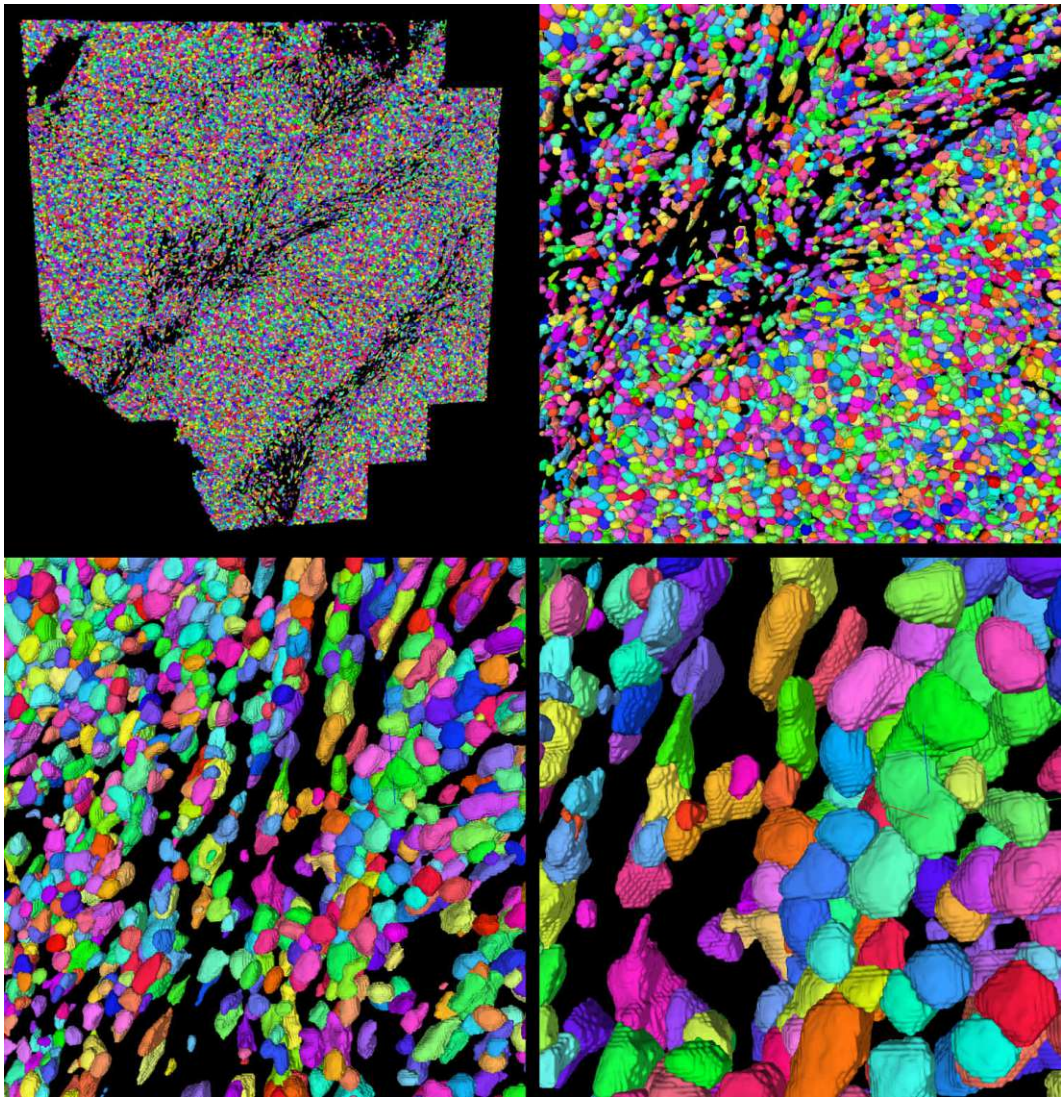

Figure 4.21: A 3D reconstruction of a dataset containing 43,000 cells provided by Yapp et al. [YNZ+24], visualized in *neuroglancer* [MSa] by utilizing the *precomputed* [MSb] format. The panels show how the LOD dynamically increases with the zoom, allowing an interactive visualization of this magnitude.

4.5 Interactive Mechanisms

This section describes the development of interactive mechanisms that were created to improve and connect the three views discussed in Section 4.4 in a way that provides a conceptual analysis workflow for the connectivity network created from the processes designed in Section 4.3. Since all three views are different representations of the same data, the interaction with one view is synchronized in parallel to the other views when it can be projected in a meaningful way. As in Section 4.4 this section mainly focuses on the graphs and meshes generated from the kidney datasets [MZS+24].

The 3D view described in Section 4.4.3 is augmented with orbital controls that utilize the mouse of the user. This facilitates the 3D navigation of the meshes by allowing the user to change the camera target position, the rotation around the target and the zoom. In order to reduce the learning curve, the orbit controls should use a common scheme that can be found in other 3D viewers of widespread applications.

Since the connectivity overview (see Section 4.4.1) shows clusters based on interconnectivity, the interconnectivity metric as well as the components belonging to said cluster should be displayed to the user on demand. This is done by introducing a tooltip as well as highlighting the contents in the 3D view. Figure 4.22 shows that on hovering the mouse over a cluster node in the connectivity overview, a tooltip appears that displays the cluster ID as well as the number of glomeruli nodes and the highest interconnectivity between two glomeruli in the cluster. The interaction is synchronized with the 3D view by giving all the meshes that do not belong to the hovered cluster a gray texture and making them transparent.

The main purpose of the connectivity overview is to give the user the ability to explore the whole network. To make this possible with the applied clustering, the user can expand the clusters by clicking on them. Figure 4.23 shows the expansion of a cluster by creating the nodes and edges that are contained in it, linking the new nodes to existing ones and restarting the simulation of the force directed layout. For larger datasets, it would be important to make the clusters collapsible again as to not overload the user with information or create a cluttered view like the one shown in Figure 4.15. This could be achieved by creating a dynamically growing outline around an expanded cluster, with the contents as a separate force-directed layout. This could further be improved by limiting the node count in the visualization and automatically collapsing some to clusters depending on the number of expanded nodes. Since the implementation of this concept requires a custom tailored solution that needs to be developed around an existing framework, it is not in the scope of this thesis.

The ego graph that makes up the detail view described in Section 4.4.2 focuses on a single node and its connection by constructing a subgraph from the whole network. The user needs to be able to specify which node they want to focus. Therefore, simply clicking the node of interest in either the connectivity overview or the 3D view results in the creation of an ego graph with the clicked node as the ego. Figure 4.24 shows how the views adapt after a node has been selected. The constructed ego graph is shown in the bottom right

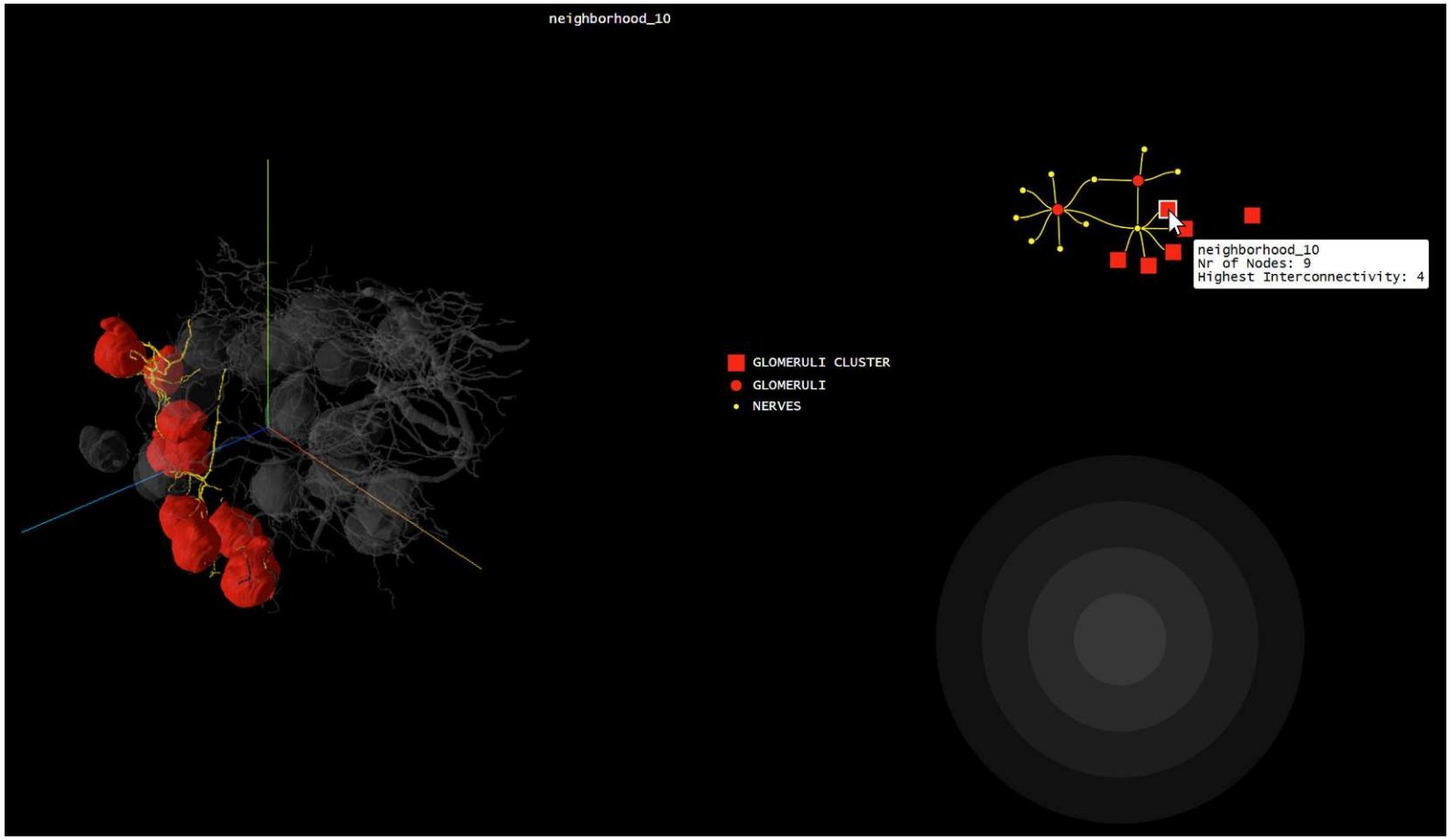

Figure 4.22: When hovering over a rectangular glomeruli cluster node, a tooltip containing the cluster ID as well as the number of glomeruli nodes and the highest interconnectivity between two glomeruli in the cluster appears. The corresponding meshes are highlighted in the 3D view by graying out unrelated meshes and making them transparent.

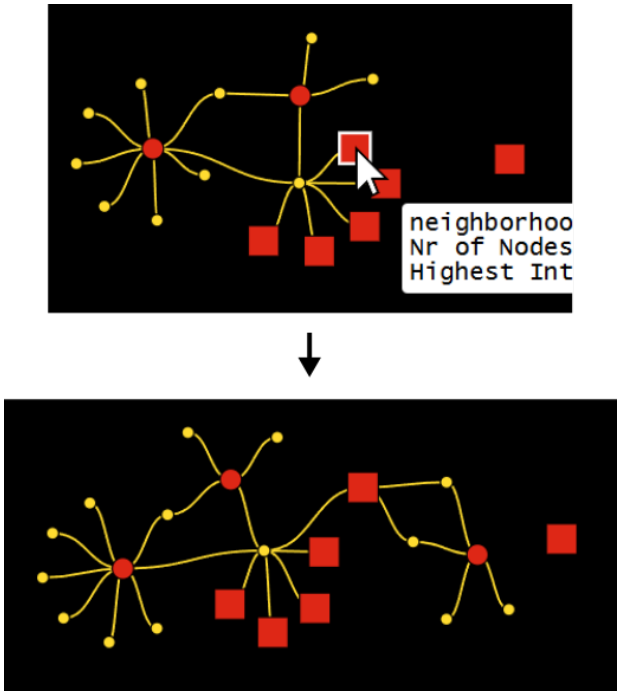

Figure 4.23: Clicking a cluster node dynamically expands the graph with its contents.

corner, and the color scheme of the different layers is propagated to the other views. In the connectivity overview, possible cluster nodes are colored depending on the containing node that has the lowest hop distance to the selected node. If the selected node is in a cluster that has not been expanded, the whole cluster is colored appropriate to the color of layer zero. Nodes, edges as well as meshes that are not part of the ego graph are grayed out, and in the case of meshes made transparent, to indicate that they are not connected. The legend of the connectivity overview is reduced to shapes, while the legend for the layer colors of the ego graph is shown. Even though Figure 4.24 shows the selection of a glomeruli node, it is not limited to glomeruli and the whole visualization is capable to show the selection of any node irrespective of type.

With three different representations of the same data, it becomes important to be able to associate regions of interest in one visualization with the corresponding parts of the other visualizations. This is done by showing the ID of a hovered entity in either a tooltip or at the top of the view for the graph views and 3D view respectively, while also highlighting the corresponding parts in all views by adding a glowing white border. The white border is easily recognizable on the dark background due to its maximal contrast, as can be seen in Figure 4.25. When a node is hovered that belongs to a cluster that has not been expanded, the entire cluster is highlighted.

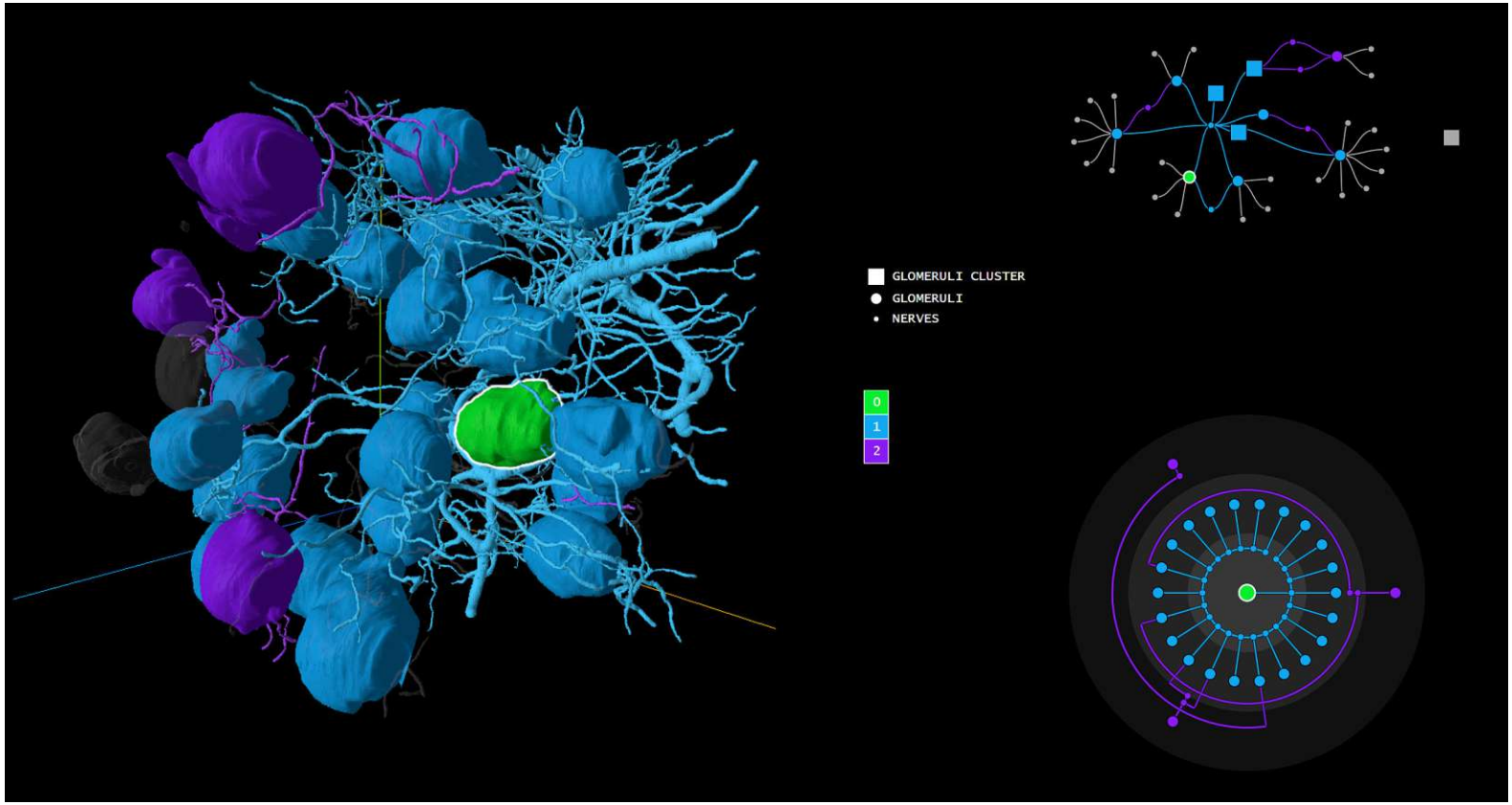

Figure 4.24: The selection of a node or mesh in either the connectivity overview or the 3D view leads to the generation and visualization of an ego graph (bottom right), while adapting the color scheme to be consistent throughout the three views.

Figure 4.25: Hovering a node in any view highlights the corresponding parts in all views by showing a glowing white border and a tooltip with the components ID.

CHAPTER 5

Implementation

This chapter discusses the implementation of the design choices described in Chapter 4. It presents the used technologies and languages while explaining possible advantages and disadvantages. The goal is to select frameworks and languages that are supported on a wide variety of operating systems and are easy to use for scientists with less experience in the field of computer science.

5.1 Implementation Overview

The application is split into two independent parts. Firstly, the data processing part, which is mainly concerned with the network generation pipeline that handles the mesh generation (see Section 4.2) and network generation (see Section 4.3). Secondly, the visualization part (see Section 4.4) allows the analysis of the data produced by the network generation pipeline.

