Institute of Biomechanics
Center of Biomedical Engineering
Kronesgasse 5-I
8010 Graz, Austria

Diploma Thesis

Determination of Microstructural Parameters from TEM-Images Featuring Collagen Fibrils and Proteoglycans

to achieve the degree of
Dipl-Ing.

Author: Emanuel Sandner
Supervisor: Dipl.-Ing. Dr.techn. Andreas J. Schriefl

Head of Institute: Professor Gerhard A. Holzapfel

March 6, 2013

Contents

List of Figures

Abstract

A common type of cardiovascular disease is atherosclerosis, which can cause a stenosis (narrowing) in tubular structures, such as arteries. One typical treatment is the so-called angioplasty, where a balloon is inserted into the narrow area, gets dilated and thus widens the artery. However, this widening causes an unnatural overstretch of the arterial wall and thus induces microstructural damage in the vessel. In order to predict the extent of damage inflicted to the tissue due to for example overstretch, micro-mechanical models are being used that describe the macroscopic mechanical tissue response. These models require proper parameters to characterize the individual tissue components.

In this thesis, a plugin for the program ImageJ has been developed and tested to automatically extract such microstructural parameters for collagen fibrils and proteoglycans from transmission electron microscope images. To test the developed algorithm, porcine and human aortic tissue were used to determine collagen fibril diameters, interfibrillar distances and proteoglycan orientations with respect to the alignment of their associated fibrils. Different settings are at hand to limit the results according to, e.g., the maximum neighbor count, the neighborhood distance or the allowed fibrillar overlap. Various methods for visualizing and evaluating the results were implemented, so that the user can exactly see what was measured.

Since ImageJ is written in Java, the developed plugin can be used on any operating system, as long as the java runtime environment is installed. The functionality can be extended through add-ons, that enable, e.g., additional parameter acquisition methods or an automated statistical analysis of the results.

Kurzfassung

Eine übliche Form von Herz-Kreislauferkrankungen ist die Arteriosklerose, die im weiteren Verlauf zu einer Stenose (Verengung) in röhrenförmigen Strukturen wie z.B. Arterien führen kann. Die typische Behandlung dieser Erkrankung ist die sogenannte Angioplastie, bei der ein Ballon in den verengten Bereich eingeführt und aufgeblasen wird, wodurch die Arterie geweitet wird. Diese Weitung führt allerdings zu einer unnatürlichen Überdehnung der Arterienwand, was zu mikrostrukturellen Schäden in dem Gefäß führt. Um abschätzen zu können, wie viel Schaden am Gewebe durch z.B. eine Überdehnung verursacht wird, werden mikromechanische Modelle verwendet, die das makroskopische Antwortverhalten des Gewebes beschreiben. Diese Modelle benötigen bestimmte Parameter, um die individuellen Gewebekomponenten zu charakterisieren.

Im Zuge dieser Arbeit wurde ein Plugin für das Programm ImageJ geschrieben und getestet, das automatisch bestimmte mikrostrukturelle Parameter von Kollagenfibrillen und Proteoglykanen aus transmissionselektronenmikroskopischen Aufnahmen extrahiert. Um den entwickelten Algorithmus zu testen, wurden Gewebeproben von schweinischen und menschlichen Aorten untersucht und der Kollagenfibrillendurchmesser, der interfibrilläre Abstand und die Proteoglykanorientierung in Bezug zur Ausrichtung der zugehörigen Kollagenfibrille ermittelt. Verschiedene Einstellungen stehen zur Verfügung, um die Ergebnisse nach bestimmten Kriterien einzuschränken wie z.B. die maximale Anzahl der Nachbarn, die Distanz der Nachbarschaft oder die erlaubte Überlappung zweier Fibrillen. Unterschiedliche Methoden zur Visualisierung und Bewertung der Ergebnisse wurden implementiert, damit der Benutzer genau sieht, was gemessen wurde.

Da das Programm ImageJ in Java geschrieben ist, kann das entwickelte Plugin mit jedem Betriebssystem verwendet werden, das die Java-Laufzeitumgebung installiert hat. Die Funktionalität kann beliebig durch add-ons erweitert werden, um z.B. weitere Methoden zur Parametergewinnung zu implementieren oder um eine automatische statistische Analyse der Ergebnisse durchzuführen.

Acknowledgement

I would like to express the deepest appreciation to all people, who supported me constantly during my student days and the development of this thesis.

First, I owe special thanks to my supervisor Dipl.-Ing. Dr.techn. Andreas J. SCHRIEFL for his guidance. Without his support and help this thesis would not have been possible.

Furthermore, I want to thank the head of the Institute of Biomechanics at the Graz University of Technology, Professor Gerhard A. HOLZAPFEL for his support and for giving me the possibility to write this thesis.

I also want to thank Christian VIERTLER and Andrea KOSCHEL for the preparation as well as Dagmar KOLB-LENZ (from the Center for Medical Research (ZMF), Graz) and Andrea KOSCHEL (from the university clinic (LKH), Graz) for providing the electron microscope images of human tissue samples.
Further on, I would like to thank Dr. Nadejda MATSKO and Mag. Toni UUSIMÄKI (both from the Institute for Electron Microscopy and Fine Structure Research (FELMI), Graz) for providing the electron microscope images and the tissue preparation of porcine tissue samples.

I also want to express my sincere gratitude to my family, particularly to my parents Rudolf and Silvia for giving me the opportunity to study in the first place, and for their neverending support and help in all situations and during tough times.
My brother Martin deserves to be mentioned separately, as he encouraged me to not only focus on a particular field of study, but also to widen my horizon for different topics, as he continuously came up with interesting and extraordinary project ideas.

Moreover, I owe special thanks to my lovely fiancé Christiane, who supported and encouraged me throughout the whole study journey, and was my major driving force especially during the last months. I love you and I'm looking forward to the wonderful years ahead in our lives.

Finally, I want to thank all of my friends for the special memories and moments during our time as students in Graz.

1 Introduction

1.1 Motivation

According to statistical data from the world health organization (WHO), cardiovascular diseases are responsible for the highest death rate compared to other causes globally and annually. Back in the year of 2008, it is estimated that 17.3 million people died due to cardiovascular diseases, which corresponds to 30 % of all global deaths. Furthermore, statistical projections show that by the year 2030, the death toll due to cardiovascular diseases will rise to about 25 million people. To approach this problem, the highest priority has to be the prevention of such diseases in the first place, which can be accomplished by reducing specific risk factors such as tobacco use, unhealthy diet, physical inactivity and others (World Health Organization, 2012).

Until then, it is in humanities best interest to apply proper treatments in order to minimize the mortality rate of cardiovascular diseases. A common disease of this kind is for example atherosclerosis, which can cause a stenosis through accumulation of fatty substances, calcium, collagen fibers, cellular waste products and fibrin in tubular structures like arteries (Holzapfel et al., 2000). This accumulation leads to a narrowing of these structures and may ultimately develop into a so-called stricture. Depending on the localisation of this stricture, different severe health injuries or even death may be imminent through heart attacks, smokers leg or strokes. To prevent this from happening, one common therapy is the angioplasty, where an empty balloon is inserted into the narrow area. Afterwards, the balloon gets dilated to widen the structure and therefore reestablishes unhindered blood flow. If necessary, a stent is inserted at the widened area to ensure that the artery won't shrink again after balloon removal. Due to the fact that the atherosclerotic plaque (Holzapfel et al., 2000) is not removed in this process, the whole artery wall in the affected area is superphysiologically strained and thus receives an unnatural amount of stress. This outward bending induces microstructural damage in the vessel wall which can lead to further rupture. In order to better understand the microstructural behaviour of soft biological tissue under such superphysiological conditions, and to predict how much damage is inflicted to the tissue due to, for example, overstretch and to improve common therapeutic procedures, micro-mechanical models are being used to describe the macroscopic mechanical tissue response (see Gasser et al., 2006; Kroon and Holzapfel, 2008; Holzapfel and Ogden, 2010; Balzani et al., 2012). These models require experimentally determined tissue parameters to characterise the individual tissue components. In this regard, a method for the determination of the microstructural model parameters from collagen fibrils and proteoglycans (PG) was developed during this thesis to automatically obtain such parameters.

1.2 Composition of the arterial wall

Figure 1.1 shows the schematic structure of a healthy arterial wall, which consist of three distinct layers, the intima (tunica intima), the media (tunica media) and the adventitia (tunica externa). Each layer is composed of different cells and filaments that fulfil their own specific tasks.

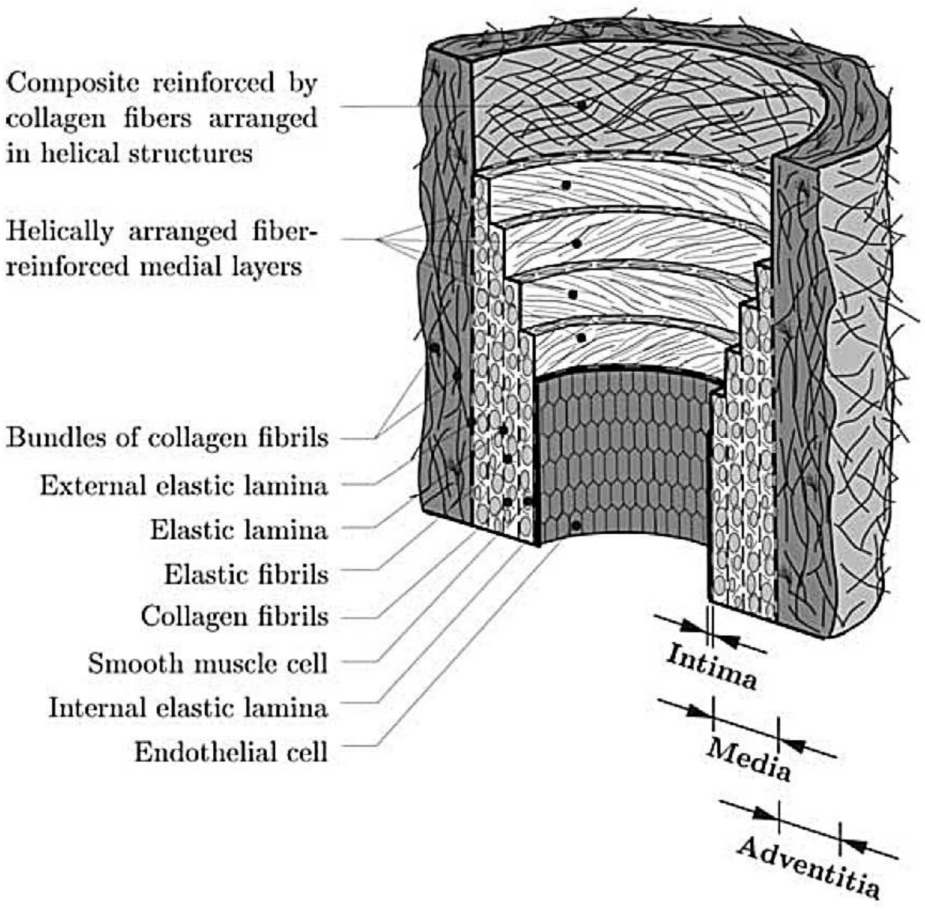

Composite reinforced by collagen fibers arranged in helical structures
Helically arranged fiber-reinforced medial layers
Bundles of collagen fibrils
External elastic lamina
Elastic lamina
Elastic fibrils
Collagen fibrils
Smooth muscle cell
Internal elastic lamina
Endothelial cell
Intima
Media
Adventitia

Figure 1.1: Schematic structure of the arterial wall consisting of three layers: intima, media and adventitia. Additionally, the constituents of each layer are presented. Image was taken from Holzapfel et al. (2000).

1.2.1 Tunica intima

The intima consists of a thin endothelial cell layer that forms a hollow tube on the inside, called the lumen, through which the blood flows. In its early, young and healthy stage the intima bares no major contribution to the biomechanical properties of the arterial wall. In time, the intima thickens and stiffens which may result in a more significant contribution to the mechanical characteristics (Holzapfel et al., 2000).

1.2.2 Tunica media

The media consists of a complex three-dimensional network of smooth muscle cells, elastin and collagen fibrils. Thin layers of a proteoglycan-rich extracellular matrix, the collagen fibrils and the smooth muscle cells can be found between fenestrated sheets (lamellae) formed by the elastin (Wagenseil and Mecham, 2009). The intima and the adventitia are separated by internal and external elastic lamina from the middle layer. The specific structural arrangement of the media's constituents ensure its high strength, resilience and load resistance in the longitudinal and circumferential direction (Holzapfel et al., 2000; Schriefl et al., 2012, 2013).

1.2.3 Tunice adventitia

The main constituents of the outermost arterial layer are fibroblasts and fibrocytes (which produce the proteins collagen and elastin), histological ground substance and thick bundles of collagen fibers. These fibers form helical structures that serve to reinforce the arterial wall and contribute significantly to its stability and strength (Schriefl et al., 2012). To prevent the artery from overstretch and rupture, the collagen fibers straighten and change the adventitia to a stiff tube at high pressures (Holzapfel et al., 2000).

1.3 The extracellular matrix

The extracellular matrix (ECM) is a non cellular component present in the intercellular spaces in all tissue types and organs (Frantz et al., 2010). Its basic structure is defined by a scaffold that is made up from different collagen types. Other constituents like glycoproteins and proteoglycans adhere to the scaffold and interact with surrounding or incorporated cells (Bosman and Stamenkovic, 2003). Elastin is another major structural protein present in the extracellular matrix and is, in conjunction with fibrillin, responsible for the flexibility of many tissues. The communication between cells and the extracellular matrix is established through the crosstalk among the tansmembrane receptors integrins, which establish an integrated link between the outside and the inside of a cell, and the actin cytoskeleton (Boudreau and Jones, 1999; Kim et al., 2011).

1.3.1 Role of the extracellular matrix

The extracellular matrix has multiple functions such as:

- providing the shape and resilience of the tissue
- providing an aqueous environment for nutrient or hormone diffusion between cells and the capillary network
- intercellular communication and chemical reactivity

- binding site for growth factors, enzymes or cells through anchoring junctions

The specific function and constitution of the extracellular matrix varies depending on its associated tissue type (Holzapfel, 2012).

1.3.2 Components of the extracellular matrix

1.3.2.1 Collagen

Collagen is the most abundant structural protein in mammals. It gives mechanical stability, strength and toughness to a range of tissues such as tendons, ligaments, bone, arteries or skin. These tissues have different mechanical requirements, some need to be elastic, store mechanical energy or need to be stiff and tough, hence collagen has a large versatility as a building material (Fratzl, 2008). Collagen occurs in more than 28 known types (Type I to XXVIII), where for example in the cardiovascular system types I, III-VI and VIII are the most common forms. Type I collagen is the most abundant collagen in the human body with a high tensile strength (Holzapfel, 2012). Figure 1.2 shows a model of the organization of collagen molecules in a collagen fibril. Each molecule has a diameter of about $1.4\ nm$ and a contour length of roughly $300\ nm$, that is approximately 4.4 times the d-spacing ($67\ nm$). The collagen molecules are interconnected by covalent cross-links forming collagen fibrils, which further aggregate to collagen fibers. (Holzapfel, 2012). Numerous studies have been conducted in order to determine individual collagen fibril lengths, mainly from chicken tendons but also from fetal rat, bovine and feline ligaments. It was found that the fibril size varies mainly depending on the tissue type and the stage of development. Furthermore, an exact length was only possible to obtain for embryonic fibrils, as mature fibrils were not trackable from their starting point to the end in STEM/TEM-images (Birk and Zycband, 1994; Birk et al., 1995, 1996; Kadler et al., 1996; Birk et al., 1997; Graham et al., 2000; Holmes et al., 2001). Silver et al. (2001) used a computational approach to determine the collagen fibril length from human skin, Alloderm® and processed dermis with the results $54.8\ \mu m$, $63.7\ \mu m$ and $48.8\ \mu m$, respectively. In another study from Provenzano and Vanderby (2006), 7275 fibrils were examined over a total combined tissue length of approximately $2.1\ mm$ without revealing an end in mature tissue.