Due to their independence, different frameworks are used for the implementation of both parts in order to utilize the individual strengths of the frameworks for the respective areas. Figure 5.1 shows a general overview of the implementation and interplay of its parts. The data processing part is implemented as a python package called *Connectivity Network Generator (CNG)* that incorporates a simple and installable command-line interface (CLI) as well as python notebooks that allow more fine-grained control over the pipeline. Python was chosen since it is widely used in the scientific community, as it is easy to use, offers a lot of science related tooling support and can be employed on Linux, MacOs and Windows. The CNG outputs a directory that includes both the network as a JavaScript Object Notation (JSON) file and the mesh files that correspond to the nodes of this network. This directory is then served through a basic Hypertext Transfer Protocol (HTTP) server implementation. The visualization is implemented as a web application with the open-source JavaScript library *React* [Mc] and the open source JavaScript library *d3.js* [BDH+] as its main components. A web application was chosen

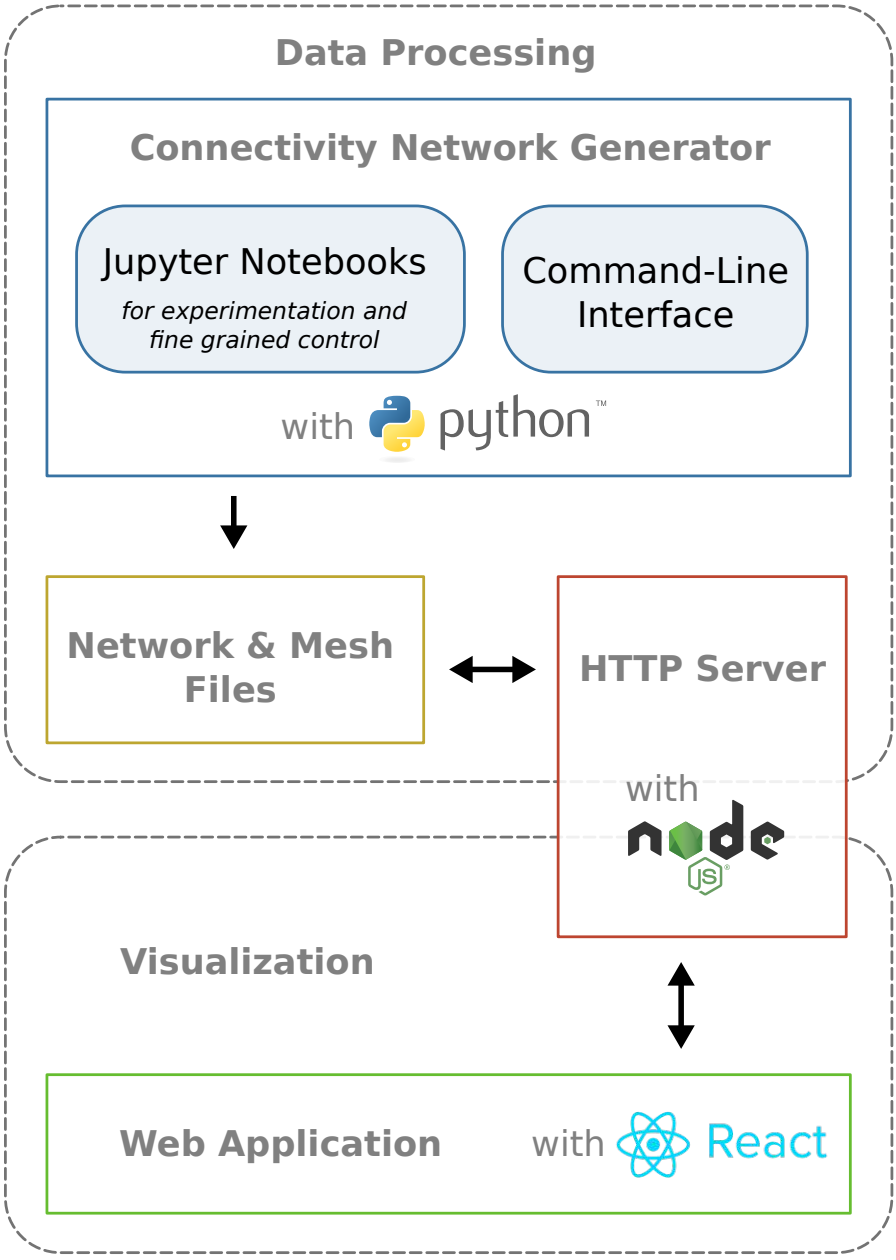

Figure 5.1: A general overview of the implementation and the relationships between the different parts.

to support multiple operating systems without much overhead. Since the research group behind this thesis, developed the React based Vitessce [KGM+24], the decision was made to use it here as well so it can possibly be incorporated into their system with less effort. *d3.js* is a library for interactive data visualization with Scalable Vector Graphics (SVG) that can handle large and dynamic datasets and includes force simulation basics, making

it appropriate for the intended design. The web application allows the selection of the data source by fetching the data from a user defined HTTP server.

5.2 Connectivity Network Generator

The CNG is a python package that contains the implementation of the pipeline shown in Figure 4.1. The functionality is offered in the form of a CLI that can be installed with a python package manager like *pip* [Bc]. For an even easier installation that isolates the application in its own virtual environment, one can use an application management tool like *pipx* [Scb] or *uv* [MBc]. As it is, the CLI comes with three commands that separate the pipeline into three parts:

1. `cng generate_meshes CONFIGURATION_FILE`
2. `cng split_glomeruli CONFIGURATION_FILE`
3. `cng generate_network CONFIGURATION_FILE`

All commands make use of a configuration file that contains the parameters of the respective step. Due to its minimal syntax and good readability, YAML [EBKdN] was chosen as the data format for the configuration. The implementation of the network generation pipeline will be discussed by explaining the three commands above in more detail.

5.2.1 `cng generate_meshes`

This command covers the first four steps of the network generation pipeline seen in Figure 4.1 and results in individual mesh files for the connected component of the FTU that is specified in the configuration. Since all collaborators supplied their volume data as OME-TIFF [GAB+05] files, the mesh generation was implemented in a way that requires the volume data to be a multipage TIFF file. As the input can not only be a binary segmentation mask, but also contain multiple components, and since the parameters for mesh generation can vary depending on the FTU, the configuration was split into two files. One file specifies the FTUs that are contained in the volume data, while the one passed to the command specifies the applied parameters and FTU that should be used for the mesh generation.

Listing 1 shows an example of a volume data configuration. The configuration can contain multiple FTU definitions. If the dataset file specified with `filename` also contains multiple FTUs, `is_labeled` has to be set to `True` and `id` has to match the label in the volume data. In case binary data is already representing connected components, `has_components` needs to be set to `True` to skip the first step of the pipeline. `original_z_size` specifies the size of the sample on the Z-axis, while `current_z_size` is the size of the segmented volume data in the Z-axis. The *z_depth*,

```yaml
name: kidney
directory: E:/Studium/MedInfo/Masterarbeit/Data/kidney
ftus:
  ducts:
    id: 1
    is_labeled: False
    filename: ducts_binary.ome.tiff
    has_components: False

  glomeruli:
    id: 2
    is_labeled: False
    filename: gloms_binary.ome.tiff
    has_components: False

  nerves:
    id: 3
    is_labeled: False
    filename: nerves_binary.ome.tiff
    has_components: False

original_z_size: 1148
current_z_size: 2131
```

Listing 1: An example for a volume data configuration with three FTUs from three OME-TIFF files containing binary segmentation masks.

calculated as a ratio between the two sizes (see Equation (5.1)), is an aspect ratio that is used to correct for the stretching of the anatomy caused by downsampling.

$$z\_depth = \frac{original\_z\_size}{current\_z\_size} \tag{5.1}$$

Listing 2 shows a complete example of a configuration file that can be passed to the command. The volume data is linked by supplying the corresponding configuration file. This file also points to the *Dask* configuration, which will be further discussed in Section 5.2.4. For the mesh generation itself, the FTU is specified with the key `ftu` and the downsampling factor in the X and Y-axis can be set through `downsampling`. To reduce RAM usage and also, parallelize processes, the pipeline cuts the data into chunks at multiple points. The 3D size of those chunks can be defined with `chunk_size` with the axes corresponding to $[z, y, x]$.

In the first step of the mesh generation, the connected components of a FTU are calculated.

```yaml
data:
  configuration: volume_data_config/kidney_data.yaml
  output_directory: output

dask:
  configuration: pipeline_config/dask_config.yaml
  num_of_workers: 8

mesh_generation:
  ftu: glomeruli
  downsampling: 1
  chunk_size: [200, 1024, 1024]
  max_meshes: null
  min_voxel_count: 50000
  skip_volumes_on_boundary: False
  simplification_ratio: 0.1
  file_format: ply
  create_fbx_scene: False
```

Listing 2: An exhaustive configuration for the `generate_meshes` command, containing parameters for a glomeruli dataset.

For this, the Python library connected-components-3d (cc3d) [Sil] is employed, because it implements a state-of-the-art algorithm in C++. The algorithm is pixel-based like ICTCL but adapted for 3D and uses a decision tree based on the optimized connected component labeling algorithm [WOS09]. The resulting connected component labeling data is stored in a memory-mapped file with *NumPy* [Oc] to save RAM and subsequently saved to disk as a compressed TIFF file.

For every component, the bounding box, voxel count, and centroid is calculated with cc3d and stored in a separate statistics file. To improve parallelization, the largest k meshes are extracted and saved to individual compressed TIFF files with the offset of their bounding box stored as metadata. k is specified by setting `max_meshes` in the configuration file. If `max_meshes` is `null` all the connected components are saved as files. Similarly, the minimum voxel count a component needs to have can be specified with `min_voxel_count`, but can also be 0.

After the connected components are found, their statistics calculated and the individual files are generated, the meshes are created through a custom implementation of the process designed in Section 4.2.2 that utilizes the marching cubes implementation from the *scikit-image* [vdWc] library for isosurface extraction. All previously created component files are created in parallel. In case a component is bigger than the defined `chunk_size`, it is split into chunks and the process is further parallelized by creating

parts of the mesh and stitching it back together as shown in Figure 4.3. By setting `skip_volumes_on_boundary` to `True` the components that touch the volume data boundary are skipped when generating the meshes.

After isosurface extraction, every mesh is simplified with the *fast-simplification* [Kca] Python package, which implements the quadric error metric-based edge collapse algorithm by Garland et al. [GH97] described in Section 4.2.3. The user can specify the reduction ratio of triangles in the mesh by setting `simplification_ratio`. For example, if the ratio is set to 0.1, as can be seen in Listing 2, the algorithm tries to reduce every mesh's triangle count to 10%.

Finally, the position of the vertices are corrected by shifting them by the offset stored in the metadata and scaling them with the *z_depth* ratio (Equation (5.1)). The individual mesh files are then generated in the format specified by the configuration key `file_format`. The library used to save the meshes is *meshio* [Sca], so the mesh export supports all file formats of this package. Optionally, the meshes can be combined in a single scene and exported as a FBX file when `create_fbx_scene` is configured with `True`. All output files are stored in the directory specified with `output_directory`. The tool automatically creates a readable folder hierarchy that mirrors the configuration, so that the output of multiple runs and also multiple input datasets can be stored in the same directory.

5.2.2 `cng split_glomeruli`

This command implements the specific quality assurance steps for kidney datasets designed in Section 4.2.4. It is intended to be run between the mesh generation and network generation and serves as an example of possible quality assurance steps that can be developed for other datasets at this point in the pipeline.

Listing 3 shows a configuration file that can be provided for this command. By specifying the keys `mesh_generation_output`, `dataset_name`, `mesh_configuration` and `mesh_file_format` the tool is able to load the corresponding files that were generated with the `generate_meshes` command. The clustering is implemented with the help of a *k*-means implementation from the *scikit-learn* [Ccb] library, since the library is needed

```
mesh_generation_output: output
dataset_name: kidney
mesh_configuration: 1_38comps_10pfaces
mesh_file_format: ply
glom_mean_size: 162.44107176765564
distance_to_border: 5
create_fbx_scene: False
```

Listing 3: An exhaustive configuration for the `split_glomeruli` command.

for another algorithm and its speed is not pivotal for this computation. The mean size of the bounding box of all non-merged meshes that is used for the calculation of the number of clusters necessary for the k-means clustering, is specified with the configuration key `glom_mean_size`. For the mesh manipulation with cutting planes, the Python helper module *PyVista* [Scc] for the `C++` *Visualization Toolkit* (VTK) [Inc] is utilized and to fix the meshes after splitting, an extension module for *PyVista* that implements the *MeshFix* algorithm by Attene [Att10] is used. Meshes with a distance less than `distance_to_border` from the volume border are removed entirely. The meshes are exported to a folder named *postprocessed* located in the directory of the input meshes. As with the mesh generation, a scene containing all post-processed glomeruli is created when `create_fbx_scene` is set to `True`.

5.2.3 `cng generate_network`

The last three steps of the network generation pipeline seen in Figure 4.1 are implemented by this command. It generates a connectivity network between the meshes of multiple FTUs based on a flexible input configuration that includes FTUs, connectivity metrics as well as merge, network and clustering specifications. As with the other commands, the command is configured with a configuration file that is passed as an argument. Listing 4 shows the first part of such a configuration file. The output directory and volume dataset name are configured with `mesh_generation_output` and `name` respectively. Like the `generate_meshes` command, the configuration includes *Dask* [Roc] which will be covered in Section 5.2.4.

In a first step, the connectivity metrics defined by `metric_configurations` are calculated between the FTUs specified in `input_configurations`. Possible parameters for the metric calculation can also be set, as can be seen with the `DIRECTION_ANGLE` in Listing 4. Currently, the following four metrics are implemented:

- **`EUCLIDEAN_DISTANCE`:** The distance between the closest vertices of two meshes is found with the help of a k-d tree implementation from the *scikit-learn* package, which is used for its speed due to the use of *NumPy*.
- **`LOCALIZED_EUCLIDEAN_DISTANCE`:** The localized variant of the Euclidean distance is implemented as described in Section 4.3.1. The process is additionally split into chunks to prevent heavy RAM usage.
- **`DIRECTION_ANGLE`:** The calculation of a_{PCA} is implemented as designed in Section 4.3.2 by utilizing *NumPy* for the PCA and closest neighbors computation.
- **`CONNECTION_ANGLE`:** As with a_{PCA}, a_{Bridge} is implemented as designed in Section 4.3.2 with the help of *NumPy*.

All metric calculation implementations support parallelization, which will be further discussed in Section 5.2.4.

```yaml
data:
  mesh_generation_output: output
  name: cells

dask:
  configuration: pipeline_config/dask_config.yaml
  num_of_workers: 8

input_configurations:
  - ftu: nerves
    configuration: 1_500comps_10pfaces
    file_format: ply
    color: 0xffdc30
  - ftu: glomeruli
    configuration: 1_38comps_10pfaces\postprocessed
    file_format: ply
    color: 0xde2716

metric_configurations:
  - a: nerves
    b: glomeruli
    metrics:
      - EUCLIDEAN_DISTANCE
  - a: nerves
    b: nerves
    metrics:
      - EUCLIDEAN_DISTANCE
      - DIRECTION_ANGLE:
          vertex_range: 20
      - CONNECTION_ANGLE
...
```

Listing 4: The first part of the configuration file for the `generate_network` command, including the data, dask, FTU input and connectivity metric configuration.

Listing 5 shows the second part of a configuration file that can be supplied to the `generate_network` command, and contains the configuration for the graph creation and clustering described in Section 4.3.3 and Section 4.3.4 respectively. With the key `merge_configuration`, the user can specify components that should be merged together based on metric thresholds. This is a generalized way to implement the connectivity suggestion from Section 4.3.2 that allows even more freedom. The configuration in Listing 5 shows an example of how to merge nerve components based on the metrics

```
...
merge_configuration:
  - a: nerves
    b: nerves
    merged_by:
      - type: EUCLIDEAN_DISTANCE
        value: 25
      - type: DIRECTION_ANGLE
        value: 20
      - type: CONNECTION_ANGLE
        value: 20

network_configuration:
  name: glomeruli_nerves_network_pipeline
  connectivity_configuration:
    - a:
        id: glomeruli
        nodes_without_links: True
      b:
        id: nerves
        nodes_without_links: False
      connected_by:
        - type: EUCLIDEAN_DISTANCE
          value: 25

clustering:
  ftu: glomeruli
```

Listing 5: The second part of the configuration file for the `generate_network` command, including the merge, network and clustering configuration.

designed in Section 4.3.2. The implementation merges the components by giving them a new identifier and considering them as a single component for the subsequent network generation. The merging of components is optional. In the `network_configuration` a name for the network is given and multiple `connectivity_configurations` define the generated graph. Every connectivity configuration specifies two FTUs and a series of metric thresholds placed under `connected_by`, below which a link is created between the components of the FTUs. A node is created for every component of both FTUs that has a link to another component. The user can also include all components of a FTU by setting `nodes_without_links` to `True`. The network creation is implemented solely with multidimensional *NumPy* arrays and Python dictionaries. Optionally, an interconnectivity clustering as designed in Section 4.3.4 can be created. This will be

stored in addition to the nodes and links of the generated network. To achieve this, the FTU whose interconnectivity is of interest, needs to be specified. Currently, the implementation is rather basic and considers all other FTUs as the connectors and always clusters in pairs. The implementation utilizes *NumPy* and recursive methods.

Lastly, a directory named `network_configuration.name` is created and the nodes and links of the graph together with the potential clustering are saved into it as `network.json`. Then, the folders containing the relevant meshes are symlinked into this directory so that it can be served with a HTTP server. The created directory can be found at:

```
[OUTPUT_FOLDER]/[DATASET_NAME]/networks/[NETWORK_NAME]
```

5.2.4 Parallel and Distributed Computing

Most processes of the network generation pipeline that need heavy computation or consist of many tasks are implemented in a way that they can be run in parallel. For this, the parallel and distributed computing library *Dask* [Roc] is used, since it is a pure Python solution and has a good *NumPy* integration. The configuration of both the `generate_meshes` and `generate_network` command allows for specifying a *Dask* configuration file with `dask.configuration` and the number of workers with `dask.num_of_workers` in case of distributed computing. Listing 6 shows a basic *Dask* configuration file with the configuration for a single machine at the top and the configuration for a local cluster at the bottom. This configuration file is also supplied to *Dask* so any kind of additional configuration that the library allows can be placed inside it. However, the implementation of the CNG currently only supports *Dask* to be run on a single machine or a local cluster.

```
# Single Machine Configuration:
scheduler: processes

OR

# Local Cluster Configuration:
jobqueue:
  local:
    name: Simple Local Scheduler
```

Listing 6: A basic *Dask* configuration file showing both single machine or local cluster configuration.

5.2.5 Jupyter Notebooks

For more fine-grained control of the CNG package as well as experimentation or extension purposes, four *Jupyter* [Pc] notebooks that cover the same functionality as the CLI are included in the solution. If used, the installation of a virtual Python environment is suggested. All the necessary dependencies can be installed through the requirements file with:

```
pip install -r requirements.txt
```

5.3 Web Application

The visualization is implemented as a web-based application, to support compatibility with most operating systems in a web browser. Since the project was created with the idea in mind to integrate it into Vitessce [KGM+24], the *React* [Mc] library was used as a backbone. The project does not use a special framework with *React*, since it was never intended to be its own big application, but rather a tech demo or plugin that can be integrated with a larger system. By combining the base *React* library with the *JavaScript* module bundler *webpack* [Kcb], a very slim distribution of the web application is achieved, while allowing an easy integration of *TypeScript* [HM] and Syntactically Awesome Style Sheets (SASS) [NW], which improve the development experience. When opening the app in a browser, the user is prompted to enter the address of the HTTP server that exposes the network data which was generated with the CNG. After that, the user is redirected to the main user interface, which is implemented with three distinct but synchronized views as designed in Section 4.4.

5.3.1 Graph Views

The graph views use the library *graphology* [Pli22], which is a multipurpose graph implementation for *TypeScript*, to store node and edge attributes as well as gain information about neighbor relationships. For the visualization, both views utilize interactive SVG elements that are controlled by *d3.js* [BDH+], which stands for *Data-Driven Documents* and is a library that specializes in SVG visualizations. This includes the effects that result from the interactive mechanisms described in Section 4.5.

Connectivity Overview

To visualize the connectivity overview as designed in Section 4.4.1, a custom graph implementation, called a `ClusterGraph`, was developed. It extends the `UndirectedGraph` of the *graphology* library and uses the clustering created by the CNG to construct a hypergraph with three kinds of nodes: simple nodes, connecting nodes and cluster nodes. A connecting node might be a connecting FTU like a nerve or a vessel, while a cluster node can contain all three kinds of nodes as well as their connecting edges. This creates a tree-like data structure that is flattened when a cluster is expanded.

The visualization uses the nodes and edges of the `Clustergraph` and assigns them SVG objects with *d3.js*. By utilizing the *d3-force* module, a force directed layout that adjusts the position of the SVG objects based on a force simulation, is created. For this, a link force, a many-body force and a position force on the y-axis is combined with a boundary force from an additional d3 plugin called *d3-force-boundary* [Gó]. When a cluster node is clicked, it expands as described in Section 4.5, the SVG objects for the contents are generated and the force simulation is restarted to position the new objects. In case that there is no clustering defined in the loaded `network.json` the `ClusterGraph` is replaced by a simple `UndirectedGraph` and a similar less sophisticated view that is equipped to handle that.

Detail View

When a node or component is selected in either the connectivity overview or the 3D mesh view, an ego graph is constructed from the initial graph, with the help of an `UndirectedGraph` from the *graphology* library. This is implemented to work as described in Section 4.4.2, by constructing the graph starting from the selected node, while eliminating edges between nodes of the same layer and removing nodes that are not part of the ego graph. The nodes and edges of the generated ego graph are then used by a custom ego graph view implementation that creates SVG elements with *d3.js* and places them like designed in Figure 4.19. The ego graph node is placed in the center and the nodes of consecutive layers are placed on a circle around it at a fixed angle. After placing the nodes the edges are created by dividing the space between the layers dependent on the edge count and circling them around until they reach both of the nodes they connect.

5.3.2 3D View

For the 3D rendering of the component mesh files, the *JavaScript* library *three.js* [Cca] is employed. The implementation of the 3D view uses a renderer provided by *three.js* that employs *WebGL* [Gro] to utilize the GPU of a PC. As it is, the implementation only supports loading the mesh files in the polygon file format (PLY) [Tur94]. This format was chosen since it has a small uncompressed binary file size and *three.js* comes with a `PLYLoader` implementation that allows fetching the file asynchronously from a web server. The meshes are fetched with this loader and put into a *three.js* scene with a material that matches the color that was specified in the network generation configuration (see Listing 4). Basic lighting is added through a `DirectionalLight` combined with an `AmbientLight`. For the camera, the `PerspectiveCamera` together with the `OrbitControls` implementation from *trhee.js* is used to navigate through the scene as described in Section 4.5. The hover effect that has also been mentioned in Section 4.5, is implemented by utilizing the *postprocessing* [vRC] library for *three.js*. In post-processing, a rendering pass is added that puts an `OutlineEffect` on the hovered or selected mesh to achieve a similar visual appearance as in the graph views. To improve the spatial orientation in the 3D scene, an `AxesHelper` in the form of three colored lines for all three axes is added.

CHAPTER 6

Results

In the course of this chapter, the contributions designed in Chapter 4 and implemented in Chapter 5 are applied to real data, while discussing how they answer the posed research questions. The datasets described in Section 2.3 are used as input for the CNG, introduced in Chapter 5, to generate various meshes and connectivity networks of different FTUs and cells. This chapter discusses the results and presents a case study of 9 kidney samples at $20x$ resolution, which compares the results to the analysis by McLaughlin et al. [MZS+24].

For most of the computations and visualizations that are discussed and shown in this chapter, a machine with an AMD Ryzen 7 2700X was utilized. It was equipped with 16GB of RAM, a GTX 1070Ti as its dedicated GPU and a 500GB Solid-State Drive (SSD) for storage. At the time of writing this, the machine can be considered as a slightly above average consumer PC and will be referred to here as *Benchmark PC*.

6.1 Innervation of the PanIN

The labeled volume data of the pancreas dataset provided by Kiemen et al. [KBG+22] has a size of $408 \times 12177 \times 15921$ voxels and is the first dataset that was made available for experimentation. Due to the size, the pipeline was incrementally optimized in regard to RAM consumption and parallel processing. However, creating meshes from this dataset without downsampling still requires a high-end PC, since the dataset contains large components which span the entire volume.

Among other things, the researchers that created this dataset are interested in the innervation of the PanIN that form in the ductal epithelium. Therefore, the meshes for the nerves and PanIN are generated with the CNG. Since this dataset has the largest voxel count, the computation without downsampling is not feasible on the *Benchmark PC*. This is mostly due to the connected component calculation, which is not optimized enough

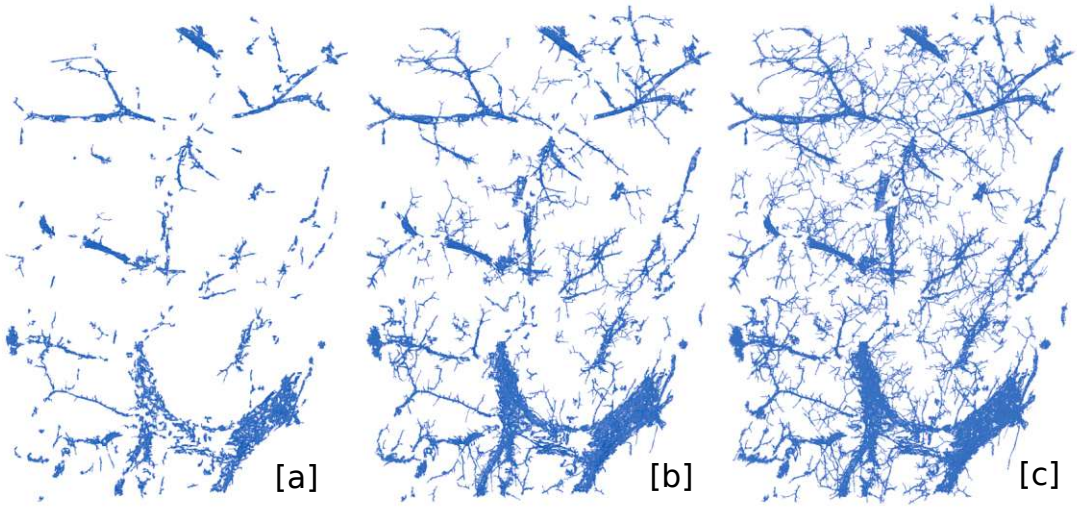

Figure 6.1: The impact downsampling has on the tubular structure of the ductal epithelium in the pancreas. The meshes are downsampled in X and Y by a factor of 4 [a], 3 [b] and 1 [c].

in terms of RAM consumption. Reconstructing the meshes without downsampling has a huge impact on the continuity of tubular structures, as demonstrated with the meshes of the ductal epithelium shown in Figure 6.3. As can be seen, a lot of the fine-grained structure of the ducts is lost with a downsampling factor of 3 compared to 1. Therefore, a comparatively stronger machine, equipped with a Ryzen 9 3950X CPU, 64GB of RAM, a 2TB SSD and a RTX 3080TI as its GPU, is utilized for the mesh generation of the nerves without downsampling. For the lesions, the researchers supplied a dataset containing a component labeling of 54 individual PanIN. Due to this and the fact that the PanIN have a lot more volume than the nerves, there is not much to be gained from generating their meshes without downsampling, and so a downsampling factor of 1 was applied during generation. With the aforementioned system, the mesh generation takes roughly 2 hours, with the bottleneck being the RAM consumption during the connected component labeling and the following statistics calculation. This could definitely be sped up by skipping some steps of the pipeline and tailoring the process to the specific dataset, but the goal was to test the generalized pipeline on various datasets.

The biggest hurdle this dataset poses for the mesh reconstruction is the inaccurate Z-continiuity that stems from the mechanical processes that are employed to cut the sections of the biological sample. Although a registration step is performed in an attempt to stitch the sections back together, Figure 6.2 shows very clearly that this is more than subpar for tubular structures, as the duct was reconstructed as a series of disjoint rings. Therefore, it is fair to say that when the continuity of tubular structures at this resolution is of interest, mechanical cutting of the sample should be avoided. This will further be highlighted through the reconstruction of the kidney datasets (Section 6.3), which were

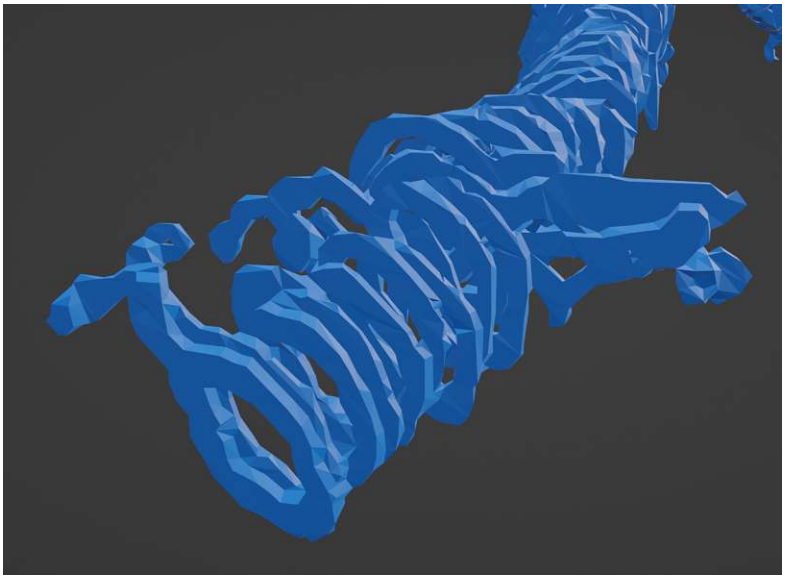

Figure 6.2: A reconstruction of ductal epithelium in the pancreas. Due to mechanical processes in the image acquisition, the Z-continuity of small tubular structures is disrupted.

acquired through volumetric imaging.

However, the nerves of this dataset have slightly more volume, as their labelling does not leave them hollow like the ducts. This marginally improves their Z-continuity, so an attempt at creating the connectivity network between nerves and PanIN is made. For this, nerves with a distance of less than 100 voxels and $\phi < 20°$, $\theta < 20°$ are merged as one component, following the tubular mesh connectivity suggestion principle from Section 4.3.2. A pancreatic intraepithelial neoplasia (PanIN) is considered to be connected by a nerve when their distance is less than 200 voxels. Distances and angles are chosen based on three-dimensional (3D) inspection of the meshes. All pancreatic intraepithelial neoplasia (PanIN) are included in the graph, while only the nerves that form a connection with a pancreatic intraepithelial neoplasia (PanIN) are added. The computation of the connectivity metrics and generation of the network was performed on the stronger machine and took 26 minutes. The duration can be mainly attributed to the distance calculation between meshes that have more vertices because of the lack of downsampling. The result is a graph with 125 nodes and 84 edges.

Figure 6.3 shows the resulting meshes and network visualized in the web application. `panin_segemented_3` is selected, and the connectivity between 6 different PanIN components is highlighted in all three views. As can be seen, many pancreatic intraepithelial neoplasia (PanIN) components are not connected by any nerve, which might be due to a lack of imaging resolution that prevents finer structures from being captured, but could also be the correct biological representation. Re-generating the network based on different threshold parameters, with the metrics already computed, takes less than a minute, which lends itself for experimentation and further analysis. Additionally, an attempt is made to

Figure 6.3: A connectivity network between nerves and PanIN of the pancreas dataset provided by Kiemen et al. [KDB+23], visualized in the web application. The view highlights the connectivity of the selected PanIN `panin_segemented_3` (green) by coloring the meshes and nodes on a gradient according to the layers of the ego graph and the legend in the middle.

incorporate the ductal epithelium into the network generation and visualization, but due to the bad Z-continuity the meshes of obviously connected components are scattered into small parts even with low downsampling. Consequently, the connectivity between ducts, nerves and PanIN is ambiguous. Therefore, the small and discontinuous components of the ductal epithelium are removed and the limits of the web applications are tested by generating a network between nerves, ducts and PanIN. Like with the PanIN before, ducts are considered to be connected to nerves when their Euclidean distance is below 200 voxels. As PanIN are lesions that grow in the ductal epithelium, they form an edge with a duct if they are in the very close proximity of 20 voxels. In reality, there should not be a gap between a PanIN and the ductal epithelium, but because of the Z-continuity problem one exists in some cases.

Figure 6.4 shows the resulting graph and meshes in the web application. Since there are three FTUs that can be interconnected with every other type, there is not one FTU that could be considered as the intermediary connector, and so the edges have no specific coloring. Compared to the previous graph in Figure 6.3, this graph shows far more connectivity, which might be relevant when investigating the innervation of the PanIN. Figure 6.5 shows the selection of the same PanIN as in Figure 6.3 with the ductal epithelium included. The ego graph is limited to show only a 3-alter ego graph, since the high connectivity would lead to a lot of clutter as there are 11 layers. Because there is no intermediary connector, the edges in the ego graph simply represent a closeness of the respective nodes and not a node bundling.

With the size of the connectivity graph almost being at the limit, it would be worth considering designing a clustering process for graphs that have multiple FTUs. This could be done in a similar way to the already implemented clustering by taking the inter-connectivity count of nodes as the clustering metric, but could also be counterproductive for the analysis as its focus would not be on the innervation of the PanIN.