1.3.2.2 Elastin

The structural protein elastin is composed of a network of long molecules that can sustain very large strains without rupturing. Contrary to collagen fibers, elastin has no apparent hierarchical organization, as the unstructured molecules are linked to each other by distributed covalent cross-links (Holzapfel, 2012).

1.3.2.3 Proteoglycans

Proteoglycans consist of a core protein that is covalently bonded to one or more glycosaminoglycan (GAG) chains. GAGs are unbranched polysaccharide chains, that are

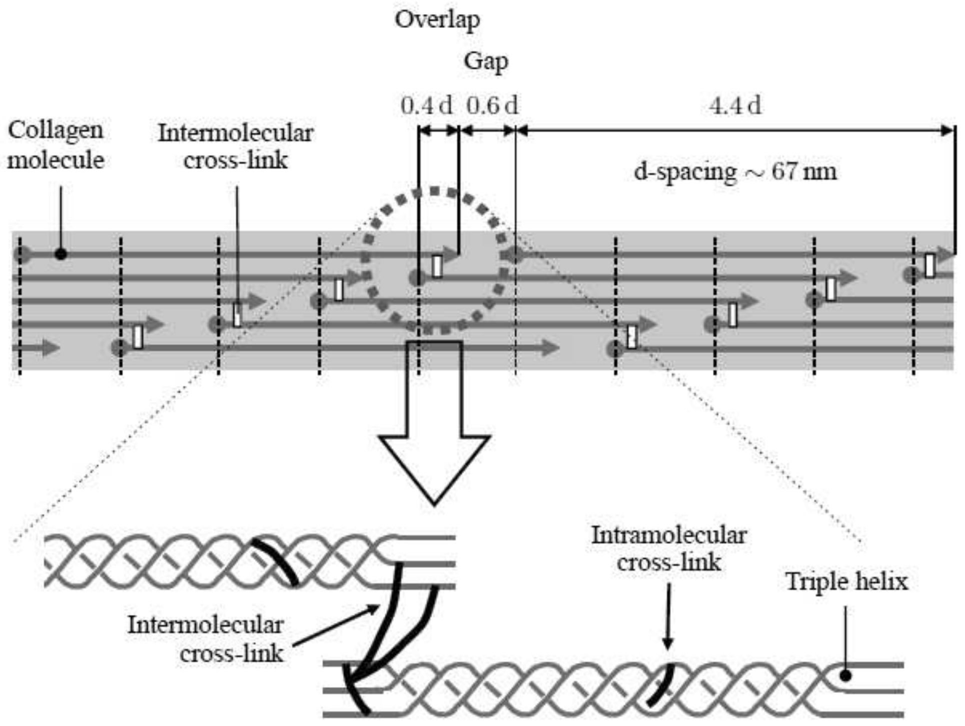

Overlap
Gap
0.4 d
0.6 d
4.4 d
Collagen molecule
Intermolecular cross-link
d-spacing ~ 67 nm
Intramolecular cross-link
Triple helix
Intermolecular cross-link

Figure 1.2: The collagen molecules form in quarter-staggered arrays to collagen fibrils. Due to the staggered conformation, the typical d-spacing, which is a periodic band of ($67\ nm$), becomes apparent. Image was taken from Holzapfel (2012)

made of disaccharide units, which can bind between each other and to fiber like matrix proteins such as collagen. Typical glycosaminoglycans are chondroitin sulfate, dermatan sulfate and heparin sulfate (Alberts et al., 2004). Since GAGs are negatively charged and therefore hydrophilic, they attract and bind a considerable amount of water and thus produce a gel that resists compression of the tissue (Meek, 2008).
It is well known that proteoglycans can form cross-links between adjacent collagen fibrils. One typical proteoglycan is decorin, which has a high affinity binding site for collagen types I-III and VI. It is a small PG with only one glycosaminoglycan chain. The model structure of the core protein is arch shaped such that exactly one collagen molecule fits in the inner concave surface (Weber et al., 1996). In Fig. 1.3 a collagen fibril to proteoglycan interaction model is shown.

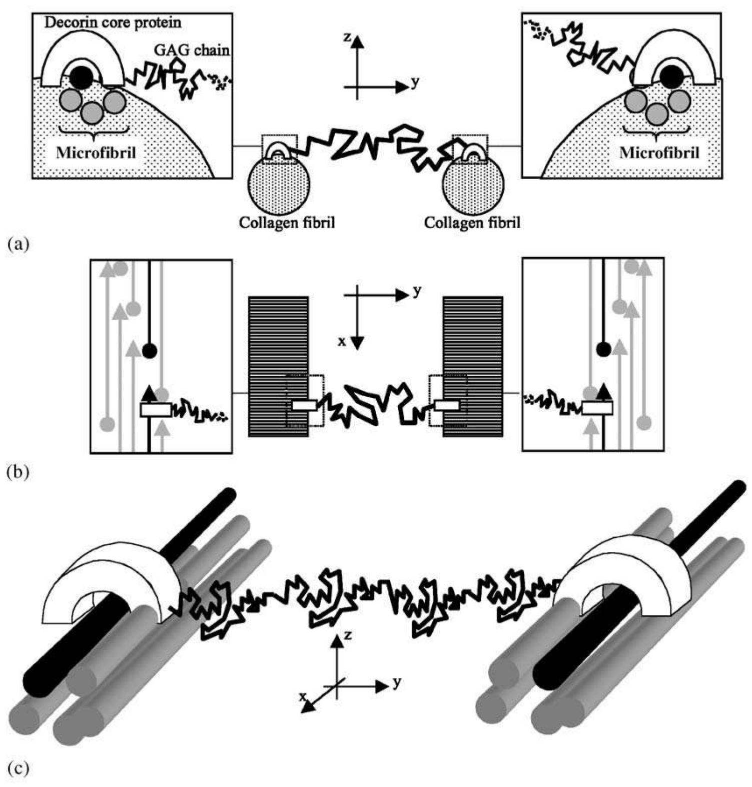


Figure 1.3: Proposed collagen molecule - proteoglycan (decorin) interaction model. Row (a) shows enlarged the connection mediated by decorin core protein from which the GAG chain protrudes away to an adjacent collagen fibril. (b) shows the same configuration but in longitudinal section where each collagen molecules is represented as an arrow; the white rectangle illustrates the core protein. Row (c) show this assembly as a 3-D model. Image was taken from Vesentini et al. (2005).

1.4 Image processing

One of the most important senses for humans is the visualization of their vicinity. This helps for example to orientate in space, detect potential threats or register fine structures and their geometries. To obtain these kind of concious information, the input data (light) hits the retina in the eye bulb where it interferes with cones and rods. These structures use the input signal to generate electrochemical potentials (action potentials) which are further transmitted to the visual cortex of the brain. There, the information is processed and gets concious for human beings. This process is very sophisticated and effective as can be experienced in the daily life. Computer programs on the other hand are not as sophisticated as their biological counterparts and are not able to process images as effectively to detect certain features and regions of interest (ROIs). Take a look at the image presented in

Fig. 1.4. For computer programs, images are represented as an array of several rows and

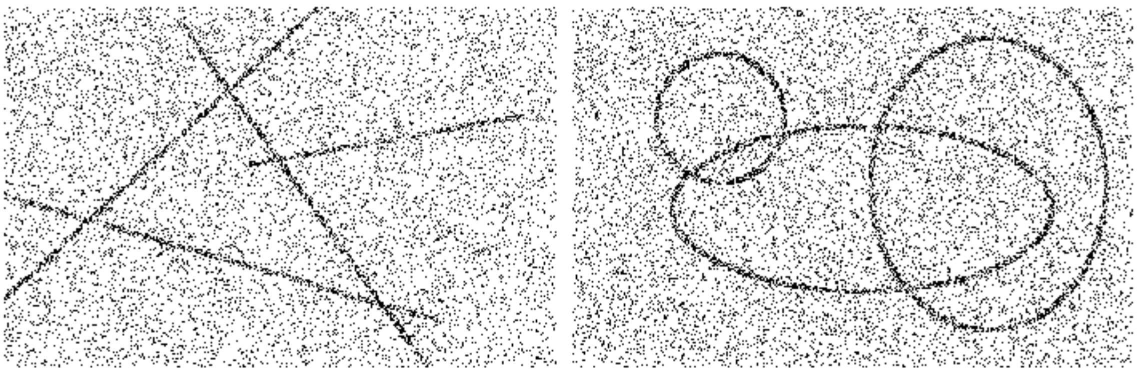


Figure 1.4: For humans it should not be too hard to identify the lines in the left image and the ellipses in the right image and distinguish the structures from the background noise. The task of this sort of feature registration in the field of digital image processing is not so trivial and easy to accomplish. Images were taken from Burger and Burge (2005).

columns filled with values according to the image type. Whether the picture is for example Red-Green-Blue (RGB), grayscale or binary the values would be for example three-layered 0-255 (each layer for one color red, green and blue), 0-255 or 0-1, respectively. Of course, the mentioned values (in case of RGB or grayscale images) depend on the bit-depth which is used (8 bit in this example). The challenging part when dealing with this type of image representation is to identify and extract specific shapes (for example lines, curves, circles etc.), edges and/or corners solely based upon the given array. Numerous different methods and algorithms have been developed in the past decades to cope with this problem, many of which are all based on one basic method and were further optimized and developed to more robust versions. One basic approach for detecting edges within an image is to use a gradient filter on an image. This method ‘searches’ for value changes from contiguous pixels and returns a binary image with the found contours. Based upon which method or operator is used, it is possible to define a threshold value for the slope, at which an edge is found. Furthermore, some operators have the option to specify the direction in which the edge shall be found. This can be very useful when expecting or specifically searching for, for example horizontal aligned edges or lines. Based on the theoretical background, it is straight forward to detect an edge within a binary image because the array values consists only of zeros and ones, so either there is a value change from one pixel (array cell) to another or not. This circumstance can not be applied as easily on grayscale or color images due to the fact that their array values may become very large, depending on the bit-depth of the image. Therefore, the mentioned threshold value for the gradient can be very useful for finding the desired edges or regions. Another important aspect of image processing is the fact, that computer programs have to cope with noise within the image. Noise can be very difficult to deal with in the sense that programs or algorithm should be robust against these kind of disturbance when analysing the image or searching for specific features. Noise makes itself noticeable as it renders homogeneous areas inside an image inhomogeneous and therefore harder for programs to distinguish between ‘true’ and ‘false’ features. Consider a homogeneous (white) region with black structure elements in an image compared to the same image when noise is added, as shown in Fig. 1.5. For

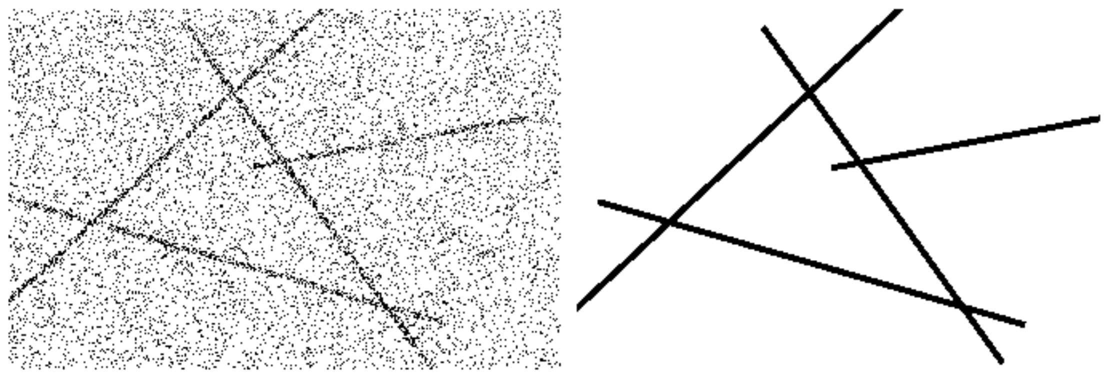


Figure 1.5: Comparison between a noisy image (left) and the same image without noise (right). The left image was taken from Burger and Burge (2005).

humans there is not much difference, with respect to the difficulty, to distinguish the black lines in both images from the background whereas a computer program may find several false lines or edges in the left image due to the noise artefacts which causes numerous jumps in adjacent pixel values. Many different filters were developed and improved in the past decades in order to encounter this issue by reducing or removing noise in an image. This filter procedure is known as smoothing or sharpening, depending which kind of filter method is used. These filters can be applied directly in the spatial domain or in the frequency domain, using a two dimensional discrete fourier transformation (2D-DFT) of the image . The latter case can yield a better computational performance , depending on the size of the filter, or can be used to realize (simple) filters which might be much more complex to realize in the spatial domain. There is no ideal or general filter available, which can or should be used for every image to yield the best possible results. Depending on the type of information that is required to extract from an image, one has to select manually the filter which is the most suited for the given task. A brief overview over the available filters, which are implemented in the program ImageJ (Schneider et al., 2012) shall be given in the next few sections. According to Gonzalez et al. (2009), the field of image processing can be divided into three parts:

Low-level processes *Low-level* processes are considered to be the preprocessing steps, which means that the focus lies on noise reduction, contrast enhancement and image sharpening. This level is characterised by the fact that both its inputs and outputs are typically images.

Mid-level processes *Mid-level* processes involve tasks like segmentation, description and classification of individual objects in an image. The characterisation of this process is that its inputs generally are images, but its outputs are attributes extracted from those images, like edges, contours or distinct parameters that identify a individual object.

High-level processes *High-level* processes try to 'make sense' of an ensemble of recognized objects and ultimately perform cognitive functions, which are normally associated with human vision.

2 Materials and Methods

The following chapter describes the materials used for analysis and how they were obtained, stored, digitized and finally analyzed. For reconstruction purposes it is crucial to know the fixation method, which staining agents were used, the imaging technique as well as the image analysis method. The imaging technique will be described briefly as it was not the main focus of this thesis, whereas the processing and analysis programs and the developed algorithm will be discussed in more detail. The chosen software for image processing and analysis was ImageJ (Schneider et al., 2012), which will also be described in this section. The use of autopsy material from human subjects for this study was approved by the Ethics Committee of Medical University of Graz (#: 21-288 ex 09/10).

2.1 Tissue sample preparation

For testing and analysis purposes, two different tissue types were used. One was obtained from a porcine aortic medial layer and the second one from a human aortic medial layer. The porcine aortic medial layer was further separated into two samples: one unstretched control sample and a second sample with 30 % circumferential and 20 % axial pre-stretch.

2.1.1 Porcine tissue for collagen visualization

Both specimens were fixed with the *High-Pressure Freezing* method. Therefore, the tissue samples were inlaid in 35 % formaldehyde, sectioned into $200\ \mu m$ pieces and mounted on aluminium platelets filled with hexadecane. This setup was immediately frozen afterwards with the high-pressure freezer HPM 010 (Bal-Tec, Principality of Lichtenstein). In the next step, a freeze-substitution was conducted followed by the embedding process.

Finally, ultrathin slices of $100 - 200\ nm$ were cut off the specimens with a diamond knife, mounted on carbon coated grids, stained with uranyl acetate and analyzed using the transmission electron microscope Tecnai 20 equipped with LaB6 electron gun.

2.1.2 Human tissue for collagen and proteoglycan visualization

The human tissue was prepared the same way as described for the porcine tissue in the previous section. Additionally, to visualize proteoglycans, the specimen was stained with the cationic dye *cupromeronic blue*.

2.2 Image processing with ImageJ

ImageJ is an open source program (not subject to copyright protection and is in the public domain (National Institutes of Health, 2004)) written in Java to handle (edit, manipulate, process, analyze,...) images of various types. Natively it supports the following image file formats (Ferreira and Rasband, 2012):

- TIFF
- GIF
- JPEG
- PNG
- DICOM
- BMP
- PGM
- FITS

This list can be extended by a number of file formats through additional plugins, which are available online at `http://rsb.info.nih.gov/ij/plugins/`. For further informations see ImageJWiki (2010). The major advantages of this program are listed below:

- It is freeware and can therefore be used for any purposes without any costs
- Cross-platform availability due to Java programming language (Win/Linux/Mac and others)
- Program extensibility based on (self-written) Macros, JavaScript and Java PlugIns
- Big community which provides improvements to the program and numerous PlugIns
- Possibility to study, improve and / or modify the implemented algorithms of all functions if necessary.

Based on these facts, ImageJ was chosen over MATLAB® from The MathWorks®. Many filters, mathematical operations, binary operations etc. are already implemented in the downloadable version.

2.2.1 Preprocessing functions

Many preprocessing functions are already implemented in ImageJ and shall be briefly introduced in the following section. The information on how the different filters are implemented was taken from the ImageJ manual by Ferreira and Rasband (2012).

Convolve This filter convolves the input image with a specified filter matrix (named *kernel*). This is accomplished by 'sliding' the kernel over the input image, such that the center element of the kernel moves over every pixel of the input image once. The structure of the kernel is a $M \times N$ matrix, where M and N must be odd numbers to ensure a true center element. As the filter matrix moves over the image, the new (filtered) pixel value (where the center kernel element is located) is computed by the sum of all (center and neighbor) image values multiplied by their corresponding (overlaid) kernel values. To calculate the edge pixels of the input image, their values are extended outwards to match the kernel size.

Gaussian Blur Here, the input image is convolved with a Gaussian function which results in a smoothing effect. The variable parameter is the standard deviation *sigma*.

Median Each pixel in the original image gets replaced by the median value of its neighbors and therefore can reduce noise. The neighborhood radius, as in all following filters, can be adjusted freely and is implemented as a circular filter mask.

Mean This filter replaces each pixel with the mean value of its neighborhood pixels and as a result smoothens the image.

Minimum / Maximum Replaces every pixel value with the minimum or maximum value of his neighboring pixels. This process is similar to the binary operations *erode* or *dilate* but applied on grayscale images.

Unsharp Mask Originally introduced for the analog film technology and later adapted for digital image processing (Burger and Burge, 2005), this filter sharpens images and enhances edges through subtracting a smoothed version of the original image (the *unsharp mask*) from the original image. Usually Gaussian filters are used to smooth or blur the image, therefore the adjustable parameters are the standard deviation *sigma* and a weighting factor *Mask Weight* which ranges from 0.1 to 0.9. This weighting factor determines the strength of filtering and may be used for additional edge enhancement.

Variance This filter replaces the target pixels value with the neighborhood variance.

2.2.2 Processing functions

Threshold In order to perform binary operations, the color or grayscale input image has to be binarized first. Therefore a thresholding method is present which consists of a lower and a upper threshold value that can be slided over the image histogram. Thus it is possible to isolate the objects of interest, as only pixel values which are between the two boundaries are considered foreground elements. This procedure can be done manually or automatically with numerous different algorithms at hand. These algorithms calculate (based on distinct conditions) one threshold value by analyzing the histogram whereas the second threshold value is fixed at either the lowest or highest pixel value. The threshold technique implemented in ImageJ only works globally and effects the whole image at once. This circumstance may cause troubles in the case of an uneven brightness distribution, making it difficult to segment the image properly. To overcome this limitation, a plugin is available from Landini (2012b), which uses local threshold methods to calculate a threshold value for every pixel, based on an adjustable radius parameter for the pixels neighborhood.

Make Binary Converts the input image to a black and white, binary image. This is done either with a previously set threshold value in the *Threshold...* dialogue or, if no value has been set, the threshold will be computed automatically. This is accomplished using a variation of the implemented IsoData algorithm (Landini, 2012a; Ridler and Calvard, 1978).

Erode This function removes one pixel from the edges of every object in the binary image. In the *binary options* dialogue, a value *Count* can be set which determines the minimum number of adjacent background pixels necessary so that one edge pixel gets removed. Erosion may be used to eliminate small outliers.

Dilate Contrary to the *Erode* function, dilation adds one pixel to the edges of every object in the binary image. The *Count* value in the *binary options* dialogue works similar as in the *Erode* function.

Open / Close The *Open* operation is an erosion followed by an dilation whereas the *Close* operation is the opposite, a dilation followed by an erosion.

Outline This option produces a one pixel wide outline around each object in the binary image. This is achieved by eroding the original image and then subtracting the eroded outcome from the original image. In the process, every pixel except the ones on the edges of objects get eliminated.

Fill Holes With this command, holes (4-connected background pixels) found in object can be filled by setting these background elements to foreground elements.

Skeletonize This procedure reduces an objects shape to its thinnest, one pixel wide representation that is equidistant to its boundaries. There are several different algorithms available to perform this task, and the one from Zhang and Suen (1984) is implemented in ImageJ. First, Zhang and Suen created a lookup table where all 256 possible 3×3 foreground pixel configurations around the center element are indexed. Then the algorithm computes repeatedly the index number of foreground (object) pixels and cross checks this number with the lookup table in order to decide whether the pixel is removable or not, until no more elimination can be done.

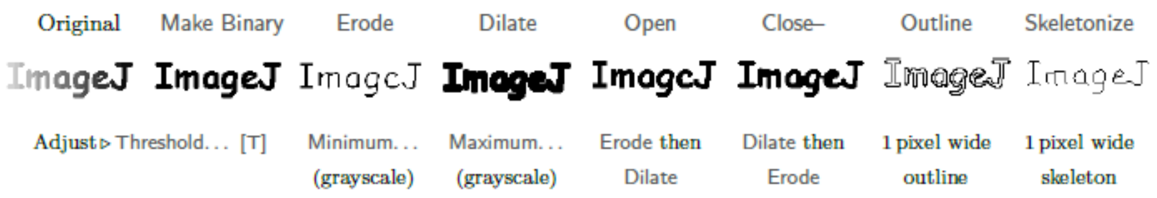


Figure 2.1: A comparison between the different binary operations is presented. This illustration was taken from (Ferreira and Rasband, 2012).

2.2.3 Analysis functions

2.2.3.1 Set Scale

By default, ImageJ uses *pixel* as unit of length. If the image contains a scale bar, it is possible to tell ImageJ the actual spatial dimensions with respect to the pixel size. The *Set Scale...* option (*Analyze -> Set Scale...*) is intended for this functionality, as it offers the possibility to define a specific scale parameter. If the $\frac{pixel}{unitlength}$ ratio is known, the values can be inserted into the corresponding fields in the dialogue and confirmed. To acquire the scale parameter from the scale bar in the image, a line has to be drawn with the *straight line tool* along the bar and then the *Set Scale...* option must be selected. Now the *Distance in pixels* field value is automatically inserted as the length of the drawn line. Next, insert the known distance (i.e. the scale bar length) in the corresponding field and finally type in the unit of length in the designated text area. If the desired scale is the same for successive images to analyze, the *Global* option may be checked so that every image is handled with the given scale value.

2.2.3.2 Set Measurements

Now that the appropriate scale is set, the desired measurements shall be chosen. These are available at the *Set Measurements* dialogue (*Analyze->Set Measurements*). For the developed plugin, the following measurements are mandatory:

- Area
- Centroid
- Fit ellipse

Furthermore, three decimal places where chosen.

Area Returns the area of each particle in $pixels^2$ or in individually given $units^2$ if they were set with the *Set Scale...* option.

Centroid This measurement gives the X and Y coordinates of the center element of each particle, which are computed as the average of the x and y coordinates of every pixel in each particle.

Fit ellipse An ellipse is fitted to each particle in the image, returning the *major*, *minor* and *angle* of the ellipse. *Major* and *minor* are the primary and secondary axis of the ellipse, respectively, while the *angle* is defined as the angle between the *major* axis and a line parallel to x-axis of the image.
With these parameters set, the particles may now be further analyzed.

2.2.3.3 Analyze Particles

This command searches for particles by scanning over the binary image and looking for edge pixels. If the algorithm finds a particle, it gets measured by means of the *Wand tool*. Several options are available at the *Analyze Particles* dialogue to alter the findings of the program.

Size Defines a area size range (default from $0 - \infty\ unit^2$) which a particle must have in order to be recognized. *unit* is either *pixel* or an individually defined unit in the *Set Scale* menu.

Circularity This parameter is kind of a shape distinction, as it defines a range in which a particles circularity value must be. The circularity is defined as

$$circularity = 4\pi \frac{[Area]}{[Perimeter]^2}. \tag{2.1}$$

and lies between $0 - 1$, where 0 is a infinitely elongated polygon and 1 represents a perfect circle.

Show The resulting measured particles can be displayed with this option. Several different visualization methods are available:

- Nothing - shows a blank image
- Outlines - shows labeled particle outlines (edges)
- Bare Outlines - same as *Outlines* but without labels
- Ellipses - draws the best fit ellipse of each particle
- Masks - shows filled particle outlines without labels
- Count Masks - particle maskes filled with a grayscale value corresponding to their label number

- Overlay Outlines - same as *Bare Outlines* but shown as image overlay
- Overlay Masks - same as *Masks* but shown as image overlay

2.2.3.4 Overlays

Overlays are non-destructive image elements, which can be overlaid over any given image and are therefore the best way of annotating said images. Contrary to the mainly used *raster graphics*, overlays are *vector graphics* and are thus not affected by scaling. Up until now, it is not possible to save these overlays separately as vector graphics but only as raster graphics. Alternatively, overlays are stored in the header of *Tiff* images and can thus be used again if the *Tiff* images are loaded again in ImageJ.

2.3 Development of feature extraction algorithm

The main goal of this thesis was the extraction and determination of microstructural parameters from proteoglycans in conjunction with collagen fibrils within TEM-images. Basically these parameters could be measured manually but since collagen fibrils and proteoglycans can appear in large quantities in such images, an automated algorithm was desired to handle this task at a low computational (low time consuming) cost. Therefore, a Java plugin for ImageJ was developed to obtain automatically and compute microstructural parameters such as

- Interfibrillar distance,
- Collagen fibril diameter,
- Spatial distribution of collagen fibrils
- Number of neighbouring collagen fibrils
- Proteoglycan orientation with respect to that of collagen fibrils

2.3.1 Collagen fibril parameter acquisition

As mentioned before, the *Analyze Particles* command returns the X and Y center coordinates of each found fibril, as well as the *minor* value from the best fit ellipse measurement.

2.3.1.1 Collagen fibril diameter

Due to the fact that collagen fibrils are cylindrically shaped , they appear as circles or ellipses (if they don't run straight out of the plane but are slightly skewed) in cross-section images. Therefore, the *minor* value will always represent the actual fibril diameter, independent of the fibril appearance.

2.3.1.2 Interfibrillar distance

The interfibrillar distance is defined as the distance between one fibril's edge to an adjacent fibril's edge. Based on the available information gathered from the *Analyze Particles* command and by the fact that every XY coordinate can be assigned to be the endpoint of a hypotenuse from a right angled triangle, it is possible to calculate the distance between each fibrils central coordinate by harnessing the *Pythagorean theorem*

$$c^2 = a^2 + b^2. \tag{2.2}$$

As all central coordinates are given, the difference amongst two X or Y coordinates yields

$$\Delta X = X_2 - X_1, \tag{2.3}$$
$$\Delta Y = Y_2 - Y_1, \tag{2.4}$$

and further with ΔX and ΔY representing the catheti of the right triangle, the distance can be obtained with

$$d = \sqrt{\Delta X^2 + \Delta Y^2}. \tag{2.5}$$

The 'real' or effective distance between the fibrils (i.e. between their closest edge points) is further calculated by subtracting both fibril radii from the calculated distance d.

$$d_{\mathrm{real}} = d - r_{\mathrm{f1}} - r_{\mathrm{f2}} \tag{2.6}$$

with

$$r_{\mathrm{f1}} = \frac{minor_{\mathrm{f1}}}{2}, \tag{2.7}$$
$$r_{\mathrm{f2}} = \frac{minor_{\mathrm{f2}}}{2}. \tag{2.8}$$

The radii r_{f1} and r_{f2} from the arbitrary *fibril 1* and *fibril 2*, respectively, are obtained by dividing the *minor* (which is the fibril diameter) from each fibril in half.

2.4 Proteoglycan orientation determination

For proteoglycan orientation measurements, longitudinal sectioned images are used because the relevant information is the relative angle between the proteoglycans and the collagen fibril they are connected to. This measurement process is not as easy to execute as for the fibril parameters due to the geometry of the features of interest. A proper staining method and process in conjunction with an appropriate imaging technique is mandatory to distinguish the proteoglycans from the collagen fibrils within the image. This is especially necessary for an automated analysis method. Figure 2.2 shows a typical transmission electron microscope (TEM) image from tissue containing PGs and collagen fibrils. As can be seen, the fibrils are not aligned parallel to each other but show a kind of waviness throughout the image plane (at least in an unloaded state). Thus it is not possible to define a specific fibril edge direction to which the orientation of the proteoglycans that protrude away from the fibril could be measured. There are TEM images available, that show that this fibril waviness disappears when applying load to the tissue, which means that an automated measurement approach is much more feasible in this case. The straightened fibril axis can be aligned along the x-axis of the image, then the orientation of the (thresholded) PGs can be measures by performing the *Analyze Particles* technique with the *Fit Ellipse* option enabled. As the fibrils are aligned horizontally at $0\,^{\circ}$, the measured angle of the best fit ellipse would represent the relative angle between the PGs and their associated fibrils.