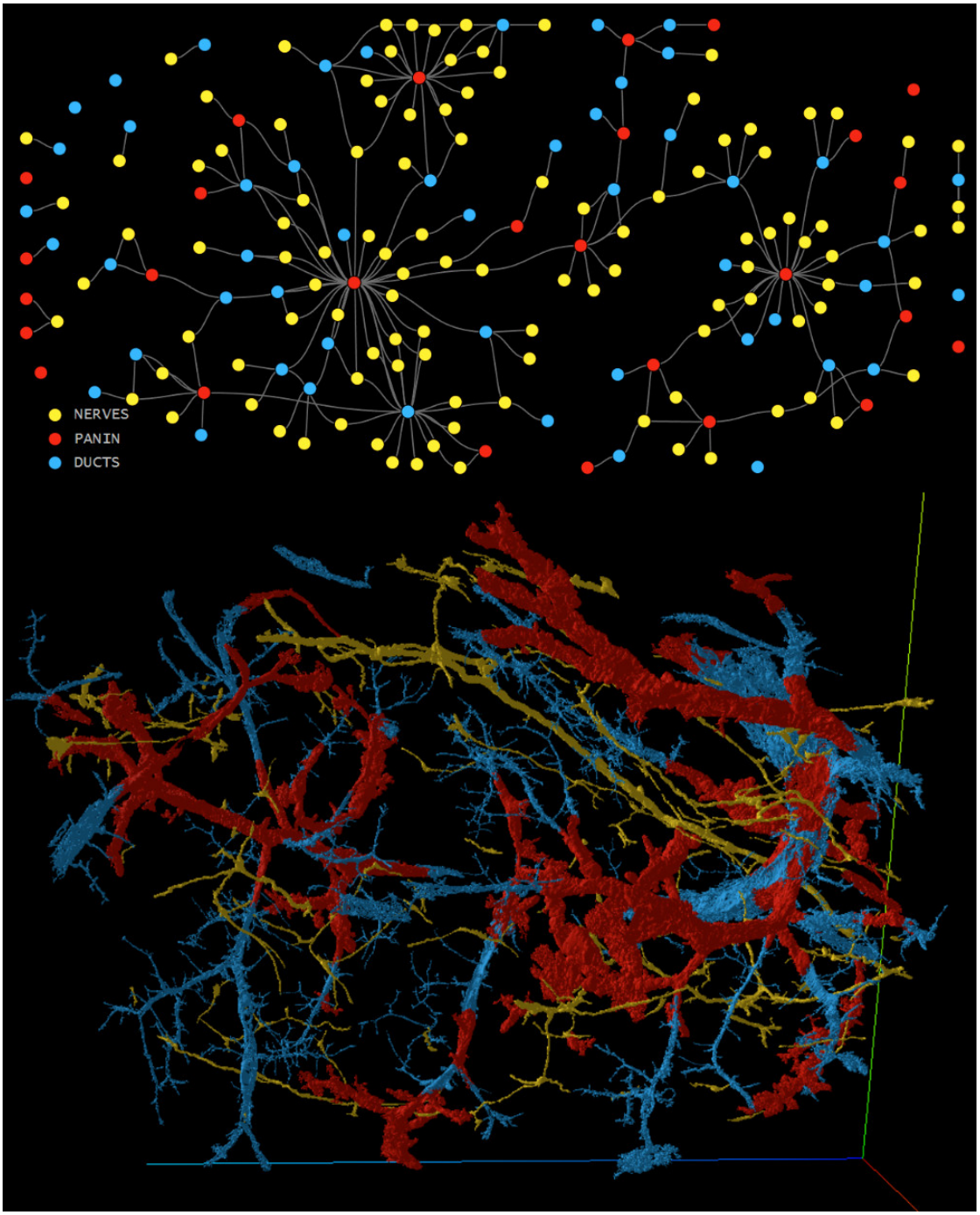

Figure 6.4: The meshes (bottom) of the nerves (yellow), PanIN (red) and ductal epithelium (blue) of the pancreas dataset provided by Kiemen et al. [KDB+23] together with a connectivity network of the three FTUs (top).

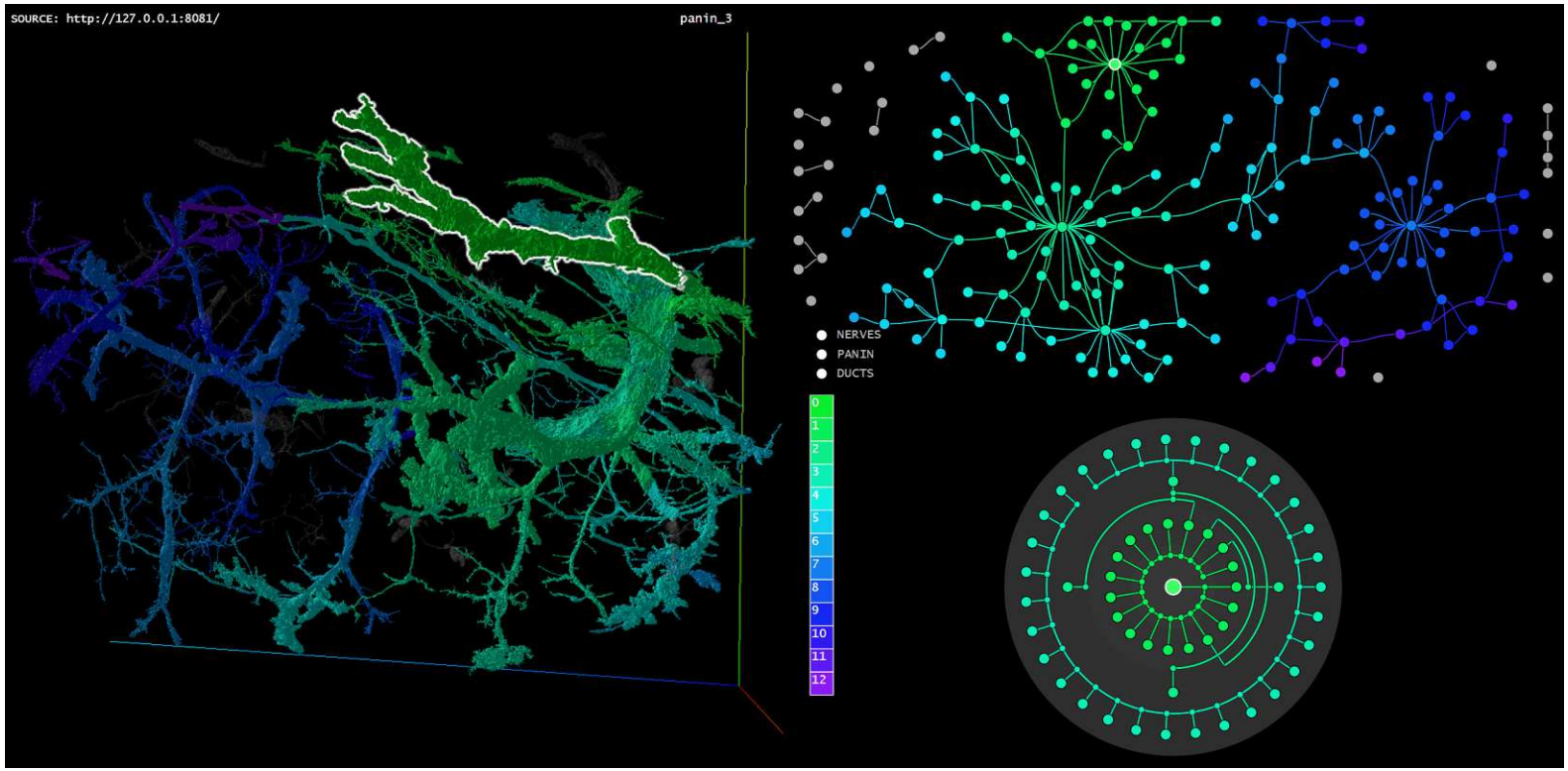

Figure 6.5: 591.5302ptA connectivity network between nerves, PanIN and ductal epithelium of the pancreas dataset provided by Kiemen et al. [KDB+23], visualized in the web application. The view highlights the connectivity of the selected PanIN `panin_3` (green) with an ego graph that is limited to 4 layers. The meshes and nodes are colored on a gradient according to the layers of the ego graph and the legend in the middle.

6.2 Melanoma Cells

Yapp et al. [YNZ+24] supplied two volumetric datasets of melanoma cells, one with 11,000 cells and one with 43,000 cells. The volumes of the individual cells are comparatively small, but the sheer amount of cells sparked the development of the heavily parallelized pipeline, and specifically that of the localized Euclidean distance computation discussed in Section 4.3.1.

The researchers providing this dataset are especially interested in the interconnectivity between touching cells. Because of this, the meshes of all cells are generated with the CNG without any simplification to get the most accurate representation of cell proximity. Generating the meshes takes between 10 to 20 minutes on the *Benchmark PC*, with parallelization and without downsampling. The meshes are then used to create a network based on the Euclidean distance to their immediate neighbors. Generating said network on the *Benchmark PC*from meshes without downsampling takes another 10 to 15 minutes.

Since visualizing this many cells is challenging in respect to both 3D and graph visualization, alternatives like *neuroglancer* were explored. Because the meshes can not be visualized in the developed tool and the nature of the connections is different when compared to the other datasets, more scalable solutions for the graph visualizations were investigated. For the final visualizations, the tool *Cosmograph* [RO] is used. This tool utilizes GPU-accelerated WebGL to render a force directed graph.

Figure 6.6 and Figure 6.7 show the generated meshes visualized in *neuroglancer* after converting them into the *precomputed* format described in Section 4.4.3. The meshes are put next to the *Cosmograph* visualization of the generated network. Since there is no synchronization between the views, as they are used as standalone applications, the nodes of the graph are projected onto the X and Y coordinates of the representing cell's centroid. Furthermore, individual cells are colored with the same random seed in both representations. Because the sample is rather thin, this works quite well without cluttering the view too much and allows connecting the views through spatial alignment and color matching. Zooming, panning and rotating the 3D view in *neuroglancer* works well with 11,000 cells, but with 43,000 cells the *Benchmark PC*struggles to keep up and the scene takes a long time until the correct LOD is loaded initially. When inspecting multiple regions of interest RAM still becomes a problem on the larger dataset.

Integrating both *neuroglancer* and *Cosmograph* into a linked and synchronized view similar to the web application could improve the analysis of this dataset further. However, generating the *precomputed* format is currently not that simple, since it requires multiple build steps until the pipeline can be utilized. To make this more accessible, it should be integrated into a tool like the CNG that can be installed with a single command.

In the frame of connectivity, Yapp et al. were also interested in the surface area of adjacent and touching cells. Designing and implementing a custom connectivity metric that is calculated from the touching surface area between two cells would allow the network generation based on this metric, but was not in the scope of this thesis.

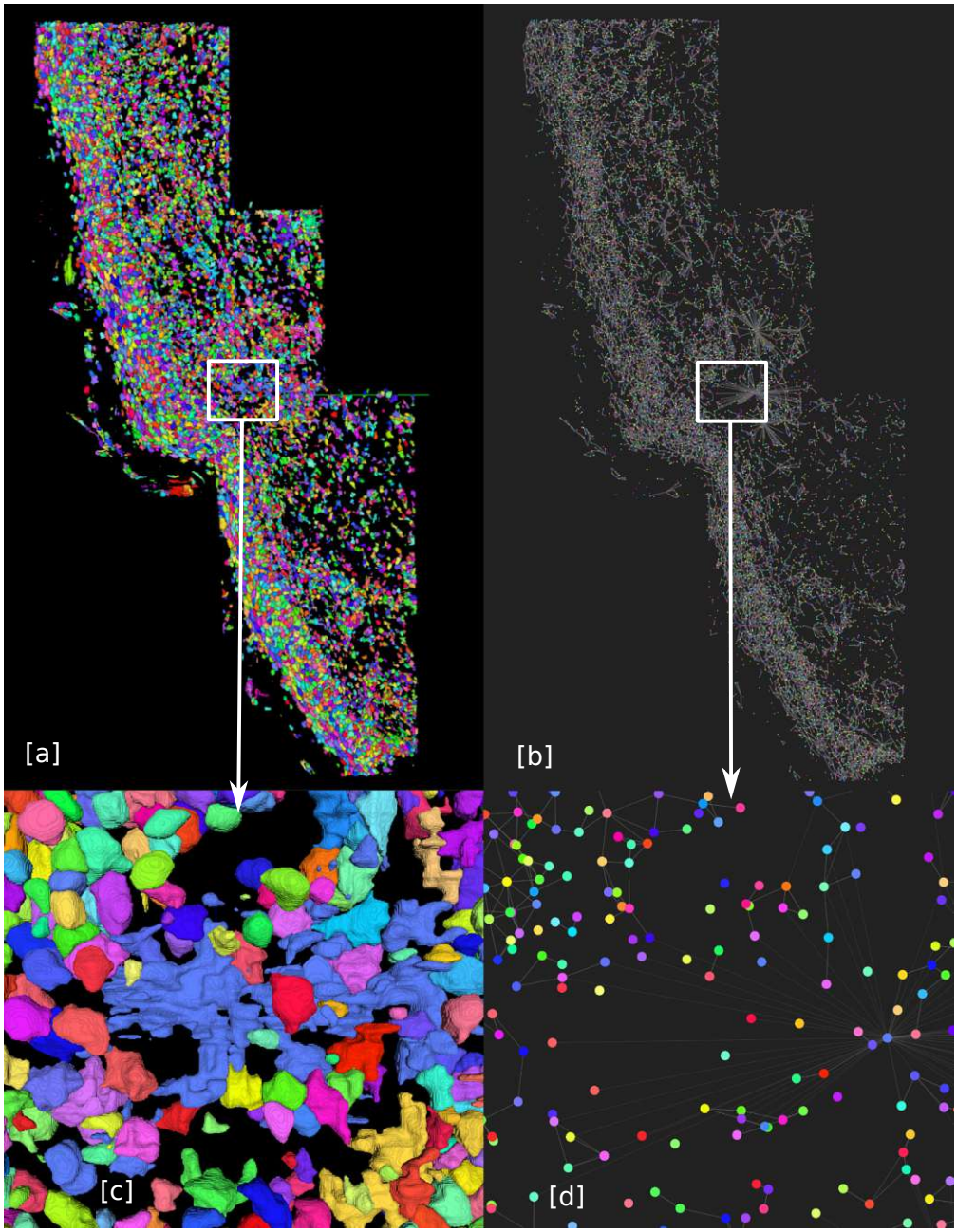

Figure 6.6: The meshes [a] and connectivity network [b] from the melanoma dataset with 11,000 cells visualized with *neuroglancer* and *Cosmograph* respectively. [c] and [d] show the same zoomed in section in both views. The nodes of the network are projected onto the X and Y coordinates of the mesh centroids.

Figure 6.7: The meshes [a] and connectivity network [b] from the melanoma dataset with 43,000 cells visualized with *neuroglancer* and *Cosmograph* respectively. [c] and [d] show the same zoomed in section in both views. The nodes of the network are projected onto the X and Y coordinates of the mesh centroids.

6.3 Case Study: Glomeruli Interconnectivity

McLaughlin et al. [MZS+24] provided 10 kidney datasets. One dataset at 5× magnification and 9 datasets containing regions of interest in the former dataset that were magnified by 20×. Since their work is closely related to interconnectivity and multiple datasets are available, a case study is conducted that tries to reproduce their findings with the developed method. The 5× dataset is disregarded, as the resolution is too low for the automatic generation of glomeruli meshes. Figure 6.8 depicts the largest three glomeruli meshes that were reconstructed from the 5× dataset. The merged meshes contain dozens of individual glomeruli, and splitting them would need human input from someone that has expertise in the field. Therefore, the focus of the case study lies on the 5× datasets.

The researchers used manual tracing and automated processes to create a graph for each of their datasets, with the glomeruli as nodes and the nerves that have left the vascular pole as edges. The glomeruli were dilated by 10μm in every direction to find the overlap with the neighboring nerves. In case of overlap, an edge between glomeruli and nerve was created. From the resulting networks, they derived seven motifs [MSOI+02], which they split into three groups:

- **Intra-community motifs:** Describe the basic interaction between glomeruli and nerves.
- **Community motifs:** Show patterns of aggregation in a community of glomeruli.
- **Inter-community motifs:** Depict relationships between different communities.

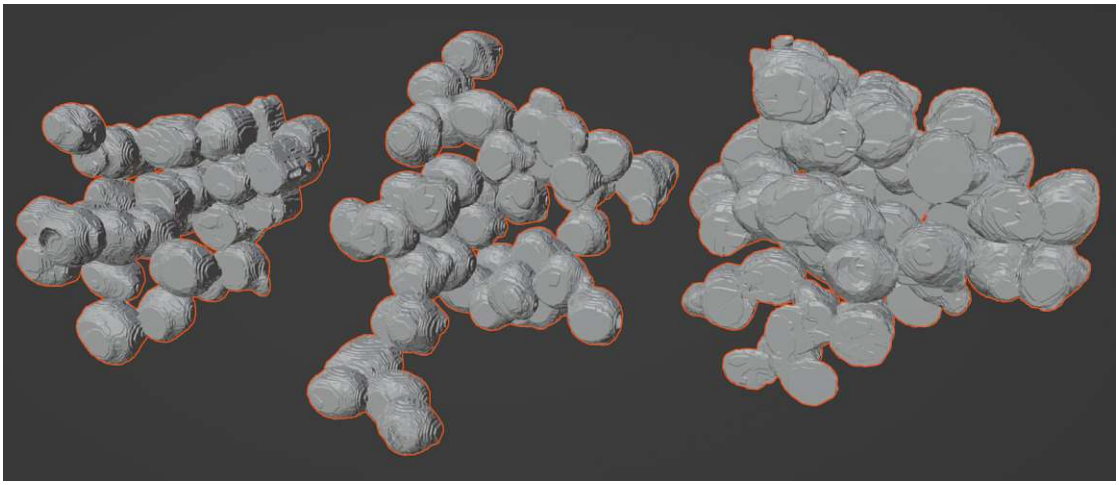

Figure 6.8: The three largest glomeruli meshes that were reconstructed from the 5× datasets provided by McLaughlin et al. [MZS+24]. Multiple individual glomeruli appear to be merged into large connected meshes.

Of the following seven motifs, the first three are considered intra-community motifs, followed by two community motifs and the last two are inter-community motifs [MZS+24]:

- **Lone Motif:** Consists of a single glomerulus that is innervated directly from the main nerve next to the interlobular vessel (Figure 3.1A).
- **Paired Motif:** Two glomeruli that are in close proximity to each other, are supplied by a forking afferent arteriole, and have a tightly linked neural network (Figure 3.1B).
- **Pyramid Motif:** A glomerulus that appears to be upstream of a glomeruli network by being connected to every glomerulus in the network, while resembling a pyramid in 3D. (Figure 3.1C).
- **Keychain Motif:** A circular connectivity pattern between multiple glomeruli with similar node degree in an area where their density is low. (Figure 3.1D).
- **Grape Motif:** Highly interconnected glomeruli in a densely packed area where they surround a centralized nervous hub and interlobular vessel. (Figure 3.1E).
- **Hourglass Motif:** Two communities that receive distinct vascular inputs are connected through one or a few glomeruli (Figure 3.1F).
- **Lattice Motif:** Three or more communities are connected by a central glomerulus called the *mother glomeruli* (Figure 3.1G).

For the case study, the 9 datasets at 20× magnification are used to generate the meshes of the glomeruli and nerves and a network is created by choosing the parameters in a way that replicates the process of McLaughlin et al. The generated networks and meshes are then inspected in the developed web application described in Section 5.3 and analyzed in regard to the previously defined graph motifs.

The meshes of the glomeruli are generated with a downsampling factor of 1 in the X and Y axes and components with a voxel count below 50000 are ignored. The faces of the meshes are simplified by a simplification ratio of 0.1. The glomeruli mesh generation takes about a minute for a single dataset on the *Benchmark PC*. Merged glomeruli are split by the implementation of the algorithm designed in Section 4.2.4 and the meshes on the border of the volume sample are removed entirely. The *Benchmark PC*handles this in 30 seconds. The meshes of the nerves are generated without downsampling to preserve the connectivity of tubular structures. Components with a voxel count below 500 are removed to prevent tiny speckles that have no impact on connectivity and might be caused by extremely thin nerve fibers or inaccuracies during the imaging process. As with the glomeruli, the faces are simplified by a simplification ratio of 0.1 to prevent unnecessarily large mesh files. The *Benchmark PC*takes about 3 minutes to create 400 to 900 nerve meshes. The connectivity network is generated by merging nerves that are 25 voxels apart and satisfy $\phi < 20°$ as well as $\theta < 20°$. An edge between glomeruli

and nerves is formed when they are closer than 15 voxels, which equates to the 10μm McLaughlin et al. used for their analysis. All the glomeruli components are added to the graph as nodes, but the nerves are only included if they connect to at least one glomerulus. Finally, the graph is clustered based on interconnectivity of glomeruli, with the nerve nodes as intermediary connector. Depending on the number of nerves, the connectivity metric calculation and consequent network generation takes between 4 to 15 minutes on the *Benchmark PC*.

After using the CNG to generate the meshes and connectivity networks for the 9 datasets, they are served to the web application as a data source. The created networks are searched for the defined motifs by interactively utilizing the three developed visualizations. The findings are discussed in the following sections. The *Lattice Motif* is not investigated, as McLaughlin et al. did not provide the larger 20× datasets they used for the analysis of cross community connectivity.

Lone Motif

Since the segmentation mask of the vasculature was not provided by McLaughlin et al., there is no way to tell if the innervation of the glomerulus starts next to the interlobular vessel. However, since all glomeruli were included in the network, independent of whether they have a connection to other glomeruli, the glomeruli which only have nerve connections can be identified. The clustering algorithm creates a cluster for all the glomeruli that have no connections to other glomeruli and terms it `no_connections`. By hovering the node of this cluster in the connectivity overview graph, the location of glomeruli that are part of a *Lone Motif* can be identified in the 3D view as shown in Figure 6.9. The `no_connections` cluster is then expanded by clicking on it. Through the selection of

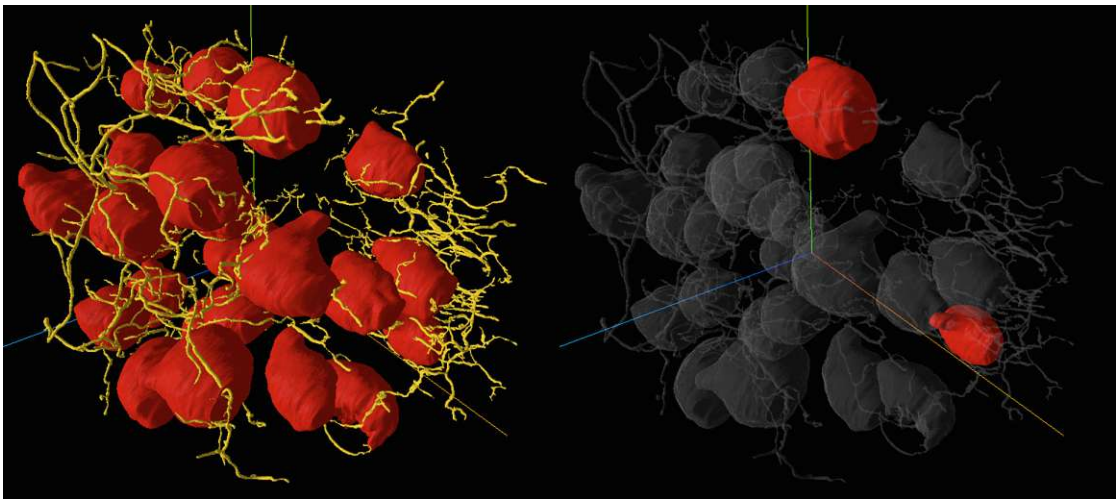

Figure 6.9: The hovering effect of the cluster node named `no_connections` in the 3D view. The meshes show glomeruli (red) and nerves (yellow).

one of the expanded glomeruli, it can be highlighted with all of the nerves that connect to it. Figure 6.10 shows the identification of a network part that corresponds to the *Lone Motif*. The parts of interest are highlighted in all three views, while the unrelated parts are held in the background. In the 9 datasets, a total of 47 *Lone Motifs* could be identified and 3 glomeruli had no nerve connection at all.

Paired Motif

Due to the missing vasculature, glomeruli pairs that are in close proximity and have any nerve connections between them are considered a *Paired Motif*. In order to identify this motif, the connectivity overview is utilized. Because the glomeruli are clustered based on their nerve interconnectivity, the *Paired Motif* always appears as a separate cluster without connections. Figure 6.11 shows how the location of this motif can be highlighted in the 3D view by hovering such a cluster node in the connectivity overview graph. After the pair is highlighted, the close proximity criteria can be visually determined as shown in Figure 6.12. The connectivity can be further investigated by expanding the cluster and selecting any of the contained nodes. The 9 datasets contained 12 *Paired Motifs* where the glomeruli were visually less than the width of a glomerulus apart from each other. Most of the *Paired Motifs* consisted of two glomeruli that had to be split during the quality assurance step of the network generation pipeline.

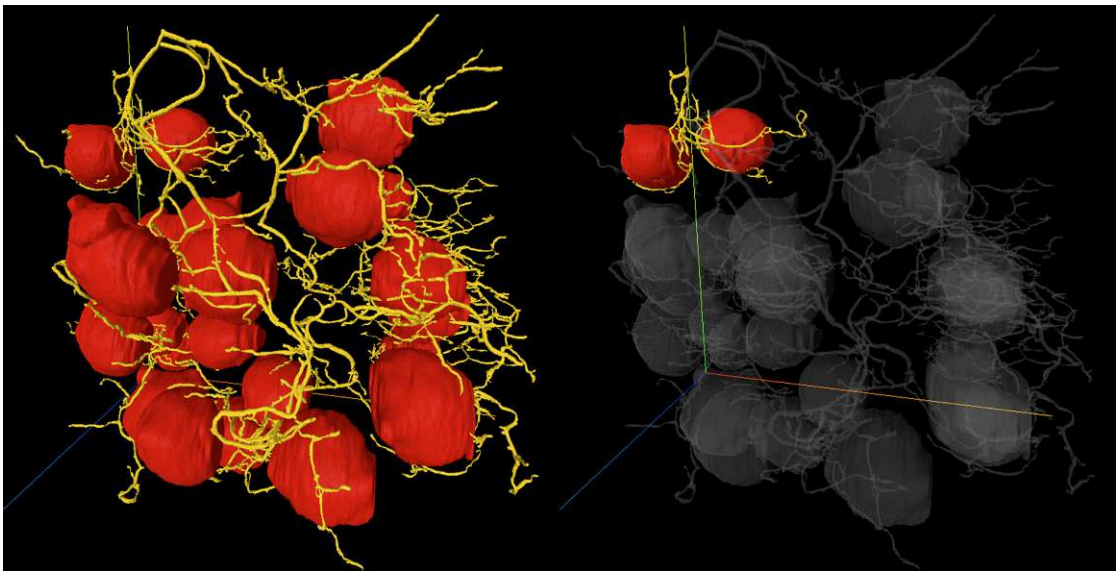

Figure 6.11: The hovering effect, of a cluster node that has no connections to any other node in the connectivity overview graph, on the 3D view. The meshes show glomeruli (red) and nerves (yellow).

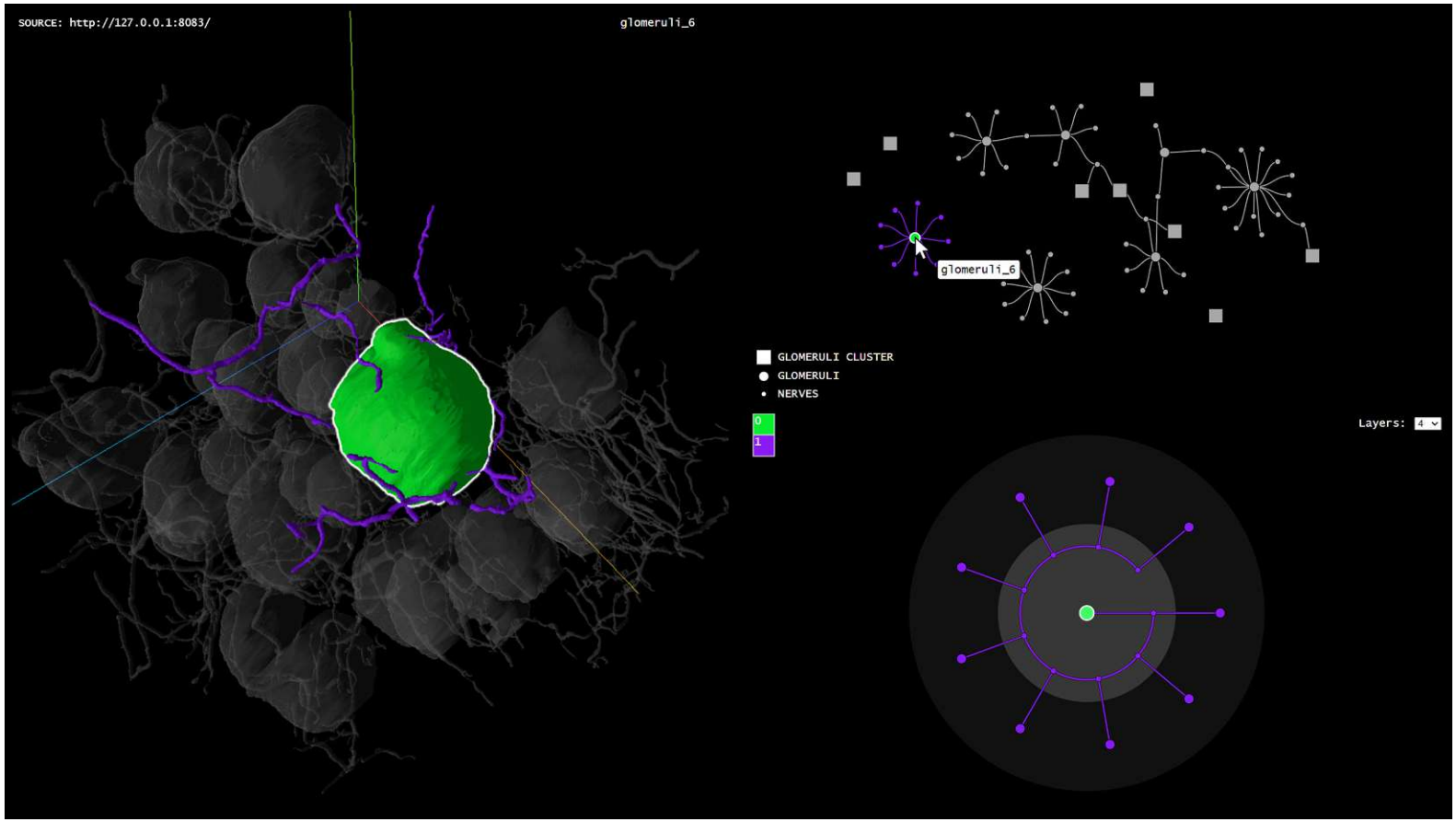

Figure 6.10: The selection of a glomerulus (green) with no connections to other glomeruli highlights the network part that corresponds to a *Lone Motif* in all three views. The nerves that are connected to the glomerulus are display in violet.

Figure 6.12: The selection of glomerulus `glomeruli_33` (green) which is connected to a single other glomerulus (violet) in the web application. This is considered to be a *Paired Motif* as the distance between the glomeruli (red line) is less than that of either of the connecting glomeruli (blue lines).

Pyramid Motif

To identify this motif, the connectivity overview graph is first scanned for sub-graphs with a higher node count than three nodes. Then, an arbitrary glomerulus is selected to highlight its connectivity in the 3D view. The 3D view can then be inspected in regard to the pyramidal shape while taking into consideration the connectivity of the included glomeruli that is shown in the ego graph and switching the selected node in the connected sub-graph. The glomeruli forming the base of the pyramid can be identified by looking at the glomeruli in the 3D view that are colored the same as the first layer of the ego graph. The definition McLaughlin et al. give for the *Pyramid Motif* is rather vague, so the search for this motif is oriented mainly on the figure they published (Figure 3.1). Among the 9 datasets, only 2 sub-graphs that roughly conform with the *Pyramid Motif* are found in the generated networks of two different datasets. Figure 6.13a-b show these two *Pyramid Motifs* with the top node of the pyramid selected. Even though they fulfill the criteria of pyramidal shape in 3D, their base is not fully interconnected, as can be seen in Figure 6.13c-d.

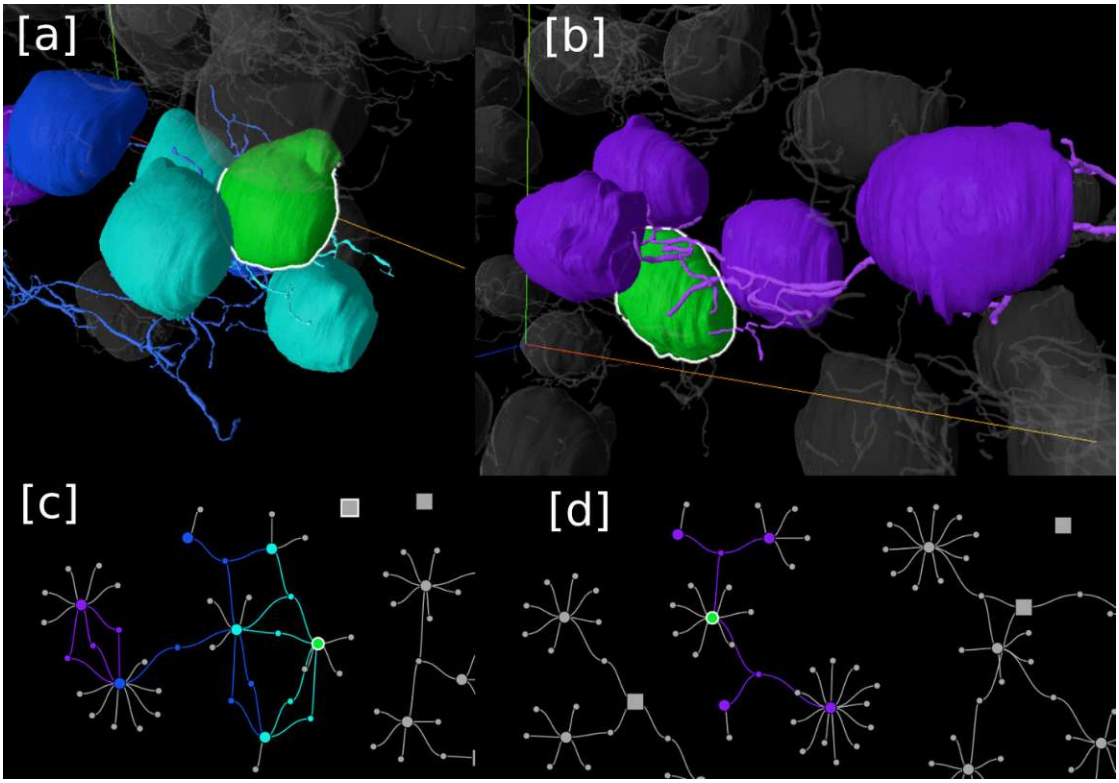

Figure 6.13: Two sub-graphs of the networks generated from two different kidney datasets that most closely conform to the *Pyramid Motif*. The meshes form a pyramid in 3D (a-b) with the tip being the selected glomerulus (green), but the base of the pyramid is not a fully connected network (c-d). The meshes and nodes are colored on a gradient (green to violet) according to their hop distance from the selected node.

Keychain Motif

The *Keychain Motif* is identified by utilizing the connectivity overview graph and the 3D view, since circular connectivity is inherently not visible in the ego graph. First, a single node of a sub-graph that contain more than three nodes is selected in the connectivity overview graph. This highlights the corresponding glomeruli in the 3D view, where the density is inspected. In case of a low density area, both views are used to detect if the connectivity of the sub-graph is circular and if it conforms with the *Keychain Motif*. Only one dataset contains a single sub-graph that can be interpreted as a *Keychain Motif*. Figure 6.14 shows this sub-graph in the 3D view and connectivity overview graph consisting of the green and turquoise meshes and nodes. The node degree is somewhat similar, but the density of the glomeruli is not that low. Through all the datasets, the low density glomeruli seldom had any connection, explaining the low count of *Keychain Motifs*. The manual tracing of nerves by McLaughlin et al. that included expert knowledge might be what led to these diverging results.