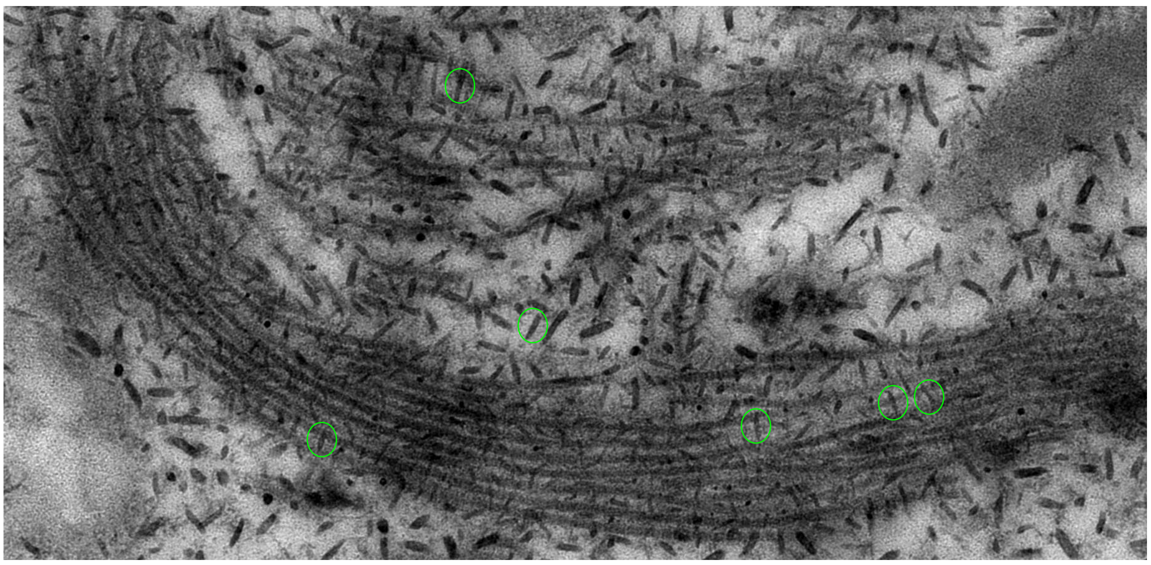


Figure 2.2: TEM images featuring proteoglycans (for example, in green circles) and collagen fibrils (long dark stained fibers).

To obtain the proteoglycan orientation with ImageJ, a manual approach was chosen by measuring the angle between a collagen fibril edge and the main axis of the protruding proteoglycan. This was done by using the implemented *Angle tool* from ImageJ. The main PG axis was determined by processing the image with the *Skeletonize* operation which, as described earlier, ‘turns’ the PG into a one pixel wide line with the same orientation as the unprocessed object. For clearer appearance the skeleton image was further colored green and overlaid over the original image, see Fig. 2.3. Now, the PG angle was measured by manually aligning the *Angle tool* along a collagen fibril edge until the intersection point with the skeletonised PG. From there, the measurement line was pulled in the direction of the protruding PG so that both lines (from the skeleton and the *Angle tool*) overlapped and therefore yield the desired angle.

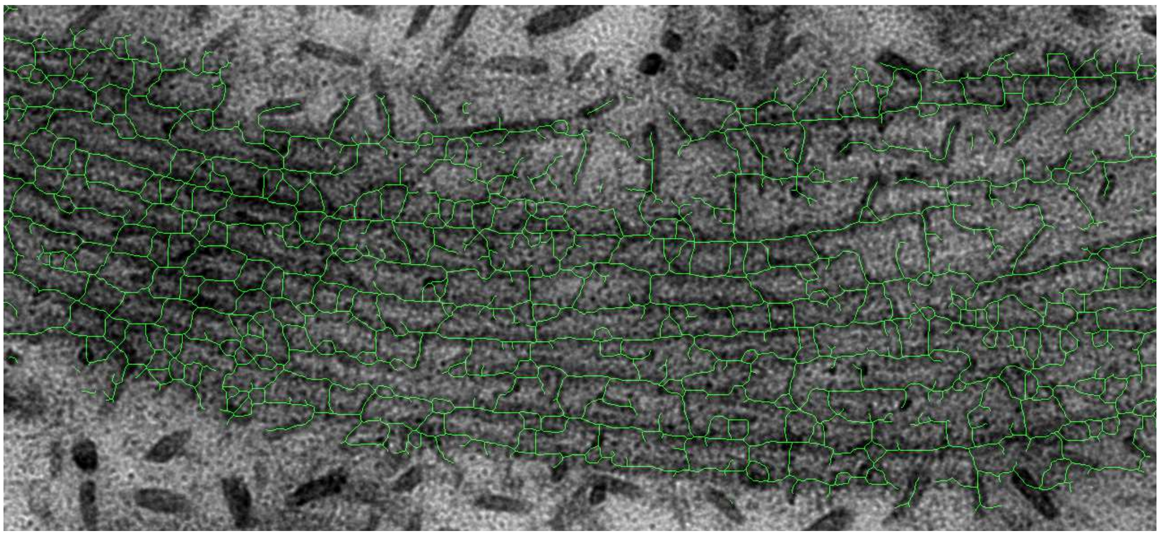


Figure 2.3: Transmission electron microscope images featuring proteoglycans and collagen fibrils overlaid with a skeleton mask.

2.5 Development of the ImageJ plugin

2.5.1 Installation

Installation of a Java plugin in ImageJ is quite easy:

- If the '.java' file is available, which contains the plain source code, just copy and paste it into the 'plugins' directory or subdirectory of ImageJ, select 'Plugins' in the menu bar and click on 'Compile and Run...'. A 'File open' window will appear, select the copied '.java' file and click 'Open'. The 'Compile and Run...' step only needs to be done once, because afterwards a '.class' file will be created, which has the same name as the '.java' file. This '.class' file represents the compiled version of the plugin and also has to reside within the 'plugins' folder or subfolder. From now on (after a restart of ImageJ), the plugin is accessible under 'Plugins' (it appears as the given filename, without the underscore).

- If either the '.class' or the '.jar' file is available, copy and paste it within the 'plugins' folder / subfolder as mentioned and described above. Again, a restart of ImageJ is necessary for the plugin to appear in the 'Plugins' menu

Important notes:

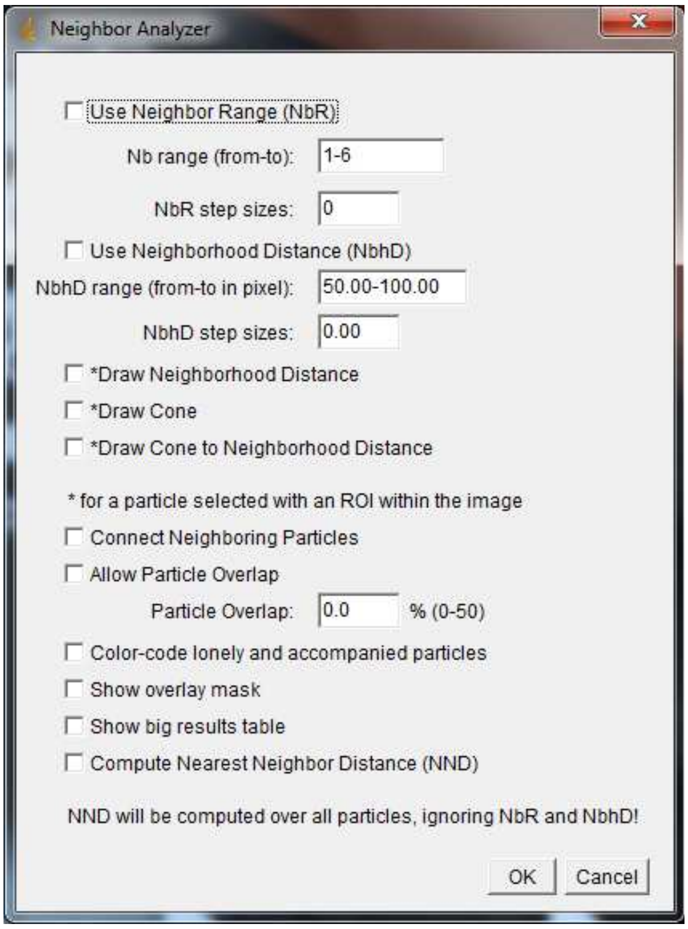
- ImageJ plugins must be in the 'plugins' folder or in one of its subfolders

- The plugin name must contain an underscore (for example name_.java), otherwise ImageJ won't list it under 'Plugins'

- (when modifying the source code) The filename and the classname (in the source code) must be the same

2.5.2 Usage

To prevent any ‘naming’ confusions in the upcoming part it shall be mentioned, that due to the purpose for which this plugin was developed, ‘particles’ (as they are called in ImageJ) are from now on referred to as (collagen) fibrils. Just to clarify, this program can of course be used to analyze any kind of particle appearances in images - preferably circular ones (in the current version). The interface of the plugin is shown in Fig. 2.4 As can be seen


Neighbor Analyzer

☐ Use Neighbor Range (NbR)
Nb range (from-to): 1-6
NbR step sizes: 0
☐ Use Neighborhood Distance (NbhD)
NbhD range (from-to in pixel): 50.00-100.00
NbhD step sizes: 0.00
☐ *Draw Neighborhood Distance
☐ *Draw Cone
☐ *Draw Cone to Neighborhood Distance

* for a particle selected with an ROI within the image
☐ Connect Neighboring Particles
☐ Allow Particle Overlap
Particle Overlap: 0.0 % (0-50)
☐ Color-code lonely and accompanied particles
☐ Show overlay mask
☐ Show big results table
☐ Compute Nearest Neighbor Distance (NND)

NND will be computed over all particles, ignoring NbR and NbhD!

Figure 2.4: ImageJ plugin interface

in Fig. 2.4, several options are available which alter the image analysis results and the representation of the resulting overlay.

Use Neighbor Range This option enables the neighbor range, which consists of a lower and an upper value. The program will iterate starting from the *from* value to the *to* value at given *NbR step sizes*. If the step size is 0, only the two given range values will be used, i.e. two iterations. The neighbor range may help to reduce the programs execution time.

Use Neighborhood Distance and Neighborhood Distance Range The first option tells the plugin to search only within a given distance (second option) circular around a fibril. The distance is measured from fibril edge to edge. Like the 'Neighbor Range' option, this can help to reduce computation time and more important limits the range of finding 'true' adjacent fibrils, which are interconnected by proteoglycans. The unit (in Fig. 2.4 *pixel*) varies according to any given unit in the 'Set Scale...' setting, as described previously in section 2.2.3.1. Like for the neighbor range, the step size tells the program 'how fast' it shall go from *from* to *to*.

***Draw Neighborhood Distance** This option defines whether the 'Neighborhood Distance' shall be drawn in the resulting overlay and has therefore no further impact on the results. It is intended for visualization purposes to see which area is occupied by the fibril neighborhood distance (if it is given).

***Draw Cone and *Draw Cone to Neightborhood Distance** Like the former option, these two have no direct influence on the findings. With this selection, a cone is drawn from the (selected) fibril to its neighboring fibrils or to the (given) neighborhood distance. The latter option may be interesting to see how the *Particle Overlap* function works.

Note The options marked with a * will only be considered, if a single fibril in the image mask is selected with one of the available selection tools from ImageJ.

Connecting Neighboring Particles If it is desired, this option visually connects adjacent fibrils through lines with each other. Note that whether this is selected or not, lines between a selected fibril and its neighboring fibrils will still be drawn, if the 'Draw Cone' option is activated.

Allow Particle Overlap and Particle Overlap If there are for example two fibrils which can both be considered as neighbors to a third fibril, but are close one after another and overlap, then this option comes into play. With a percentage value between 0 and 50, an 'allowed' overlap can be defined at which the backmost overlapping fibril is still considered a valid neighbor to a third fibril. This behaviour is illustrated in Fig. 3.1 - 3.3.

Color-code lonely and accompanied particles With this option enabled, fibrils with or without neighbors will be drawn green or red, respectively, as can be seen in Fig. 3.1.

Show overlay mask This option must be set in order to see the visualization of the results.

Show big results table In addition to the standard *Statistic Table*, *Distance Table* and *Diameter Table*, the *'Big Results Table'* shows all informations about a fibril and its neighbor fibrils including their origin and edge coordinates or their angles to one another. These pieces of information are in most cases and for the most users not interesting as they are intended for debugging purposes.

Compute Nearest Neighbor Distance (Nnd) As the name suggest, with this option enabled the Nnd will be presented in an extra table. In conjunction with the *Show Mask* and the *Connect Neighboring Particles* options, the nearest neighbor distances will be drawn in a different color in order to identify them easily in the overlay mask.

To demonstrate how this plug is used, a suggested workflow for analysing images is illustrated by flowcharts in the upcoming sections, beginning with the first shown in Fig. 2.5 (flowcharts were created using `http://www.gliffy.com/gliffy/`). Afterwards, an example analysing procedure is given in Fig. 3.4, 3.5 and 3.6.
First, an image containing the desired features has to be loaded into ImageJ. Second, it is highly recommended to create a duplicate of the original image and later on of any altered subsequent images. Therefore it is easier to recover from wrong processing decisions. Even though an *undo* option is available, it is, based on experience, wiser to make a duplicate after each crucial processing steps which brought good results. The following steps would be to locate the regions of interest in the image. Note: If more than one area contains the desired features, the following selection process should be done separately for each ROI. By using an appropriate selection tool from ImageJ, the ROI can be isolated from the rest of the image by drawing a frame around the region and selecting *Edit -> Clear Outside* from the menu. This command blanks the whole image except the selected area. The resulting 'image mask' should be saved and if other regions of interest were present in the original image, this selection process can be done again as often as necessary by using the initially created duplicate (after duplicating it again of course).
Basically it is possible to either leave the whole image as it is and continue directly to the image processing phase or at least group all sections together on one plane and process the resulting image mask. The reason for the rather laborious procedure described above is that every ROI has its own characteristic processing necessity. Filters or binary operations used to get rid of noise and certain unwanted structures can be applied successfully on one ROI but may fail and remove precious features on other ROIs. Therefore it was found that the best results can be achieved by separating the desired areas from each other and process each of them individually. Afterwards it is possible to combine the final binary masks again into one big image so that all ROIs can be analyzed at once, but this is not recommended due to computational costs as will be discussed later on.

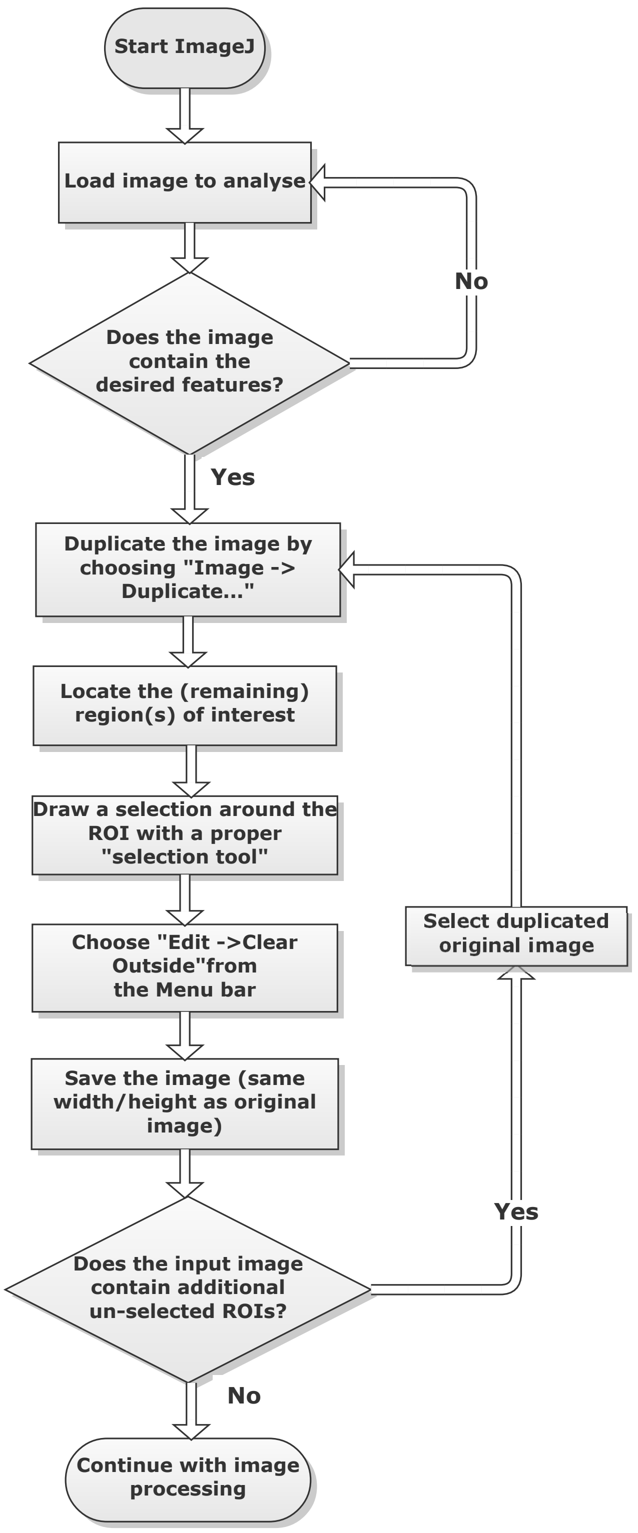

Start ImageJ
Load image to analyse
Does the image contain the desired features?
No
Yes
Duplicate the image by choosing "Image -> Duplicate..."
Locate the (remaining) region(s) of interest
Draw a selection around the ROI with a proper "selection tool"
Choose "Edit ->Clear Outside"from the Menu bar
Save the image (same width/height as original image)
Does the input image contain additional un-selected ROIs?
Yes
Select duplicated original image
No
Continue with image processing

Figure 2.5: The given flowchart illustrates how to prepare images properly for analysis prior to any (pre) processing applications. Following this 'guide' is advised when working with ImageJ.

The next flowchart in Fig. 2.6 illustrates a basic image processing routine towards the analysis phase.
Starting with a previously sectioned image, different image processing techniques (like filter, contrast enhancement,...) can be applied in order to reduce or remove noise or artefacts and enhance the distinguishability from the features to their vicinity. Following this step, the resulting image should be duplicated and binarized by applying a threshold either manually (preferred) or automatically. If the binary results are satisfying, i.e. no major unintended structures interfere with the features of interest, and all collagen fibrils or proteoglycans are clearly visible, further binary operations like *erode*, *dilate*, *open* or *close* may be applied to get rid of remaining unwanted objects. Otherwise if the image was not properly segmented, either another threshold value yield better results or the whole (pre)processing procedure has to be rerun.
After the application of binary operations, the image mask is either ready for further analysis or it may need some 'polishing' by manually erasing (small) spurious objects. Ultimately, before going onwards to the image analysis section, the created image mask should be saved for, e.g., later use.

The final flowchart regarding the plugin usage is presented in Fig. 2.7. In case the *Set Scale* parameter wasn't set previously, it should be defined here if a scale bar is given in the original image or the $\frac{pixel}{unitlength}$ ratio is known. Working with the real length scale provides easier determination of options regarding the size or length of features of interest. Furthermore, all results from measurements and calculations will be presented in the given length scale which makes them easier to interpret 'on sight'. For a proper plugin execution, it is mandatory, as was formerly stated, that the measurement fields *Area*, *Centroid* and *Fit ellipse* are all selected in the *Set Measurements...* dialogue. Now the *Analyze Particles* command can be executed to analyze the present features in the image in order to receive their coordinates and size. Note that none of the available options in the command dialogue of *Analyze Particles* is necessary to be selected, still it is recommended that *Show* is set to anything other than 'nothing' due to the fact that this will show a new image with all analyzed particles which were found in the input image. This helps to visually confirm that all desired features were detected, especially if the *Size* and/or *Circularity* settings were adjusted. Furthermore, the latter two mentioned settings may help to ignore certain structures which exhibit a certain shape or size that shall not be analyzed.
After the outcomes have been verified, the new *Neighbor Analyzer* may now be started and configured preferably. It shall be mentioned here that high *Neighbor Range* values in conjunction with large *Neighborhood Distance* values may result in excessively long computation times ($>$10 minutes). The created image mask consists solely of overlay structures which can be overlaid over the original image to see if the results match the initially desired features. Additionally, as stated earlier overlays are vector graphics and thus don't get pixelated so the final results can be examined in great detail. It is advised to save the image mask in the *TIFF* file format in order to keep the overlays for later use.

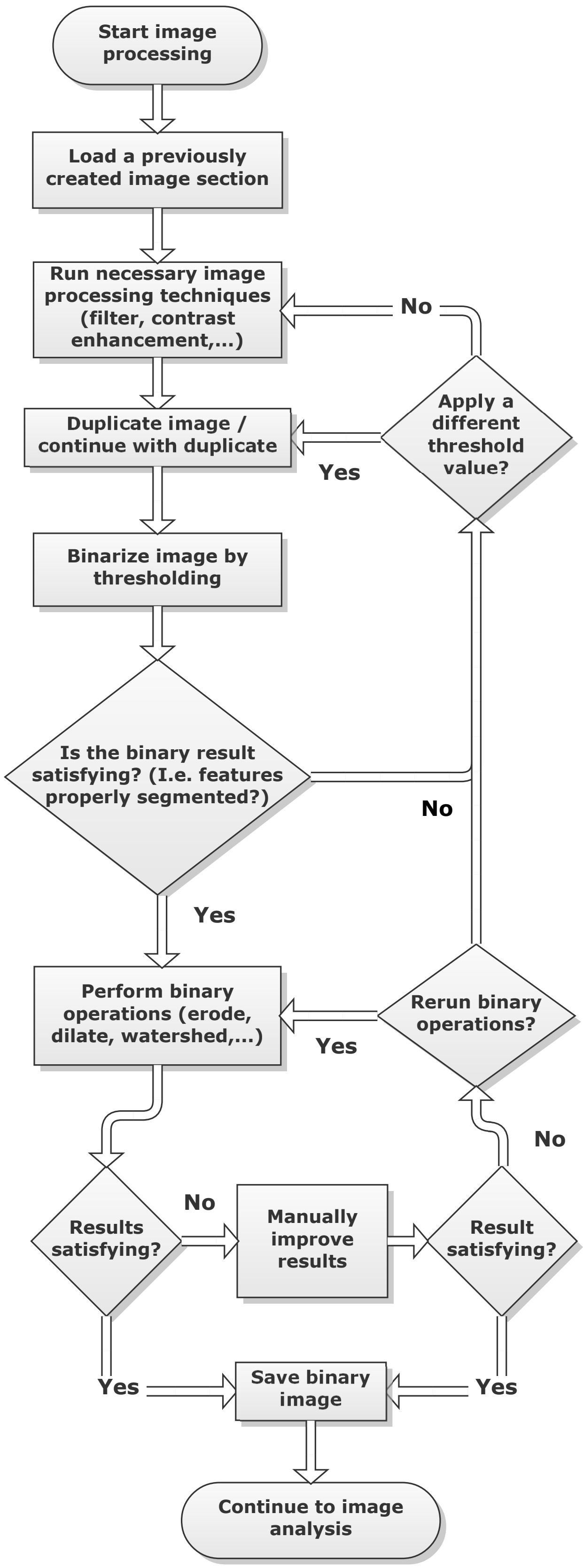

Start image processing
Load a previously created image section
Run necessary image processing techniques (filter, contrast enhancement,...)
No
Duplicate image / continue with duplicate
Apply a different threshold value?
Yes
Binarize image by thresholding
Is the binary result satisfying? (I.e. features properly segmented?)
No
Yes
Perform binary operations (erode, dilate, watershed,...)
Rerun binary operations?
Yes
No
Results satisfying?
No
Manually improve results
Result satisfying?
Yes
Save binary image
Yes
Continue to image analysis

Figure 2.6: The basic procedure regarding the image processing part is shown. A more detailed description to the presented flowchart is given in the text. Afterwards the image should be properly prepared for further analysis.

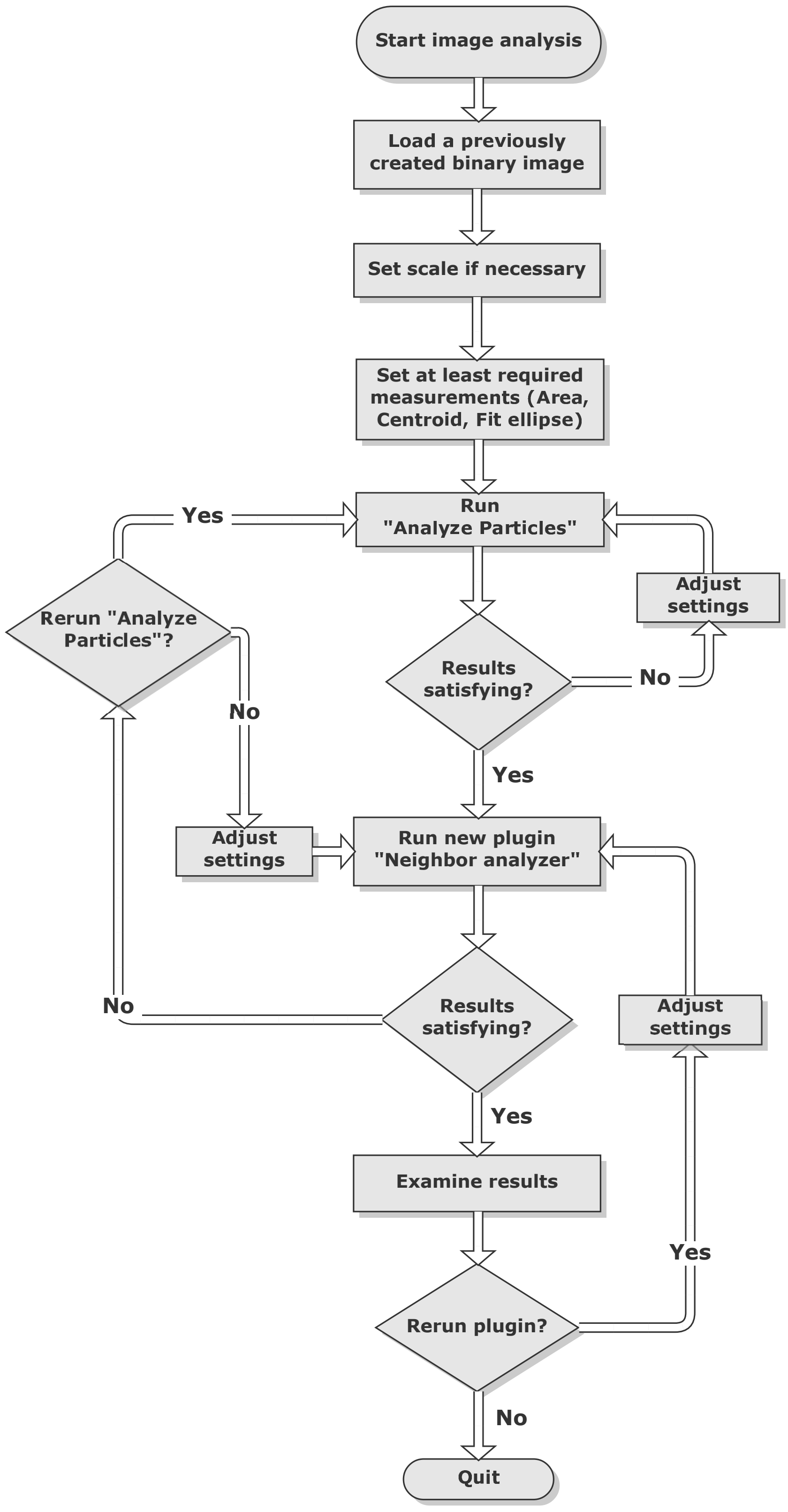


Figure 2.7: Demonstration of the suggested image analysis procedure after the desired image has been properly prepared. This final step inhibits the ‘Analyze Particles’ command as well as the developed ‘Neighbor analyzer’ plugin. An in depth description is discussed in the continuous text.

Alternatively these masks can get *flattened* which means that the overlays are converted into raster graphics that can be saved in more common file formats such as *GIF*, *JPEG* or *PNG*. The downside of this method is that they get pixelated when enlarged and also can't be overlaid over other images anymore.
If it is desired the plugin can be run again with different settings as the *Analyze Particles* results are still available.

2.5.3 Internal plugin functionality

The upcoming part focuses on the actual realization of the plugin. To get a quick overview over the main parts of the program, flowcharts were designed for each major section which can be seen from Fig. 2.8 - 2.12. Additionally, for abbreviation explanations a textbox was included on every flowchart page that includes all necessary information regarding the used variable names.

2.5.3.1 Main function

The main part is illustrated in Fig. 2.8. After passing the initial check whether the results from *Analyze Particles* are available and valid, the user gets the plugin interface presented (Fig. 2.4). With accordingly set parameters, the program calculates the required Neighbor Range (NbR) and Neighborhood Distance (NbhD) iterations by

$$itR = \left\lceil \frac{maxNbR - minNbR}{NbRss} \right\rceil, \tag{2.9}$$

$$itD = \left\lceil \frac{maxNbhD - minNbhD}{NbhDss} \right\rceil, \tag{2.10}$$

where *itR* and *itD* are the NbR and NbhD iterations, respectively. The variables *maxNbR*, *minNbR*, *NbRss*, *maxNbhD*, *minNbhD* and *NbhDss* are the maximum NbR, minimum NbR, NbR step size, maximum NbhD, minimum NbhD and NbhD step size values, respectively. Furthermore, a first approximation regarding the required total processing steps *without* the Nearest neighbor distance (Nnd) computation (for a progress bar visualization of the remaining computation time) is given by

$$tPSs = nItems(nItems - 1)(itR + 1)(itD + 1) + itR + itD + 2. \tag{2.11}$$

If a Nnd determination is performed, Eq. (2.11) becomes

$$tPSs = nItems(nItems - 1)((itR + 1)(itD + 1) + 1) + itR + itD + 3. \tag{2.12}$$

In both Eq. (2.11) and (2.12) *nItems* is the number of particles found returned by the *Analyze Particles* operation. The abbreviation *tPSs* stands for *total Progress Steps*. The reason for the different equation is that an extra loop is performed when the Nnd is calculated due to the fact that this computation must be conducted without any NbhD restrictions.