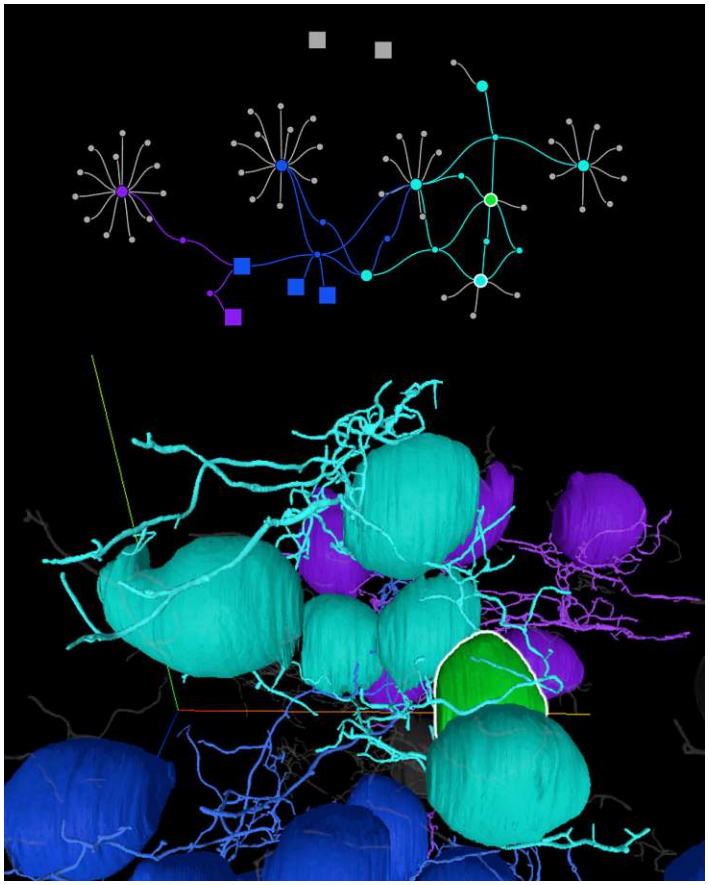

Figure 6.14: The single sub-graph that conforms to a *Keychain Motif* consists of the green and turquoise meshes and nodes showed in the connectivity overview graph (top) and 3D view (bottom).

Grape Motif

Because there is no vasculature, the focus lies on glomeruli that surround a centralized nervous hub. If more than four glomeruli are connected by one nerve while being densely packed together, the sub-graph is considered a *Grape Motif*. Finding this motif is done by looking at the connectivity overview graph and finding clusters or sub-graphs with either a high interconnectivity or nerve nodes that are connected to many glomeruli. After selecting a node of the sub-graph, the density of the highlighted glomeruli is inspected in the 3D view. Using the ego graph during selection of nerves also helps in identifying the grape like structure. In all 9 datasets 10 *Grape Motifs* can be identified. Figure 6.15 shows the largest grape motif that could be found. In this sub-graph, 22 glomeruli are connected by a single central nerve.

Hourglass Motif

Since the distinct vascular input can not be detected, only nerve connections are considered. A sub-graph is believed to be an *Hourglass Motif* if two interconnected communities of three or more glomeruli are connected by one or two glomeruli. This is achieved by selecting nodes of appropriately sized interconnected communities in the connectivity overview graph, while inspecting the connectivity in the ego graph. There, the connection through one or two glomeruli becomes very clear. In case there are more than one glomerulus connecting the communities, the 3D view is inspected to see if they are close to each other. The 9 datasets contained a total of 6 *Hourglass Motifs*. Figure 6.16 and Figure 6.17 show an example of an *Hourglass Motif* sub-graph with scattered and dense communities, respectively, where the connecting nodes are selected.

Total motif counts for every motif resulting from the analysis of the provided 20× datasets are shown in Table 6.1. The number of glomeruli as well as the number of glomeruli that have no nerve connection are included as well. The low count for *Pyramid* and *Keychain Motifs* might be a result of the nerve tracing being done automatically and the missing data for the contextual FTUs also mentioned in the definition of the motifs.

Table 6.1: The glomeruli count, glomeruli without nerve count and motif count for each of the nine 20× datasets provided by McLaughlin et al. [MZS+24].

Motif	01	02	03	05	09	10	11	12	13	Total
Glomeruli	31	22	27	28	25	24	33	23	32	245
No Nerves	0	1	0	0	1	0	1	0	0	3
Lone	8	7	2	3	9	4	4	1	9	47
Paired	1	2	3	0	2	1	0	0	3	12
Pyramid	0	0	0	0	0	1	0	0	1	2
Keychain	0	0	0	0	0	0	0	1	0	1
Grape	1	1	2	1	0	1	2	2	0	10
Hourglass	0	0	1	0	0	1	1	2	1	6

Figure 6.15: The largest sub-graph that is also a *Grape Motif* shown in the web application. The central nerve connecting 22 glomeruli is highlighted in the 3D view (white outline), while all clusters are expanded in the connectivity overview. A glomerulus of the grape is selected (green) to highlight the interconnectivity. The meshes and nodes are colored on a gradient according to the layers of the ego graph and the legend in the middle.

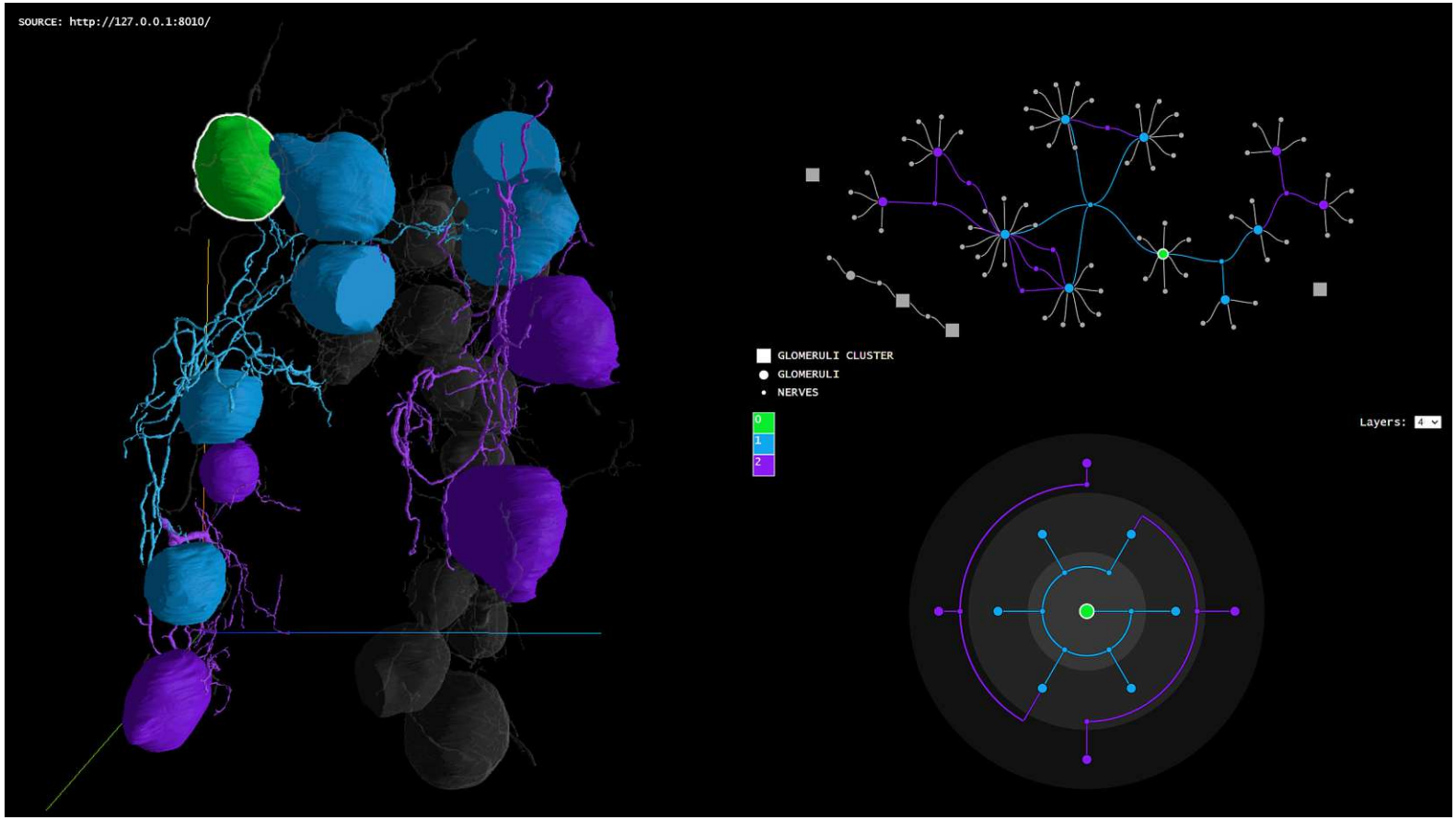

Figure 6.16: An hourglass motif with sparsely connected communities from the dataset kidney dataset `20_10`. The connecting node is selected (green), so the hourglass shape becomes most evident in the connectivity overview graph. The meshes and nodes are colored on a gradient according to the layers of the ego graph and the legend in the middle.

Figure 6.17: An hourglass motif with densely connected communities from the dataset kidney dataset 20_03, that is connected by two glomeruli. One of the connecting nodes is selected (green), so the hourglass shape becomes most evident in the connectivity overview graph. The meshes and nodes are colored on a gradient according to the layers of the ego graph and the legend in the middle.

6.4 Critical Review

One major bottleneck of the CNG is the RAM consumption of the connected component labeling with the following calculation of bounding boxes, centroids and voxel counts. Depending on the available RAM, this works fine for moderately sized datasets but can quickly get out of hand, as shown with the pancreas dataset. The problem is that the process is not parallelizable in its current state due to limitations of the cc3d package. A possible solution to this would be the suggestion by the creator of the cc3d package, that tries to label the volume in slices while stitching consecutive slices together and remapping the labels of the combined section. To save RAM, the slices are stored in a compressed binary format that supports the necessary remapping operation. An attempt was made to incorporate this method, but the binary format package by the same author is in *Beta* state and currently has issues that could not be solved without going beyond the scope of this thesis. The statistics calculation could likewise be done in slices or chunks that are tailored to fit into RAM. This would also allow the parallelization of the whole process.

Even though the connectivity metric calculation was designed so that it can be augmented with custom metric computations, this still needs knowledge in *Python* and the library has to be extended by cloning the repository and updating it. A plugin system that allows the incorporation of third party connectivity metrics would greatly improve the usability for both the developer of said plugins and the users of the tool.

The visualization of the connectivity suggestion for tubular meshes, discussed in Section 4.3.2, is currently only possible when utilizing the *Jupyter* notebook file named `Network_Generation.ipynb` included in the repository (Figure 6.18). This could be improved by showing the user an interactive visualization that changes depending on the chosen thresholds, so that there is no deeper coding knowledge required.

While the CNG supports parallel computing, the distributed computing support is very bare bones and only really supports a local cluster. This could be further improved to support common job queuing systems used in supercomputers of academic research institutions to enable faster computation of huge datasets.

The web application can not handle huge mesh quantities like the ones showcased with the melanoma datasets, as there is no LOD scaling implemented for the 3D visualization of the meshes. In order to support this, a visualization tool like *neuroglancer* would need to be incorporated as well as improving the clustering algorithm and visualization or utilizing a more scalable graph viewer like *Cosmograph*.

The user interface of the web application is purposefully held minimal to keep interactions simple and intuitive. This comes at the cost of functionality that could be useful for further connectivity analysis. The following are possible usability improvements that were discovered during the development and use of the tool through application of real data. In order to increase visibility, the interface can offer a setting that allows the user to toggle the visibility of intermediary connectors that only have one connection.

Similarly, the option to show meshes that are not included in the network, but that are relevant for context, can be added. Allowing the user to add multiple data sources can add context as well as offer an easier way to compare different parameters. Furthermore, making the colors customizable while offering example gradients can greatly improve the accessibility. The scalability of the clustering graph can be improved by making the clusters can manually collapsible or automatically collapsible based on the number of nodes that are visible. The clearness of the ego graph can be further improved by placing the nodes in a way that optimizes the length of edges. Finally, graph metrics like centrality can be presented to the user through visual encodings.

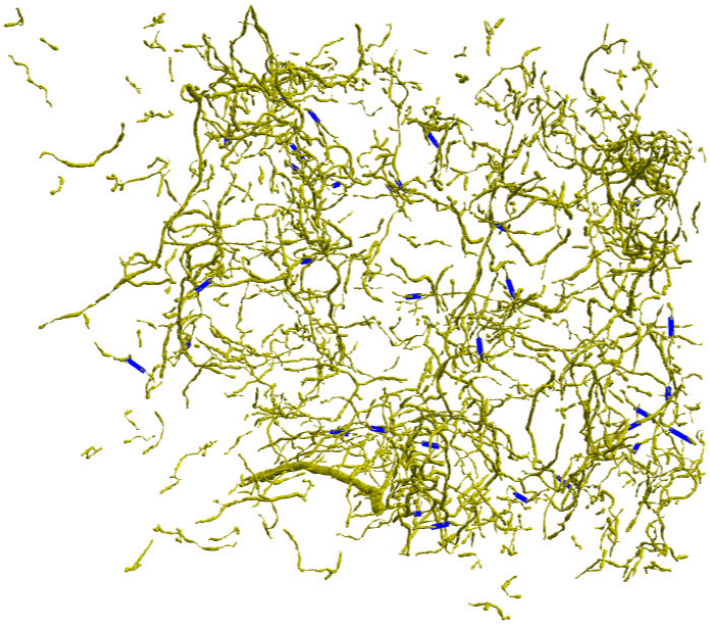

Figure 6.18: A tubular mesh connectivity suggestion visualization of the nerve meshes from a 20× kidney dataset that can be created with the *Jupyter* notebook file `Network_Generation.ipynb`, based on custom threshold parameters.

CHAPTER 7

Conclusion and Future Work

This chapter concludes the thesis by summarizing the presented contributions and discussing their limitations in the current state. Finally, the gained insights are used as a basis to propose possible improvements and future work that could be done in this research direction.

7.1 Summary

Firstly, this thesis presents the Connectivity Network Generator (CNG), an installable *Python* tool that enables the 3D reconstruction of large segmented volumetric data at cell resolution. From the reconstructed meshes, it allows the generation of a connectivity network, based on metric thresholds that are defined by the user (**C1**). This incorporates a newly developed method that suggests the connectivity of incomplete tubular meshes based on the Euclidean distance and the angle metrics θ and ϕ (**C2**). Together, these contributions answer the first research question:

(RQ1) *How can the connectivity between cells or FTUs at cell resolution be effectively abstracted for analysis?* To abstract connectivity correctly and effectively, the captured shape of cells or FTUs has to be preserved, while also reducing the size of the input data to make the computation feasible on the average PC. This is solved by preventing downsampling of the data by using the original volumetric data segmentation to find connected components of cells or FTUs and creating individual meshes for the resulting components. The meshes are then simplified to reduce their size while preserving their original shape and continuity. This results in moderately sized individual components of the original data that can be used in metric calculations that are necessary for the determination of connectivity. Since metrics for the connectivity of two components are an active field of research [YNZ+24, MZS+24], only the basic Euclidean distance metric calculation is implemented, but the system utilizing it is built with the extension of custom metric computation implementations in mind. The connectivity can then be

abstracted by defining the input components together with a list of metric thresholds through which their connectivity is determined. Additionally, tubular meshes can be combined to a single component by specifying the component and metric thresholds for **C2**. Based on the configuration by the user, a connectivity graph is generated. Finally, the graph can be clustered based on the number of connections between components, resulting in a hypergraph that improves the analysis workflow in large networks. The graph is made available as a JSON file that is accompanied by the individual mesh files in a format chosen by the user.

Secondly, this thesis presents a visualization tool for the outputs of the CNG in the form of a web application that fetches the data from an HTTP server and presents it in a manner that allows connectivity analysis workflows (**C3**). This answers the second research question:

(RQ2) *How can this abstraction be incorporated into a visualization tool that enhances research workflows and supports domain-specific analysis?* The actual volumetric data of the components is rendered as colored surface meshes in a 3D view, that can be navigated with a mouse through orbital controls. This view is juxtaposed with two graph views that incorporate the connectivity abstraction generated by the CNG. The first graph view is a connectivity overview that is either a force directed graph or a force directed hypergraph depending on the availability of a clustering. The second graph is a custom-designed radial ego graph that also allows the bundling of components in case of intermediary connecting nodes like nerves or vessels. To enhance the research workflow of connectivity analysis, the three views are synchronized to highlight relevant information in unison, so that each offers a different perspective on the data.

This work yielded the following insights: The continuity of tubular structures is a focus of connectivity analysis. The Z-continuity of these structures depends heavily on the imaging and 3D reconstruction modality that is employed, with volumetric imaging methods producing better results in terms of overall mesh continuity. The proposed method for mesh generation preserves shape well, but is computationally expensive and requires good parallelization and RAM management. Visualizing connectivity with juxtaposed views speeds up connectivity analysis workflows, but requires some sort of level-of-detail zooming to be scalable.

7.2 Limitations

While the presented work offers tools that showcase the possibilities for connectivity analysis at cell resolution, the specific requirements for such tools are still hard to define, due to the novelty of the data and lack of other tools in this area. However, from the experiences gained while employing the tool on real data, some limitations were uncovered.

The scalability of the CNG is still limited, especially for datasets that have large components, as it still requires a high-end machine for their computations. The main problem

here is the chosen connected-component labeling approach, which is not parallelized or optimized for RAM consumption. In order to make the mesh generation and metric computations of large datasets feasible for the average PC, the handling of RAM has to be improved across the whole application.

The suggestion of incomplete tubular meshes works well for gaps in a single tube, but by inspecting the generated meshes of nerves, vasculature and tubule it has become evident that there is often a missing part that corresponds to a bifurcation in the structure. The proposed method is not able to suggest the connectivity that happens in a bifurcating tubular structure, since it is only able to take two meshes into account. Furthermore, this contribution is missing an accessible visualization that helps the user define the required input metrics by showing their effect.

Scalability is also an issue for the visualizations in the web application. When it comes to a large number of meshes or nodes, as demonstrated with a cell dataset that has 43 thousand individual meshes, it fails due to RAM limits or only outputs a very low frame rate. More scalable tools like *neuroglancer* [MSa] for 3D rendering or *Cosmograph* [RO] for graph visualizations might offer a solution, but are more restrictive in their integration and capabilities.