Moving on, two nested *for-loops* are used, the first iterates over the NbR and the second one over the NbhD. In each loop, the current Neighbor count $cNbC$ and the current NbhD are calculated as

$$cNbC = itRNbRss + minNbR, \tag{2.13}$$
$$cNbhD = itDNbhDss + minNbhD, \tag{2.14}$$

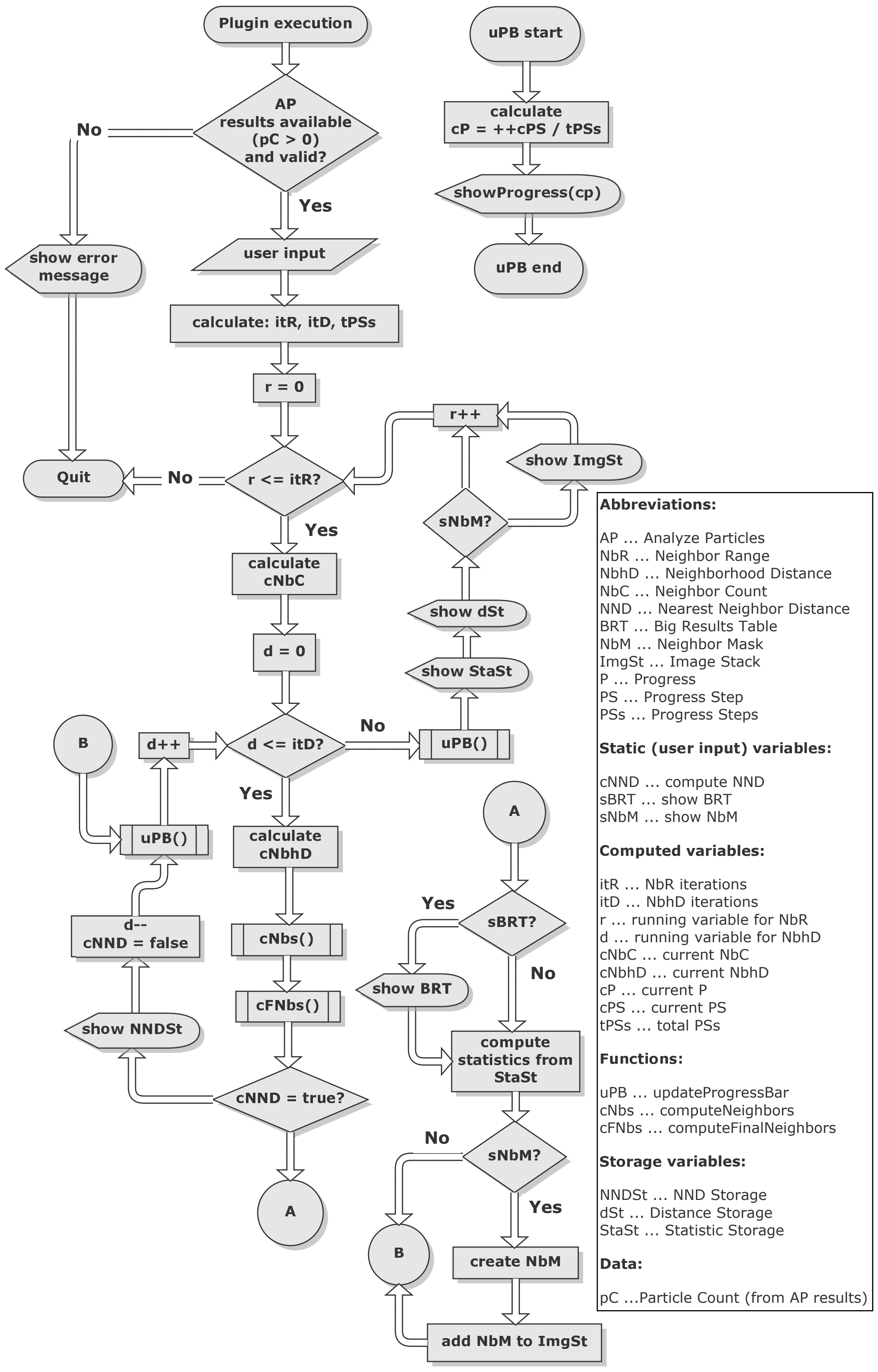
which are further used as boundary conditions in the process. The next steps are the actual neighbor computations in the *computeNeighbors (cNbs)* and *computeFinalNeighbors (cFNbs)* functions which can be seen in Fig. 2.9 and Fig. 2.12 and will be described in Sec. 2.5.3.2 and Sec. 2.5.3.3, respectively. If the Nnd is desired, the normal flow will be broken after the *finalNeighbors* computation and the inner loop gets reset. This procedure will only occur once during the whole plugin execution process.
In any other than the 'Nnd case' the program continues to calculate a number of statistical data like the mean, standard deviation (σ), median, median absolute deviation (MAD) etc. for the given distances, diameters or number of neighbors. Afterwards the progress bar gets updated by way of

$$cP = \frac{cPS + 1}{tPSs}, \tag{2.15}$$

where *cP* and *cPS* are the current progress and the current progress step, respectively.
When the inner loop is finished, the gathered statistical data is presented in a table for all processed neighborhood distances. Additionally, all measured distances for the current maximum neighbor count as well as the created overlay mask is displayed. The overlay mask is generated as an image stack which enables to whole NbhD range to be viewed as a slideshow.

2.5.3.2 ComputeNeighbors function

The first of the two neighbor computation functions is shown in Fig. 2.9. Basically it iterates two times over *nItems* in order to determine the length between each particle. Furthermore, this function already limits the possible number of neighbors by applying the current NbhD restriction on the calculated length. If the current neighbor count or the current NbhD is zero, the inner loop gets skipped because no interconnections will be present under these conditions. Within the second loop, the lengths between the selected particle from the first loop is calculated to each particle from the inner loop by using Eq. (2.2) - (2.8). When the resulting length is less than or equal to the current NbhD, the two particles are considered to be neighbors and thus get stored in a neighbor storage variable. In order to reduce computation time, both particles get stored 'in both directions' in one cycle, so when the current inner loop particle becomes the outer loop one and vice versa, the data between these two doesn't have to be computed again.
During this procedure the progress bar gets updated several times in order to keep the user up to date in regards of the remaining execution time. After all possible combination were processed, the complete neighbor storage variable gets sorted by the calculated lengths in ascending order.



Abbreviations:

AP ... Analyze Particles
NbR ... Neighbor Range
NbhD ... Neighborhood Distance
NbC ... Neighbor Count
NND ... Nearest Neighbor Distance
BRT ... Big Results Table
NbM ... Neighbor Mask
ImgSt ... Image Stack
P ... Progress
PS ... Progress Step
PSs ... Progress Steps

Static (user input) variables:

cNND ... compute NND
sBRT ... show BRT
sNbM ... show NbM

Computed variables:

itR ... NbR iterations
itD ... NbhD iterations
r ... running variable for NbR
d ... running variable for NbhD
cNbC ... current NbC
cNbhD ... current NbhD
cP ... current P
cPS ... current PS
tPSs ... total PSs

Functions:

uPB ... updateProgressBar
cNbs ... computeNeighbors
cFNbs ... computeFinalNeighbors

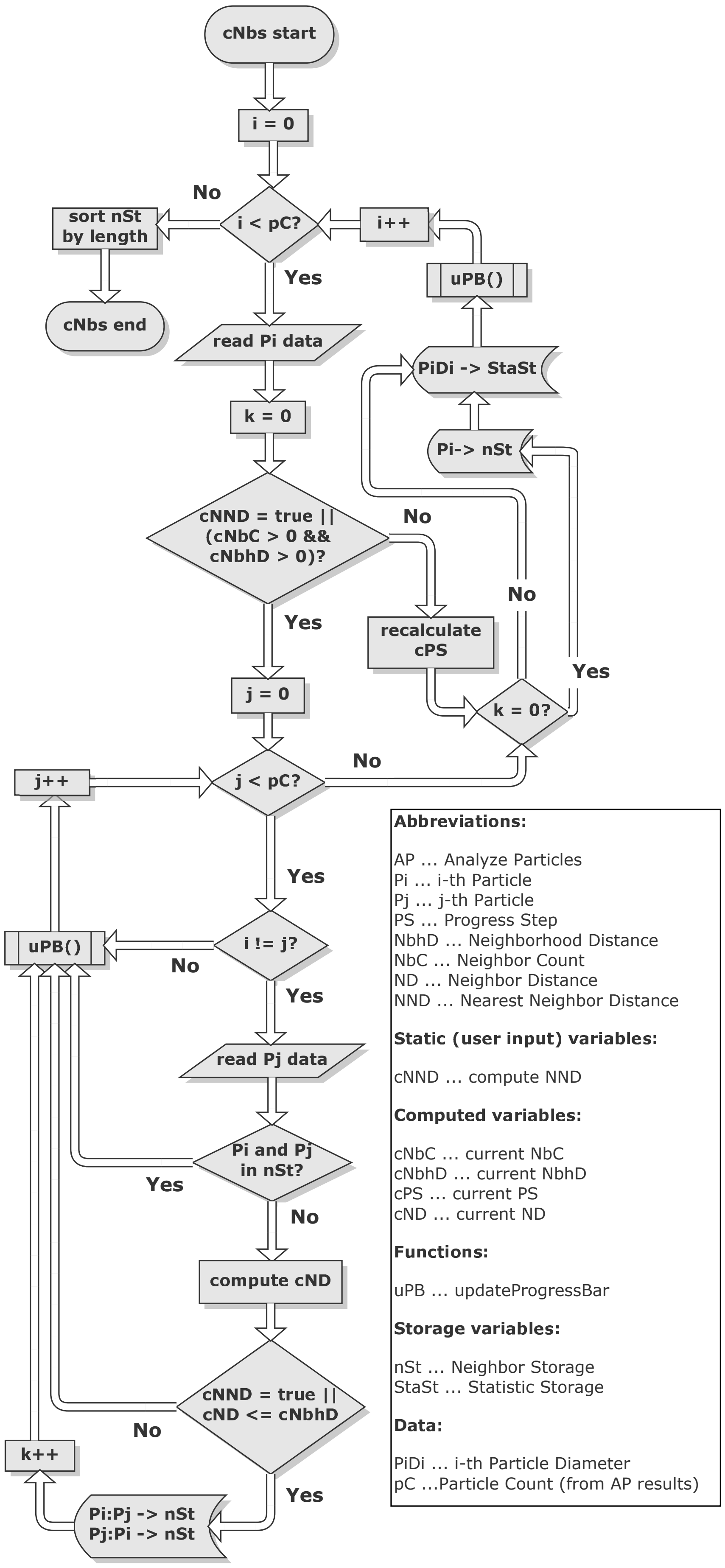
Storage variables:

NNDSt ... NND Storage
dSt ... Distance Storage
StaSt ... Statistic Storage

Data:

pC ...Particle Count (from AP results)

Figure 2.8: Main flowchart



Abbreviations:

AP ... Analyze Particles
Pi ... i-th Particle
Pj ... j-th Particle
PS ... Progress Step
NbhD ... Neighborhood Distance
NbC ... Neighbor Count
ND ... Neighbor Distance
NND ... Nearest Neighbor Distance

Static (user input) variables:

cNND ... compute NND

Computed variables:

cNbC ... current NbC
cNbhD ... current NbhD
cPS ... current PS
cND ... current ND

Functions:

uPB ... updateProgressBar

Storage variables:

nSt ... Neighbor Storage
StaSt ... Statistic Storage

Data:

PiDi ... i-th Particle Diameter
pC ...Particle Count (from AP results)

Figure 2.9: computeNeighbors flowchart

2.5.3.3 ComputeFinalNeighbors function

The final neighbor computation, illustrated in Fig. 2.12, determines the ‘real’ particle neighbors by incorporating the current neighbor count as well as the overlapping algorithm. Again two nested for loops are used to iterate over all particle combinations. When using the neighbor count limitation, it is applied within the second loop to check whether each particle has enough free ‘neighbor slots’ available to proceed. Afterwards an angle computation from particle one (P_1) to particle two (P_2) is carried out which is illustrated in Fig. 2.10. In this graphic d corresponds to the central distance between the origins of the

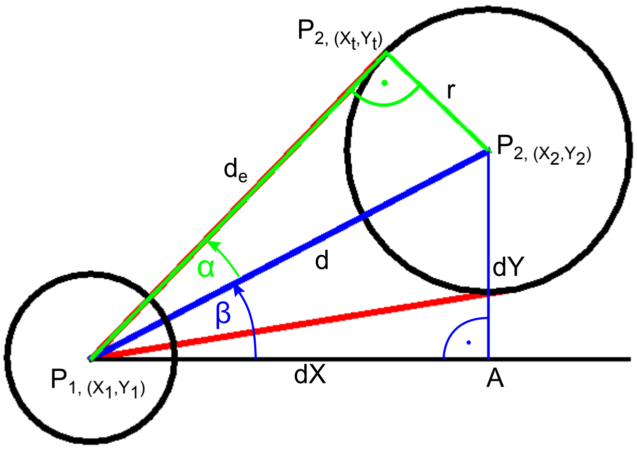


Figure 2.10: Angle measurement between two adjacent particles.

two particles which is calculated by Eq. (2.5), r is the radius of P_2 and d_e represents the length of the tangent line starting from P_1’s origin to P_2’s edge at r. In addition, dX and dY are obtained through Eq. (2.3) and Eq. (2.4), respectively. As can be seen, the whole assembly forms two right angled triangles, one (blue) with the vertices $P_{1,\mathrm{origin}}$, $P_{2,\mathrm{origin}}$ and A as well as a second (green) with the corner points $P_{1,\mathrm{origin}}$, $P_{2,\mathrm{origin}}$ and $P_{2,(\mathrm{X_t},\mathrm{Y_t})}$. The missing parameters β, α and d_e are calculated as follows:

$$\beta = \arctan\left(\frac{dY}{dX}\right) \tag{2.16}$$

$$d_\mathrm{e} = \sqrt{d^2 - r^2} \tag{2.17}$$

$$\alpha = \arctan\left(\frac{r}{d_\mathrm{e}}\right) \tag{2.18}$$

To cope with the fact that the *arctangent* function (*arctan*) in Eq. (2.16) is only defined in the range of $-\frac{\pi}{2}$ to $\frac{\pi}{2}$, but β must span the whole circle range from 0 to 2π, Java comes with a build-in function called *atan2* which internally considers all possible combinations of the input variables of dX and dY (plus, minus or equal to zero) and returns the appropriate angle in the range of $-\pi$ to π.