Since the web application is a minimal prototype that highlights the possibilities of connectivity visualization and analysis, it is not a comprehensive standalone application. Therefore, it has some limitations when it comes to configurability. Currently, only one data source is supported and the visibility of components can not be changed aside from the interactive mechanisms that move components to the background and highlight them. The color scheme of the connectivity highlighting and the component types is also not customizable through the user interface.

7.3 Future Work

Through the design, development and usage of this work, multiple avenues for future work have opened up. While the CNG is already useable for moderately sized datasets as demonstrated, the RAM consumption can be optimized to increase its applicability. Furthermore, a plugin system for implementations of custom metric calculations as well as clustering algorithms can be integrated. Another extension could be the augmentation of additional data into the network generation process, like the phenotyping data of single cells. Lastly, the groundwork for distributed computing is already implemented and can be extended to allow the computation across specific cluster networks.

The tubular mesh connectivity suggestion strategy can be improved by incorporating bifurcations in the suggestion process and making it into its own standalone tool that helps researchers to assure the quality of their tubular meshes. By offering a dynamic visualization of the suggestion process that changes based on configurable metric parameters, a human-in-the-loop workflow can be established that speeds up manual tracing.

After configuration, the tool sequentially offers suggestions which the user can accept or discard while inspecting it in the visualization.

The connectivity visualization application can be integrated into a larger system like *Vitessce* [KGM+24] that already uses synchronized views. This tool is based on modularity and focuses on single cell experiments. There, it might make more sense to integrate a more complex viewer like *neuroglancer* and views that offer customization options like color schemes are already implemented.

In order to make the hypergraph that shows the clustering of the connectivity graph more scalable, it can be improved by adding collapsible subgroups when expanding a cluster. These subgroups can also offer the ability to collapse based on the number of nodes that are shown, to enable the visualization of even larger datasets. Since this requires a nested force simulation, the current framework needs to be extended with a custom implementation.

The visualization application can be extended to support multiple data sources. This allows the comparisons of different datasets or the comparison of different connectivity network generation parameters on the same dataset. In case of the network data being augmented with additional data, it could also be compared in this fashion.

Finally, the clarity of the custom radial ego graph can be improved by positioning the nodes so that the edge distances are held minimal. This requires the development of an optimized algorithm that can be executed in real time, since the ego graph needs to be created dynamically depending on the user's input.

The tools and visualizations developed in the course of this thesis were intended as a stepping stone for researchers in the field of cell and FTU connectivity and aim to provide a basis for the initial analysis of novel datasets while contributing to the requirement analysis of future tools.

Overview of Generative AI Tools Used

LanguageTool [NM] was used for grammar and spell checking.

List of Figures

List of Tables

List of Listings

Glossary

biobank an organized collection of human biological material and associated information stored for one or more research purposes [KCT08]. 9

connectome A comprehensive list of neuronal connections in a defined region often referred to in regard to the brain.. 16

fluorescence is light that is emitted by certain substances that have absorbed electromagnetic radiation after they were exposed to ultraviolet radiation [LL99]. 8, 9, 11, 97, 107, 110

immunofluorescence is a technique that utilizes light microscopes and fluorescence to detect biomolecules in a tissue [OC13]. 9, 12, 110

intraepithelial neoplasia are abnormal or excessively growing cells in the epithelial tissue often found on the inner surfaces of body cavities blood vessels or ducts. 8, 69, 110

nephrectomy is the medical term for the operative removal of a human kidney. It is employed to treat multiple kidney diseases or to gain a kidney transplant [Rob63]. 11

pancreatic parenchyma is the functional tissue specifically found in the pancreas. 12

tractogram a 3D model of nerve connections that is created from diffusion magnetic resonance imaging (MRI) data. 17

Acronyms

2D two-dimensional. 8, 14, 15, 17, 26, 35, 39, 97

3D three-dimensional. 1–3, 8–16, 18, 20, 23–28, 30, 32, 33, 35, 39, 44, 47–53, 58, 59, 66, 69, 74, 78–80, 83–86, 89, 91–93, 97, 99–101, 107

cc3d connected-components-3d. 59, 89

CCL connected component labeling algorithm. 26

CLI command-line interface. 55, 57, 65

CNG Connectivity Network Generator. 55, 57, 64, 65, 67, 74, 79, 89, 91–93

CODA quantitative 3D reconstruction of large tissues at cellular resolution. 13

CPU central processing unit. 28, 68

CTL Contour Tracing Labeling. 27

DNA deoxyribonucleic acid. 13

FTU functional tissue unit. 23, 57, 58, 62–65, 71, 94, 105

FTUs functional tissue units. 2, 11, 12, 14, 17, 18, 20, 23–25, 33, 57, 58, 61, 63, 64, 67, 71, 72, 85, 91, 100, 105

GPU graphics processing unit. 48, 66–68, 74

H&E hematoxylin and eosin. 8, 9, 12, 13, 15, 97

HOL Hybrid Object Labeling. 27

HTAN Human Tumor Atlas Network. 1

HTTP Hypertext Transfer Protocol. 55, 57, 64, 65, 92

HuBMAP Human Biomolecular Atlas Program. 1, 9, 11, 13, 97

IBCL Improved Block based Connected-component Labeling. 27

ICTCL Improved Configuration-Transition-based Connected-component Labeling. 27, 59

ID identifier. 38, 39, 50–52, 54, 98, 99

IRCL Improved Run-based Connected-component Labeling. 27

JSON JavaScript Object Notation. 55, 92

KPMP Kidney Precision Medicine Program. 1

LOD level of detail. 30, 31, 48, 49, 74, 89, 99

LSCM laser-scanning confocal microscopy. 9, 10, 12, 13

LSFM light sheet fluorescence microscopy. 8, 9, 11, 13

MRI magnetic resonance imaging. 107

NIH National Institutes of Health. 1

PanIN pancreatic intraepithelial neoplasia. 8, 12, 13, 15, 20, 67–73, 97, 99, 100

PC personal computer. 28, 34, 48, 66, 67, 91, 93

PCA principal component analysis. 35, 36, 61, 98

PLY polygon file format. 66

RAM random-access memory. 28, 48, 58, 59, 61, 67, 68, 74, 89, 92, 93

RNA ribonucleic acid. 13

SASS Syntactically Awesome Style Sheets. 65

SSD Solid-State Drive. 67, 68

SVG Scalable Vector Graphics. 56, 65, 66

t-CyCIF tissue based cyclic immunofluorescence. 9, 12, 13

WebGL Web Graphics Library. 48, 66, 74

Bibliography

[AABS+14] Ali K Al-Awami, Johanna Beyer, Hendrik Strobelt, Narayanan Kasthuri, Jeff W Lichtman, Hanspeter Pfister, and Markus Hadwiger. Neurolines: a subway map metaphor for visualizing nanoscale neuronal connectivity. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2369–2378, 2014.

[Ack] David Ackerman. Multiresolution mesh creator. `https://github.com/janelia-cellmap/multiresolution-mesh-creator`. [accessed on 20/12/2024].

[ADS+22] Matthias Arzt, Joran Deschamps, Christopher Schmied, Tobias Pietzsch, Deborah Schmidt, Pavel Tomancak, Robert Haase, and Florian Jug. Labkit: labeling and segmentation toolkit for big image data. *Frontiers in computer science*, 4:777728, 2022.

[AHB+16] Benedict Anchang, Tom DP Hart, Sean C Bendall, Peng Qiu, Zach Bjornson, Michael Linderman, Garry P Nolan, and Sylvia K Plevritis. Visualization and cellular hierarchy inference of single-cell data using spade. *Nature protocols*, 11(7):1264–1279, 2016.

[AST21] Ali Abdollahzadeh, Alejandra Sierra, and Jussi Tohka. Cylindrical shape decomposition for 3d segmentation of tubular objects. *IEEE Access*, 9:23979–23995, 2021.

[ATC+08] Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. Skeleton extraction by mesh contraction. *ACM transactions on graphics (TOG)*, 27(3):1–10, 2008.

[Att10] Marco Attene. A lightweight approach to repairing digitized polygon meshes. *The visual computer*, 26:1393–1406, 2010.

[Bc] Ian Bicking and community. pip. `https://pip.pypa.io/en/stable/`. [accessed on 10/01/2025].

[BDDD+11] John F Bertram, Rebecca N Douglas-Denton, Boucar Diouf, Michael D Hughson, and Wendy E Hoy. Human nephron number: implications for health and disease. *Pediatric nephrology*, 26:1529–1533, 2011.

[BDH+] Mike Bostock, Jason Davies, Jeffrey Heer, Vadim Ogievetsky, and community. d3js. https://d3js.org/. [accessed on 09/01/2025].

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[Bez74] Pierre Bezier. Mathematical and practical possibilities of unisurf. In *Computer aided geometric design*, pages 127–152. Elsevier, 1974.

[Bio] Translucence Biosystems. Stitchy™. https://www.translucencebio.com/stitchy. [accessed on 14/11/2024].

[BO05] Jean-Daniel Boissonnat and Steve Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, 2005.

[BOS+18] Peter O Bayguinov, Dennis M Oakley, Chien-Cheng Shih, Daniel J Geanon, Matthew S Joens, and James AJ Fitzpatrick. Modern laser scanning confocal microscopy. *Current protocols in cytometry*, 85(1):e39, 2018.

[BS09] Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature reviews neuroscience*, 10(3):186–198, 2009.

[Car99] Stuart K Card. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[Cca] Ricardo Cabello and community. three.js. https://threejs.org/. [accessed on 10/01/2025].

[Ccb] David Cournapeau and community. scikit-learn. https://scikit-learn.org/stable/. [accessed on 10/01/2025].

[CCL04] Fu Chang, Chun-Jen Chen, and Chi-Jen Lu. A linear-time component-labeling algorithm using contour tracing technique. *computer vision and image understanding*, 93(2):206–220, 2004.

[CCY15] Wan-Yu Chang, Chung-Cheng Chiu, and Jia-Horng Yang. Block-based connected-component labeling algorithm using binary decision trees. *Sensors*, 15(9):23763–23787, 2015.

[Col07] Tony J Collins. Imagej for microscopy. *Biotechniques*, 43(S1):S25–S30, 2007.

[CWK+13] Kwanghun Chung, Jenelle Wallace, Sung-Yon Kim, Sandhiya Kalyanasundaram, Aaron S Andalman, Thomas J Davidson, Julie J Mirzabekov, Kelly A Zalocusky, Joanna Mattis, Aleksandra K Denisin, et al. Structural and molecular interrogation of intact biological systems. *Nature*, 497(7449):332–337, 2013.

[CZP+18] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[dBAA+21] Ian H de Boer, Charles E Alpers, Evren U Azeloglu, Ulysses GJ Balis, Jonathan M Barasch, Laura Barisoni, Kristina N Blank, Andrew S Bomback, Keith Brown, Pierre C Dagher, et al. Rationale and design of the kidney precision medicine project. *Kidney international*, 99(3):498–510, 2021.

[DK91] Akio Doi and Akio Koide. An efficient method of triangulating equivalued surfaces by using tetrahedral cells. *IEICE TRANSACTIONS on Information and Systems*, 74(1):214–224, 1991.

[Dun73] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[EBJ+12] Ali Ertürk, Klaus Becker, Nina Jährling, Christoph P Mauch, Caroline D Hojer, Jackson G Egen, Farida Hellal, Frank Bradke, Morgan Sheng, and Hans-Ulrich Dodt. Three-dimensional imaging of solvent-cleared organs using 3disco. *Nature protocols*, 7(11):1983–1995, 2012.

[EBKdN] Clark Evans, Oren Ben-Kiki, and Ingy döt Net. Yaml. `https://yaml.org/`. [accessed on 10/01/2025].

[EPF+24] Henry Ehlers, Daniel Pahr, Velitchko Filipov, Hsiang-Yun Wu, and Renata G Raidou. Me! me! me! me! a study and comparison of ego network representations. *Computers & Graphics*, 125:104123, 2024.

[F+02] Linton C Freeman et al. Centrality in social networks: Conceptual clarification. *Social network: critical concepts in sociology. Londres: Routledge*, 1:238–263, 2002.

[FR91] Thomas MJ Fruchterman and Edward M Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[GAB+05] Ilya G Goldberg, Chris Allan, Jean-Marie Burel, Doug Creager, Andrea Falconi, Harry Hochheiser, Josiah Johnston, Jeff Mellen, Peter K Sorger, and Jason R Swedlow. The open microscopy environment (ome) data model and xml file: open tools for informatics and quantitative analysis in biological imaging. *Genome biology*, 6:1–13, 2005.

[GH97] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997.

[GJM+23] Soumya Ghose, Yingnan Ju, Elizabeth McDonough, Jonhan Ho, Arivarasan Karunamurthy, Chrystal Chadwick, Sanghee Cho, Rachel Rose, Alex Corwin, Christine Surrette, et al. 3d reconstruction of skin and spatial mapping of immune cell density, vascular distance and effects of sun exposure and aging. *Communications Biology*, 6(1):718, 2023.

[Gro] Khronos WebGL Working Group. Web graphics library (webgl). `https://www.khronos.org/webgl/`. [accessed on 02/01/2025].

[Gó] John Alexis Guerra Gómez. d3-force-boundary. `https://github.com/john-guerra/d3-force-boundary`. [accessed on 10/01/2025].

[HBFM08] Ralph H Hruban, Kieran Brune, Noriyoshi Fukushima, and Anirban Maitra. Pancreatic intraepithelial neoplasia. In *Pancreatic Cancer*, pages 41–51. Springer, 2008.

[HCS10] Lifeng He, Yuyan Chao, and Kenji Suzuki. A run-based one-and-a-half-scan connected-component labeling algorithm. *International Journal of Pattern Recognition and Artificial Intelligence*, 24(04):557–579, 2010.

[HM] Anders Hejlsberg and Microsoft. Typescript. `https://www.typescriptlang.org/`. [accessed on 10/01/2025].

[HOP96] H HOPPE. Progressive meshes, computer graphics. In *Proc. SIGGRAPH 96*, pages 99–108, 1996.

[HQN05] Qingmao Hu, Guoyu Qian, and Wieslaw L Nowinski. Fast connected-component labelling in three-dimensional binary images based on iterative recursion. *Computer Vision and Image Understanding*, 99(3):414–434, 2005.

[HRG+17] Lifeng He, Xiwei Ren, Qihang Gao, Xiao Zhao, Bin Yao, and Yuyan Chao. The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition*, 70:25–43, 2017.

[HTH+21] Y He, X Tang, J Huang, J Ren, H Zhou, K Chen, A Liu, H Shi, Z Lin, Q Li, et al. Clustermap for multi-scale clustering analysis of spatial gene expression. nat. commun. 12, 5909, 2021.

[HuB19] HuBMAP Consortium. The human body at cellular resolution: the nih human biomolecular atlas program. *Nature*, 574(7777):187–192, 2019.

[Inc] Kitware Inc. Vtk. `https://vtk.org/`. [accessed on 10/01/2025].

[JDL11] Radu Jianu, Cagatay Demiralp, and David H Laidlaw. Exploring brain connectivity with two-dimensional neural maps. *IEEE transactions on visualization and computer graphics*, 18(6):978–987, 2011.

[JHU] Institute for Nanobiotechnology John Hopkins University. Institute for nanobiotechnology website. https://inbt.jhu.edu/. [accessed on 16/11/2024].

[JKW+21] Jared Jessup, Robert Krueger, Simon Warchol, John Hoffer, Jeremy Muhlich, Cecily C Ritch, Giorgio Gaglia, Shannon Coy, Yu-An Chen, Jia-Ren Lin, et al. Scope2screen: Focus+ context techniques for pathology tumor assessment in multivariate image data. *IEEE transactions on visualization and computer graphics*, 28(1):259–269, 2021.

[JLSW02] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, 2002.

[JPS+23] Sanjay Jain, Liming Pei, Jeffrey M Spraggins, Michael Angelo, James P Carson, Nils Gehlenborg, Fiona Ginty, Joana P Gonçalves, James S Hagood, John W Hickey, et al. Advances and prospects for the human biomolecular atlas program (hubmap). *Nature cell biology*, 25(8):1089–1100, 2023.

[KBG+22] Ashley L Kiemen, Alicia M Braxton, Mia P Grahn, Kyu Sang Han, Jaanvi Mahesh Babu, Rebecca Reichel, Ann C Jiang, Bridgette Kim, Jocelyn Hsu, Falone Amoa, et al. Coda: quantitative 3d reconstruction of large tissues at cellular resolution. *Nature Methods*, 19(11):1490–1499, 2022.

[Kca] Alex Kaszynski and community. fast-simplification. https://pyvista.github.io/fast-simplification/. [accessed on 10/01/2025].

[Kcb] Tobias Knoppers and community. webpack. https://webpack.js.org/. [accessed on 10/01/2025].

[KCÖ+22] Laura Kuett, Raúl Catena, Alaz Özcan, Alex Plüss, Peter Schraml, Holger Moch, Natalie de Souza, and Bernd Bodenmiller. Three-dimensional imaging mass cytometry for highly multiplexed molecular and cellular mapping of tissues and the tumor microenvironment. *Nature Cancer*, 3(1):122–133, 2022.

[KCT08] Francine Kauffmann and Anne Cambon-Thomsen. Tracing biological collections: between books and clinical trials. *Jama*, 299(19):2316–2318, 2008.

[KDB+23] Ashley L Kiemen, Alexander Ioannis Damanakis, Alicia M Braxton, Jin He, Daniel Laheru, Elliot K Fishman, Patrick Chames, Cristina Almagro Pérez, Pei-Hsun Wu, Denis Wirtz, et al. Tissue clearing and 3d reconstruction of digitized, serially sectioned slides provide novel insights into pancreatic cancer. *Med*, 4(2):75–91, 2023.