With these data computed, it is now possible to determine (from one particles ‘point of view') whether an alleged neighbor particle is located behind another one or not. In the standard case, even the smallest overlap is forbidden, otherwise every particle in the vicinity would be registered as a valid neighbor (within the given boundaries NbR and NbhD), even if this particle is behind another one. For such a case, an *overlap* option was implemented, that enables the user to specify an overlap percentage value at which a rear particle is still identified as a valid neighbor. The resulting behaviour is shown in Fig. 2.11. All

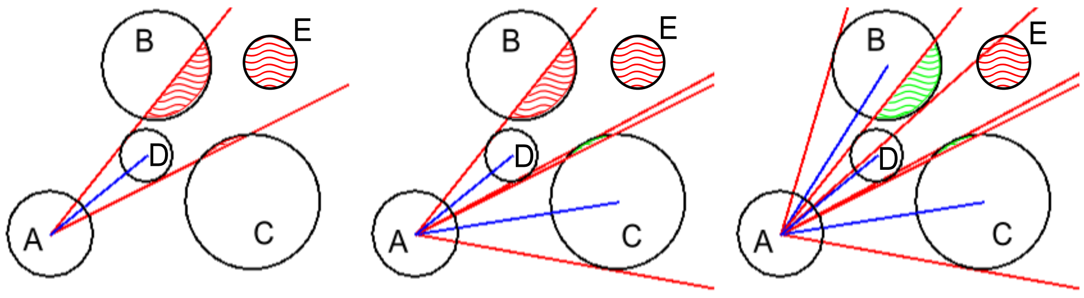
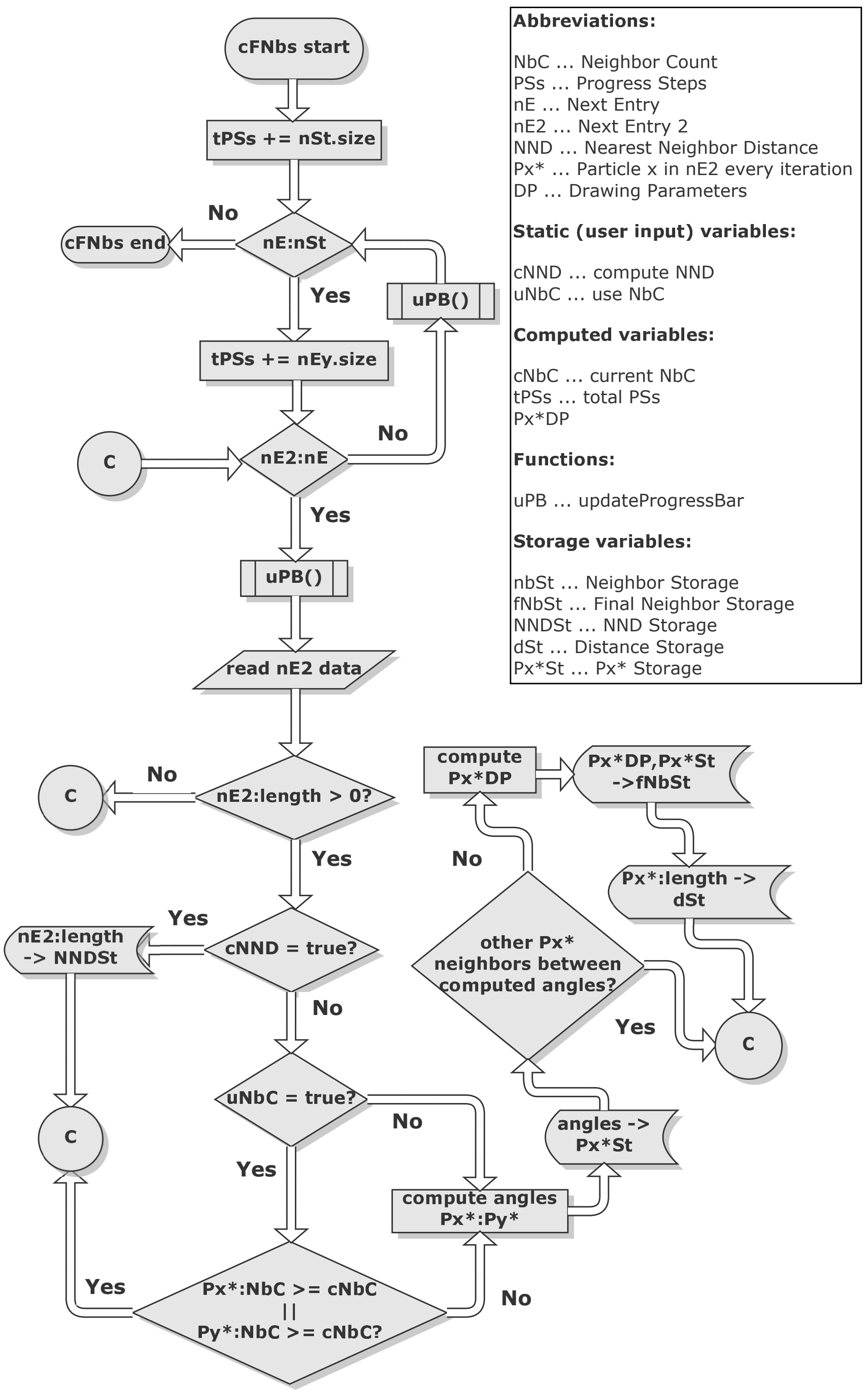


Figure 2.11: Angle overlap illustration

three images are described from particle A's point of view. The image on the left hand side is an example if no particle overlap is allowed. The red shaded areas (inserted manually after the actual computation for visualization purposes) mark the overlap regions caused by the smaller particle D in front of A. Therefore, the particles B, E and C are not recognised as valid neighbors to A. Although this behaviour seems correct with regards to particle E, B and especially C should most likely be considered as valid neighbors. The middle image shows the same particle configuration but with an allowed overlap value of $25\ \%$. This means, that $25\ \%$ of the foremost particles area (in this case D) may be occupied by rear particles on either side of D ('either side' refers to each semi-circle if the full circle for example of D is cut in half by the blue line). The percentage value only applies to particles that are partly overlapping the foremost one. If a particle is located within both red boundary lines (for example E) it gets neglected despite any overlap allowance. As can be seen in the middle image, due to the $25\ \%$ overlap particle C gets registered as a neighbor to A. Finally, the rightmost image presents the overlap behaviour at full range ($50\ \%$). In this case, B is also recognised besides C and of course D.
After the angle computation, a conditional check is performed to clarify whether the two particles are considered neighbors or not. If this check confirms their neighborship, all relevant data are calculated and stored accordingly for later usage. This concludes the *computeFinalNeighbors* function which, after all iterations, returns to the main program (Sec. 2.5.3.1).



Abbreviations:

NbC ... Neighbor Count
PSs ... Progress Steps
nE ... Next Entry
nE2 ... Next Entry 2
NND ... Nearest Neighbor Distance
Px* ... Particle x in nE2 every iteration
DP ... Drawing Parameters

Static (user input) variables:

cNND ... compute NND
uNbC ... use NbC

Computed variables:

cNbC ... current NbC
tPSs ... total PSs
Px*DP

Functions:

uPB ... updateProgressBar

Storage variables:

nbSt ... Neighbor Storage
fNbSt ... Final Neighbor Storage
NNDSt ... NND Storage
dSt ... Distance Storage
Px*St ... Px* Storage

Figure 2.12: computeFinalNeighbors flowchart

3 Results

3.1 Plugin demonstration

3.1.1 Basic plugin output

Model images are shown in Fig. 3.1 - 3.3 to illustrate the whole plugin functionality. The picture itself is named *blobs.gif* and is a sample image from ImageJ which was binarized and slightly modified for this purpose. The black circle which is present in all images represents the used NbhD.

3.1.2 Sample flow on analysing a TEM-image

Figure 3.4 - 3.6 show an exemplary workflow from the original to the final overlaid image. The individual steps are described in the figure captions.

3.2 Microstructural parameters for numerical modeling

3.2.1 Collagen fibril associated parameters

Figure 3.7 and Fig. 3.8 illustrate the interfibrillar mean distance behaviour with respect to the neighborhood distance, with and without fibrillar overlap. The measured diameters for the unloaded porcine tissue samples is presented in Fig. 3.9. The nearest neighbor distance distribution is shown in Fig. 3.10. Figure 3.12 illustrates the interfibrillar distance difference regarding the neighbor calculation when using the *overlap* option.

3.2.2 Proteoglycan associated parameters

Figure 3.13 shows the measured proteoglycan orientations from the human tissue sample.

3.3 Plugin performance

Figure 3.14 illustrates the plugin performance as execution time over various neighborhood distances. Additionally different allowed neighbor values are shown for comparison purposes. Furthermore, Fig. 3.15 - 3.16 demonstrate the overlap and the fibril count as performance influences.

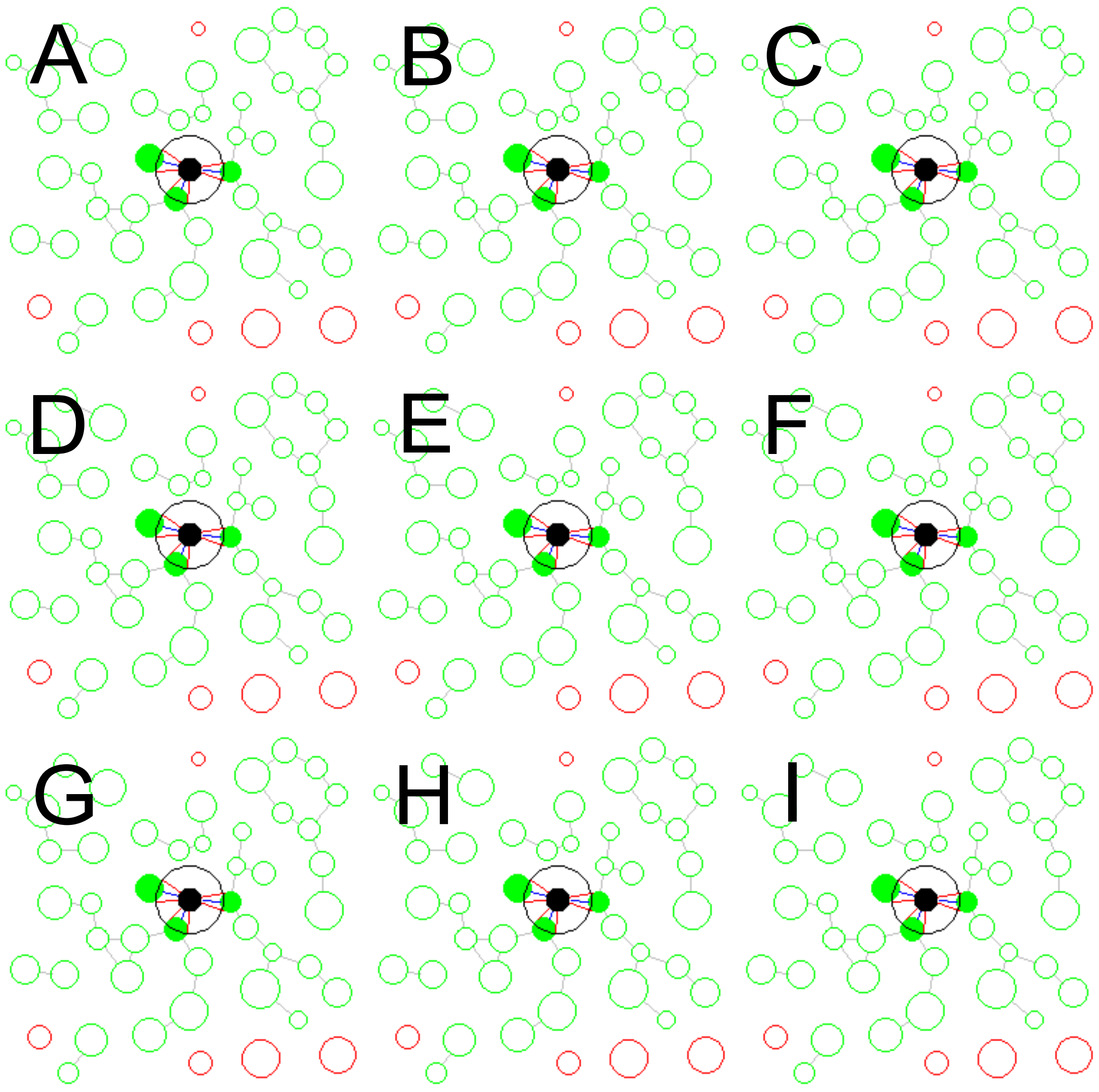


Figure 3.1: All images from $A - I$ show the plugin behaviour at a fixed NbhD of $15\ px$ but with different NbR and overlap values. Images in the columns $A - G$, $B - H$ and $C - I$ were computed for a maximum NbC of 3, 6 and no limit, respectively. For each of the rows $A - C$, $D - F$ and $G - I$ different overlap values $0\ \%$, $25\ \%$ and $50\ \%$ were used, respectively.

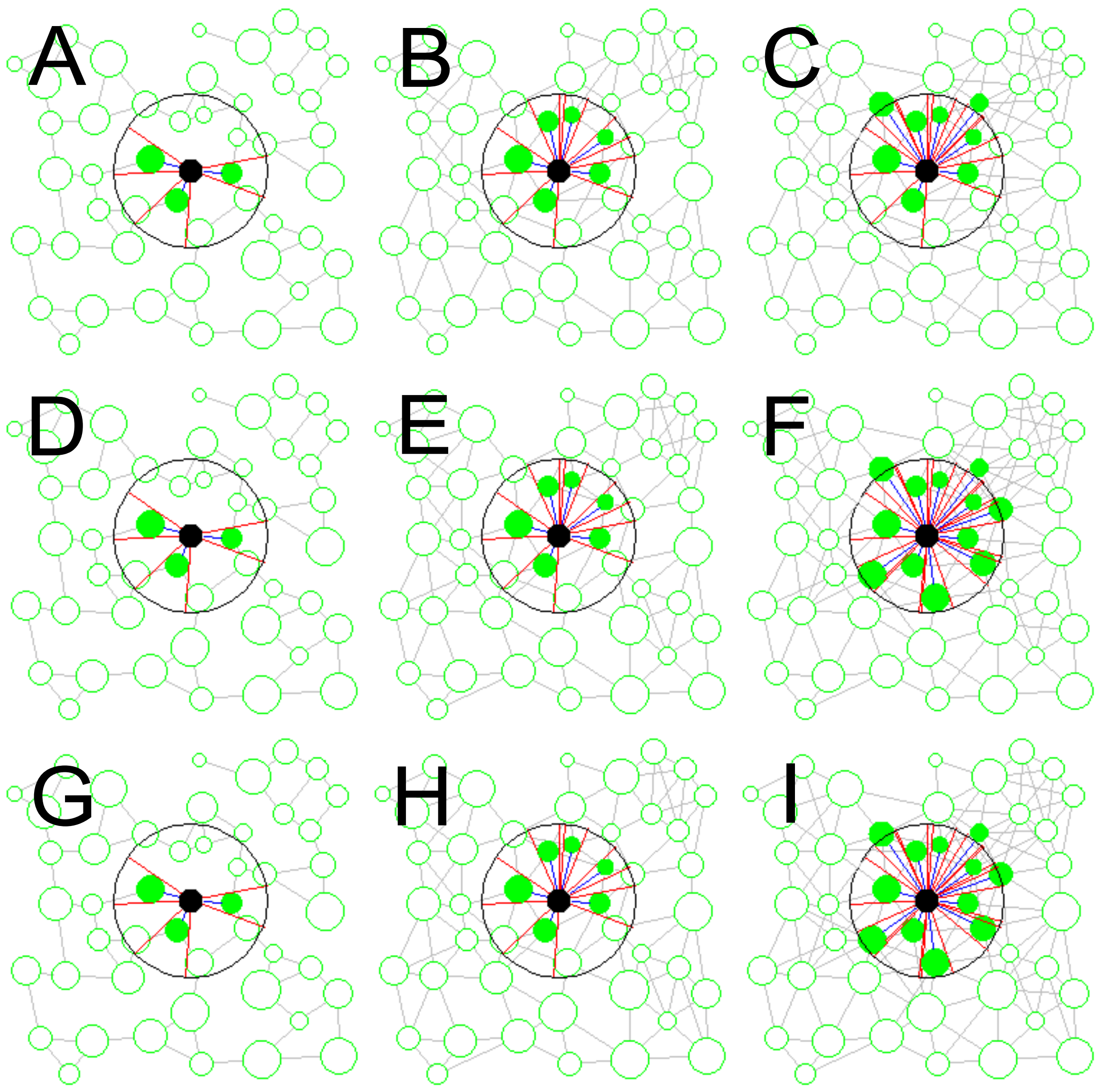


Figure 3.2: The same structure regarding the rows and columns (as in Fig. 3.1) applies here as well. The only difference is the NbhD of $45\,px$. Further details are presented in the continuous text.