[KEC06] René Keller, Claudia M Eckert, and P John Clarkson. Matrices or node-link diagrams: which visual representation is better for visualising connectivity models? *Information Visualization*, 5(1):62–76, 2006.

[KGM+24] Mark S Keller, Ilan Gold, Chuck McCallum, Trevor Manz, Peter V Kharchenko, and Nils Gehlenborg. Vitessce: integrative visualization of multimodal and spatially resolved single-cell data. *Nature Methods*, pages 1–5, 2024.

[KKG11] Paul AJ Kolarsick, Maria Ann Kolarsick, and Carolyn Goodwin. Anatomy and physiology of the skin. *Journal of the Dermatology Nurses' Association*, 3(4):203–213, 2011.

[KLS+17] Ethan Kerzner, Alexander Lex, Crystal Lynn Sigulinsky, Timothy Urness, Bryan W Jones, Robert E Marc, and Miriah Meyer. Graffinity: Visualizing connectivity in large graphs. In *Computer Graphics Forum*, volume 36, pages 251–260. Wiley Online Library, 2017.

[KLV+18] Kimmo Kartasalo, Leena Latonen, Jorma Vihinen, Tapio Visakorpi, Matti Nykter, and Pekka Ruusuvuori. Comparative analysis of tissue reconstruction algorithms for 3d histology. *Bioinformatics*, 34(17):3013–3021, 2018.

[Lab] Sanjay Jain Lab. Sanjay jain lab website. `https://www.sanjayjainlab.org/`. [accessed on 16/11/2024].

[LC98] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. Association for Computing Machinery, 1998.

[LFSCS16] Jia-Ren Lin, Mohammad Fallahi-Sichani, Jia-Yun Chen, and Peter K Sorger. Cyclic immunofluorescence (cycif), a highly multiplexed method for single-cell imaging. *Current protocols in chemical biology*, 8(4):251–264, 2016.

[Lib] LibreTexts libraries. Gross anatomy of the kidney. `https://bio.libretexts.org/Courses/Lumen_Learning/Anatomy_and_Physiology_II_%28Lumen%29/11%3A_Module_9-_The_Urinary_System/11.04%3A_Gross_Anatomy_of_the_Kidney`. [accessed on 14/11/2024].

[LL99] Joseph R Lakowicz and Joseph R Lakowicz. Introduction to fluorescence. *Principles of fluorescence spectroscopy*, pages 1–23, 1999.

[Llo82] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[Lon14] Daniel S Longnecker. Anatomy and histology of the pancreas (version 1.0). *Pancreapedia: The Exocrine Pancreas Knowledge Base*, 2014.

[LPK+13] Alexander Lex, Christian Partl, Denis Kalkofen, Marc Streit, Samuel Gratzl, Anne Mai Wassermann, Dieter Schmalstieg, and Hanspeter Pfister. Entourage: Visualizing relationships between biological pathways using contextual subsets. *IEEE transactions on visualization and computer graphics*, 19(12):2536–2545, 2013.

[LT98] Peter Lindstrom and Greg Turk. Fast and memory efficient polygonal simplification. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pages 279–286. IEEE, 1998.

[Mah19a] Vishy Mahadevan. Anatomy of the kidney and ureter. *Surgery (Oxford)*, 37(7):359–364, 2019.

[Mah19b] Vishy Mahadevan. Anatomy of the pancreas and spleen. *Surgery (Oxford)*, 37(6):297–301, 2019.

[Mat] Mathworks Inc. Matlab. `https://mathworks.com/products/matlab.html`. [accessed on 14/11/2024].

[MBc] Charlie Marsh, Zanie Blue, and community. uv. `https://github.com/astral-sh/uv`. [accessed on 10/01/2025].

[Mc] Meta and community. React. `https://react.dev/`. [accessed on 09/01/2025].

[MGP+22] Trevor Manz, Ilan Gold, Nathan Heath Patterson, Chuck McCallum, Mark S Keller, Bruce W Herr, Katy Börner, Jeffrey M Spraggins, and Nils Gehlenborg. Viv: multiscale visualization of high-resolution multiplexed bioimaging data on the web. *Nature methods*, 19(5):515–516, 2022.

[MH07] Julio Martín-Herrero. Hybrid object labelling in digital images. *Machine Vision and Applications*, 18(1):1–15, 2007.

[Mica] Carl Zeiss Microscopy. Zeiss light sheet 7. `https://www.zeiss.com/microscopy/de/produkte/lichtmikroskope/lichtblattmikroskope/lightsheet-7.html`. [accessed on 14/11/2024].

[Micb] Carl Zeiss Microscopy. Zeiss lsm 980 with airyscan 2. `https://www.zeiss.com/microscopy/de/produkte/lichtmikroskope/konfokale-mikroskope/lsm-980-mit-airyscan-2.html`. [accessed on 14/11/2024].

[Micc] Carl Zeiss Microscopy. Zen blue 3.3. https://www.zeiss.com/microscopy/de/produkte/software/zeiss-zen.html. [accessed on 14/11/2024].

[MPB+19] G Elisabeta Marai, Bruno Pinaud, Katja Bühler, Alexander Lex, and John H Morris. Ten simple rules to create biological network figures for communication, 2019.

[MSa] Jeremy Maitin-Shepard. Neuroglancer. https://github.com/google/neuroglancer. [accessed on 20/12/2024].

[MSb] Jeremy Maitin-Shepard. Neuroglancer precomputed format. https://github.com/google/neuroglancer/tree/master/src/datasource/precomputed. [accessed on 20/12/2024].

[MSOI+02] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[MZS+24] Liam McLaughlin, Bo Zhang, Siddharth Sharma, Amanda L Knoten, Madhurima Kaushal, Jeffrey M Purkerson, Heidy Huyck, Gloria S Pryhuber, Joseph P Gaut, and Sanjay Jain. Three dimensional multiscalar neurovascular nephron connectivity map of the human kidney across the lifespan. *bioRxiv*, pages 2024–07, 2024.

[Nat] National Cancer Institute. Human tumor atlas network. https://data.humantumoratlas.org/. [accessed on 14/11/2024].

[NB21] Carl J Nelson and Stephen Bonner. Neuronal graphs: A graph theory primer for microscopic, functional networks of neurons recorded by calcium imaging. *Frontiers in Neural Circuits*, 15:662882, 2021.

[NM] Daniel Naber and Marcin Miłkowski. Languagetool. https://languagetool.org/de. [accessed on 10/01/2025].

[NMSL19] Carolina Nobre, Miriah Meyer, Marc Streit, and Alexander Lex. The state of the art in visualizing multivariate networks. In *Computer Graphics Forum*, volume 38, pages 807–832. Wiley Online Library, 2019.

[NW] Chris Eppstein Natalie Weizenbaum. Syntactically awesome style sheets. https://sass-lang.com/. [accessed on 10/01/2025].

[NWHL20] Carolina Nobre, Dylan Wootton, Lane Harrison, and Alexander Lex. Evaluating multivariate network visualization techniques using a validated design and crowdsourcing approach. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–12, 2020.

[Oc] Travis Oliphant and community. numpy. https://numpy.org/. [accessed on 10/01/2025].

[OC13] Ian D Odell and Deborah Cook. Immunofluorescence techniques. *The Journal of investigative dermatology*, 133(1):e4, 2013.

[OTV21] John W Osborn, Roman Tyshynsky, and Lucy Vulchanova. Function of renal nerves in kidney physiology and pathophysiology. *Annual review of physiology*, 83(1):429–450, 2021.

[PB13] Bernhard Preim and Charl P Botha. *Visual computing for medicine: theory, algorithms, and applications*. Newnes, 2013.

[Pc] Fernando Pérez and community. Jupyter. https://jupyter.org/. [accessed on 10/01/2025].

[Pea01] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[PGS+16] Christian Partl, Samuel Gratzl, Marc Streit, Anne Mai Wassermann, Hanspeter Pfister, Dieter Schmalstieg, and Alexander Lex. Pathfinder: Visual analysis of paths in graphs. In *Computer Graphics Forum*, volume 35, pages 71–80. Wiley Online Library, 2016.

[PH17] Rory M Power and Jan Huisken. A guide to light-sheet fluorescence microscopy for multiscale imaging. *Nature methods*, 14(4):360–373, 2017.

[Pli22] Guillaume Plique. Graphology, a robust and multipurpose graph object for javascript. *Zenodo. https://doi. org/10.5281/zenodo*, 5681257, 2022.

[PNAM24] Fathima Jubina Pathari, Yvonne Nielsen, Liv Ingrid Andersen, and Georgios Marentakis. Dark vs. light mode on smartphones: Effects on eye fatigue. In *ACHI 2024, The Seventeenth International Conference on Advances in Computer-Human Interactions*, pages 150–154, 2024.

[Pra23] Praveen Krishnamoorthy, Bo Zhang and Sanjay Jain. Three-dimensional reconstruction of glomeruli from light sheet microscopy. https://hubmapconsortium.org/image-of-the-week/, 2023. [accessed on 13/11/2024].

[PRL+10] Hanchuan Peng, Zongcai Ruan, Fuhui Long, Julie H Simpson, and Eugene W Myers. V3d enables real-time 3d visualization and quantitative analysis of large-scale biological image data sets. *Nature biotechnology*, 28(4):348–353, 2010.

[PS22] Marius Pachitariu and Carsen Stringer. Cellpose 2.0: how to train your own model. *Nature methods*, 19(12):1634–1641, 2022.

[PSC+19] Young-Gyun Park, Chang Ho Sohn, Ritchie Chen, Margaret McCue, Dae Hee Yun, Gabrielle T Drummond, Taeyun Ku, Nicholas B Evans, Hayeon Caitlyn Oak, Wendy Trieu, et al. Protection of tissue physicochemical properties using polyfunctional crosslinkers. *Nature biotechnology*, 37(1):73–83, 2019.

[QSB+11] Peng Qiu, Erin F Simonds, Sean C Bendall, Kenneth D Gibbs Jr, Robert V Bruggner, Michael D Linderman, Karen Sachs, Garry P Nolan, and Sylvia K Plevritis. Extracting a cellular hierarchy from high-dimensional cytometry data with spade. *Nature biotechnology*, 29(10):886–891, 2011.

[RO] Nikita Rokotyan and Stukova Olya. Cosmograph. `https://cosmograph.app/`. [accessed on 10/01/2025].

[Rob63] Charles J Robson. Radical nephrectomy for renal cell carcinoma. *The Journal of urology*, 89(1):37–42, 1963.

[Roc] Matthew Rocklin. Dask. `https://www.dask.org/`. [accessed on 29/11/2024].

[Ros76] Azriel Rosenfeld. *Digital picture processing*. Academic press, 1976.

[RRRO+20] Orit Rozenblatt-Rosen, Aviv Regev, Philipp Oberdoerffer, Tal Nawy, Anna Hupalowska, Jennifer E Rood, Orr Ashenberg, Ethan Cerami, Robert J Coffey, Emek Demir, et al. The human tumor atlas network: charting tumor transitions across space and time at single-cell resolution. *Cell*, 181(2):236–249, 2020.

[Sca] Nico Schlömer and community. meshio. `https://github.com/nschloe/meshio`. [accessed on 10/01/2025].

[Scb] Chad Smith and community. pip. `https://github.com/pypa/pipx`. [accessed on 10/01/2025].

[Scc] Bane Sullivan and community. pyvista. `https://docs.pyvista.org/`. [accessed on 10/01/2025].

[Sco94] Roberto Scopigno. A modified look-up table for implicit disambiguation of marching cubes. *The visual computer*, 10:353–355, 1994.

[Sil] William Silversmith. connected-components-3d. `https://github.com/seung-lab/connected-components-3d/tree/master`. [accessed on 10/01/2025].

[SP24] Carsen Stringer and Marius Pachitariu. Cellpose3: one-click image restoration for improved cellular segmentation. *bioRxiv*, pages 2024–02, 2024.

[SPL24] Roberto M. Soriano, Dana Penfold, and Stephen W. Leslie. Anatomy, Abdomen and Pelvis: Kidneys. In *StatPearls*. StatPearls Publishing, Treasure Island (FL), 2024.

[SSKB14] Bahador Saket, Paolo Simonetto, Stephen Kobourov, and Katy Börner. Node, node-link, and node-link-group diagrams: An evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2231–2240, 2014.

[STAHAR18] Anas M Saad, Tarek Turk, Muneer J Al-Husseini, and Omar Abdel-Rahman. Trends in pancreatic adenocarcinoma incidence and mortality in the united states in the last four decades; a seer-based study. *BMC cancer*, 18:1–11, 2018.

[STP+14] Etsuo A Susaki, Kazuki Tainaka, Dimitri Perrin, Fumiaki Kishino, Takehiro Tawara, Tomonobu M Watanabe, Chihiro Yokoyama, Hirotaka Onoe, Megumi Eguchi, Shun Yamaguchi, et al. Whole-brain imaging with single-cell resolution using chemical cocktails and computational analysis. *Cell*, 157(3):726–739, 2014.

[SVAB+18] Dirk Schadendorf, Alexander CJ Van Akkooi, Carola Berking, Klaus G Griewank, Ralf Gutzmer, Axel Hauschild, Andreas Stang, Alexander Roesch, and Selma Ugurel. Melanoma. *The Lancet*, 392(10151):971–984, 2018.

[SWMP21] Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. Cellpose: a generalist algorithm for cellular segmentation. *Nature methods*, 18(1):100–106, 2021.

[Sza15] Fred Szabo. *The linear algebra survival guide: illustrated with Mathematica*. Academic Press, 2015.

[TKS+14] Kazuki Tainaka, Shimpei I Kubota, Takeru Q Suyama, Etsuo A Susaki, Dimitri Perrin, Maki Ukai-Tadenuma, Hideki Ukai, and Hiroki R Ueda. Whole-body imaging with single-cell resolution by tissue decolorization. *Cell*, 159(4):911–924, 2014.

[Tur94] Greg Turk. The ply polygon file format. *Recuperado de*, 1994.

[TWC+23] Jakob Troidl, Simon Warchol, Jinhan Choi, Jordan Matelsky, Nagaraju Dhanyasi, Xueying Wang, Brock Wester, Donglai Wei, Jeff W Lichtman, Hanspeter Pfister, et al. Vimo-visual analysis of neuronal connectivity motifs. *IEEE transactions on visualization and computer graphics*, 2023.

[TZCO09] Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. Curve skeleton extraction from incomplete point cloud. In *ACM SIGGRAPH 2009 papers*, pages 1–9. Association for Computing Machinery, 2009.

[VCI20] Ivan Viola, Min Chen, and Tobias Isenberg. Visual abstraction. *Foundations of data visualization*, pages 15–37, 2020.

[vdWc] Stéfan van der Walt and community. scikit-image. `https://scikit-image.org/`. [accessed on 10/01/2025].

[VPR21] Alba Vieites-Prado and Nicolas Renier. Tissue clearing and 3d imaging in developmental biology. *Development*, 148(18):dev199369, 2021.

[vRC] Raoul van Rueschen and Ricardo Cabello. Post processing three.js. `https://github.com/pmndrs/postprocessing`. [accessed on 10/01/2025].

[W+01] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[WHC+12] Pei-Hsun Wu, Christopher M Hale, Wei-Chiang Chen, Jerry SH Lee, Yiider Tseng, and Denis Wirtz. High-throughput ballistic injection nanorheology to measure cell mechanics. *Nature protocols*, 7(1):155–170, 2012.

[WKN+22] Simon Warchol, Robert Krueger, Ajit Johnson Nirmal, Giorgio Gaglia, Jared Jessup, Cecily C Ritch, John Hoffer, Jeremy Muhlich, Megan L Burger, Tyler Jacks, et al. Visinity: Visual spatial neighborhood analysis for multiplexed tissue imaging data. *IEEE transactions on visualization and computer graphics*, 29(1):106–116, 2022.

[WOS09] Kesheng Wu, Ekow Otoo, and Kenji Suzuki. Optimizing two-pass connected-component labeling algorithms. *Pattern Analysis and Applications*, 12:117–135, 2009.

[WWL+18] Mingqiang Wei, Qiong Wang, Yichen Li, Wai-Man Pang, Luming Liang, Jun Wang, Kelvin Kian Loong Wong, Derek Abbott, Jing Qin, and Jianhuang Wu. Centerline extraction of vasculature mesh. *IEEE access*, 6:10257–10268, 2018.

[YNZ+24] Clarence Yapp, Ajit J Nirmal, Felix Zhou, Zoltan Maliga, Juliann B Tefft, Paula Montero Llopis, George F Murphy, Christine G Lian, Gaudenz Danuser, Sandro Santagata, et al. Multiplexed 3d analysis of immune states and niches in human tissue. *bioRxiv*, 2024.

[YSD+19] Chun-Hung Yeh, Robert E Smith, Thijs Dhollander, Fernando Calamante, and Alan Connelly. Connectomes from streamlines tractography: Assigning streamlines to brain parcellations is not trivial but highly consequential. *Neuroimage*, 199:160–171, 2019.

[YZHSO16] Bahia Yahya-Zoubir, Latifa Hamami, Llies Saadaoui, and Rafik Ouared. Automatic 3d mesh-based centerline extraction from a tubular geometry form. *Information Technology and Control*, 45(2):156–163, 2016.

[ZHYC15] Xiao Zhao, Lifeng He, Bin Yao, and Yuyan Chao. A new connected-component labeling algorithm. *IEICE TRANSACTIONS on Information and Systems*, 98(11):2013–2016, 2015.

[Zim11] Carl Zimmer. 100 trillion connections. *Scientific American*, 304(1):58–61, 2011.

[ZXYQ13] Hong Zhou, Panpan Xu, Xiaoru Yuan, and Huamin Qu. Edge bundling in information visualization. *Tsinghua Science and Technology*, 18(2):145–156, 2013.