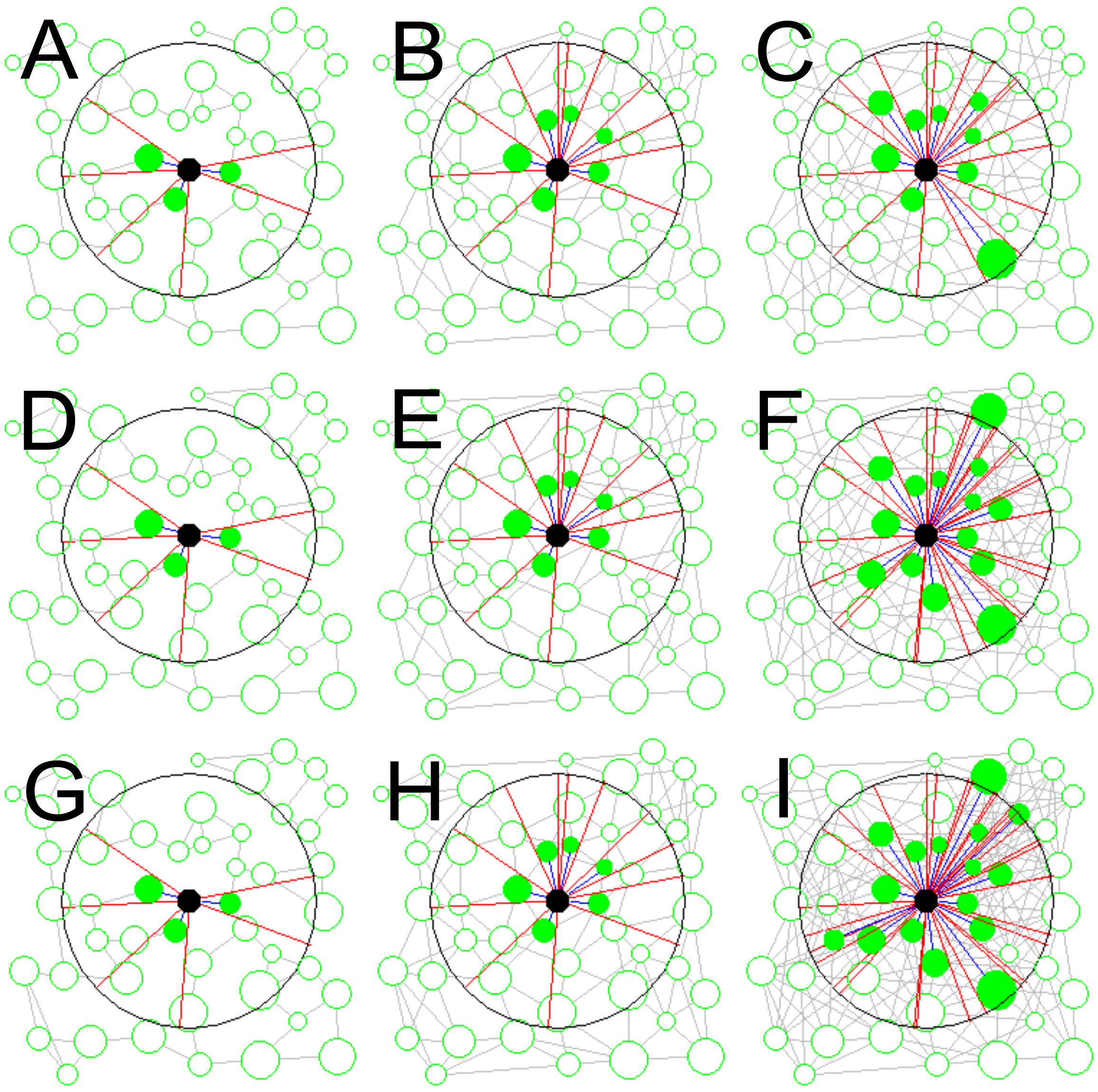


Figure 3.3: Image composition is the same as in Fig. 3.1 but with a NbhD of $80\ px$. Further informations are discussed in the continuous text.

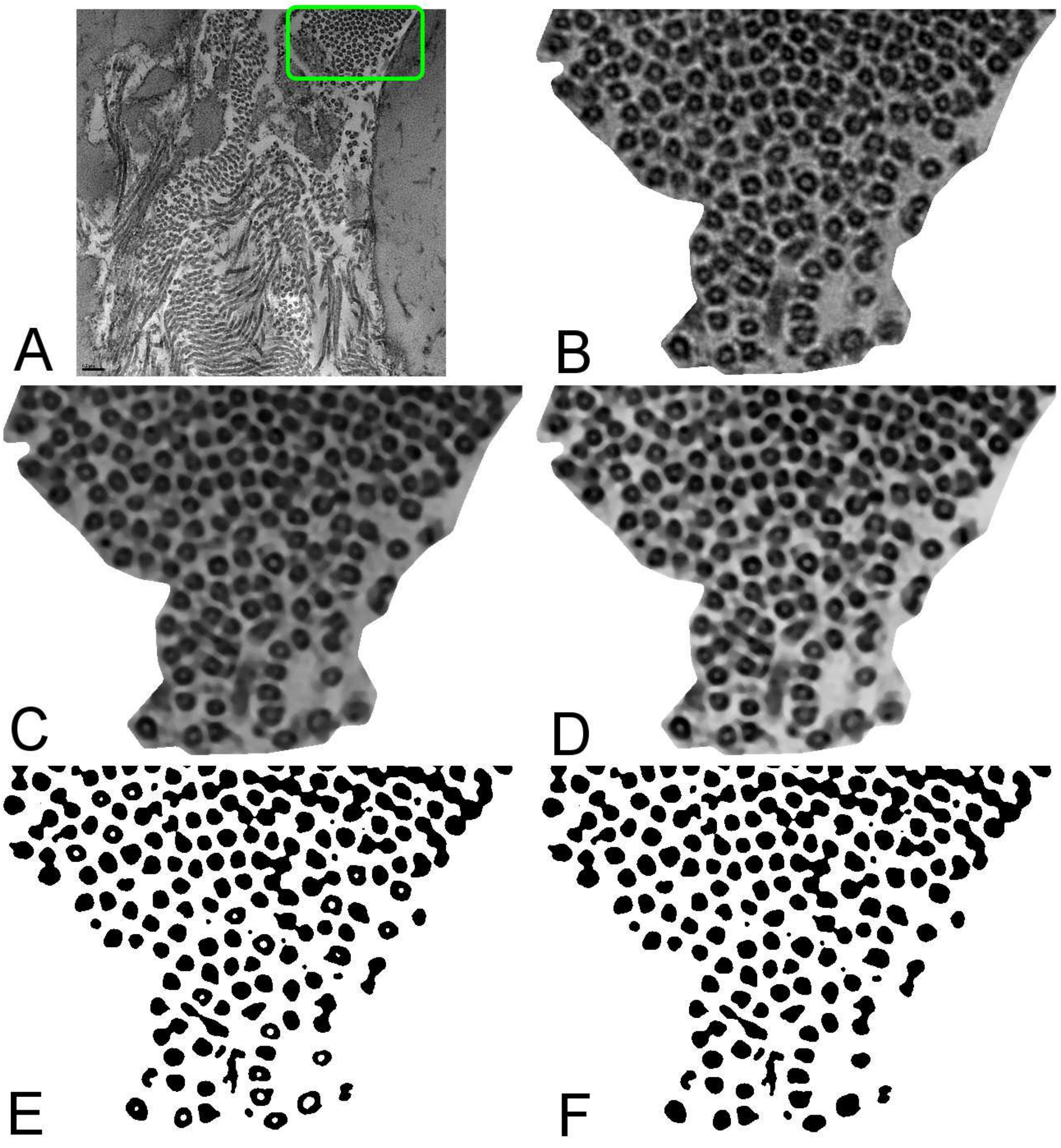
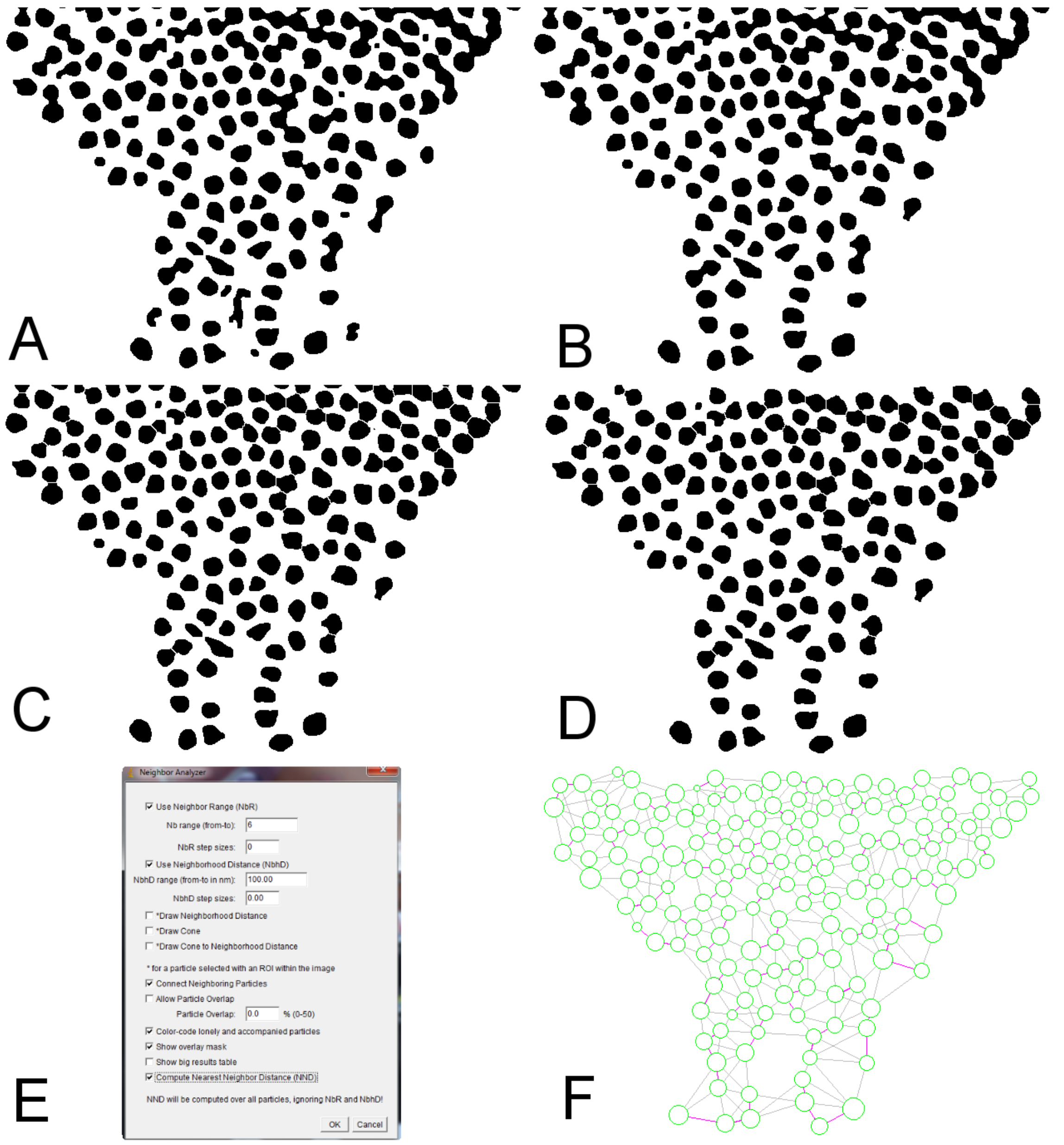


Figure 3.4: *A* shows the original image, where the region of interest was chosen to be in the top right corner (green selection). The ROI was selected with the *polygon selection tool* followed by *Edit -> Clear Outside*, which resulted in *B*. After this step, the image was saved and duplicated. (*C*) To reduce noise, a median filter was chosen with a $5\ px$ radius. In *D* the contrast was enhanced by performing *histogram equalization*. Now the threshold was set manually in order to get the best results (E) followed by the binary operation *Fill Holes* (*F*). Continued at Fig. 3.5.



☑ Use Neighbor Range (NbR)
Nb range (from-to): 6
NbR step sizes: 0
☑ Use Neighborhood Distance (NbhD)
NbhD range (from-to in nm): 100.00
NbhD step sizes: 0.00
☐ *Draw Neighborhood Distance
☐ *Draw Cone
☐ *Draw Cone to Neighborhood Distance
* for a particle selected with an ROI within the image
☑ Connect Neighboring Particles
☐ Allow Particle Overlap
Particle Overlap: 0.0 % (0-50)
☑ Color-code lonely and accompanied particles
☑ Show overlay mask
☐ Show big results table
☑ Compute Nearest Neighbor Distance (NND)
NND will be computed over all particles, ignoring NbR and NbhD!

Figure 3.5: Continued from Fig. 3.4. In *A* the binary operation *Open* with 2 iterations was applied. Note the disappearance of the small particles in the middle of the image compared to Fig. 3.4*F*. Next, in comparison with the original image, some bigger 'outliers' were removed manually using the *freehand selection tool* in conjunction with *Edit -> Clear* (*B*). In *C* the *Watershed* operation was performed to separate touching particles (taking a closer look, small gaps can be seen in the top right corner compared to *B*). Now *Analyze Particles* was executed (after setting the required measurements and the *Set Scale* parameter to $0.123\ \frac{px}{nm}$). Notice that in *D* the 'edge' particles are gone, since they are ignored in the analysis. Furthermore, to eliminate any remaining unwanted particles, the *size* parameter was set to $10 - infinity$. Finally, the new plugin was executed with $100\ nm$ NbhD and maximum 6 neighbors (*E*). The resulting mask is shown in *F*.

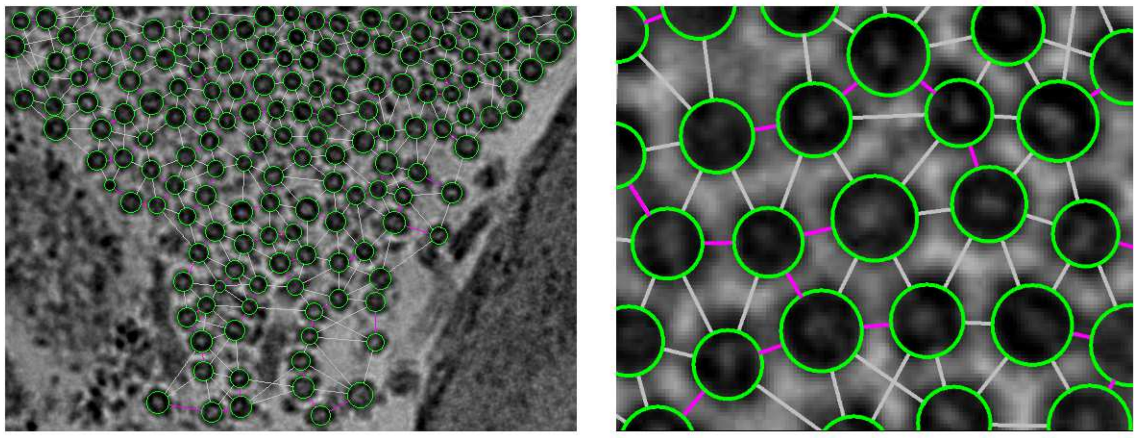


Figure 3.6: Both images show the resulting mask overlaid over the original image. Because this *overlay* is a vector graphic, it does not get pixelated when zoomed in, as can be seen in the right image. Additionally, it is now easy to verify the goodness of the processing procedure, through comparison of the overlay circles and the fibrils in the original image. The purple colored lines represent the nearest neighbor distances of each particle.

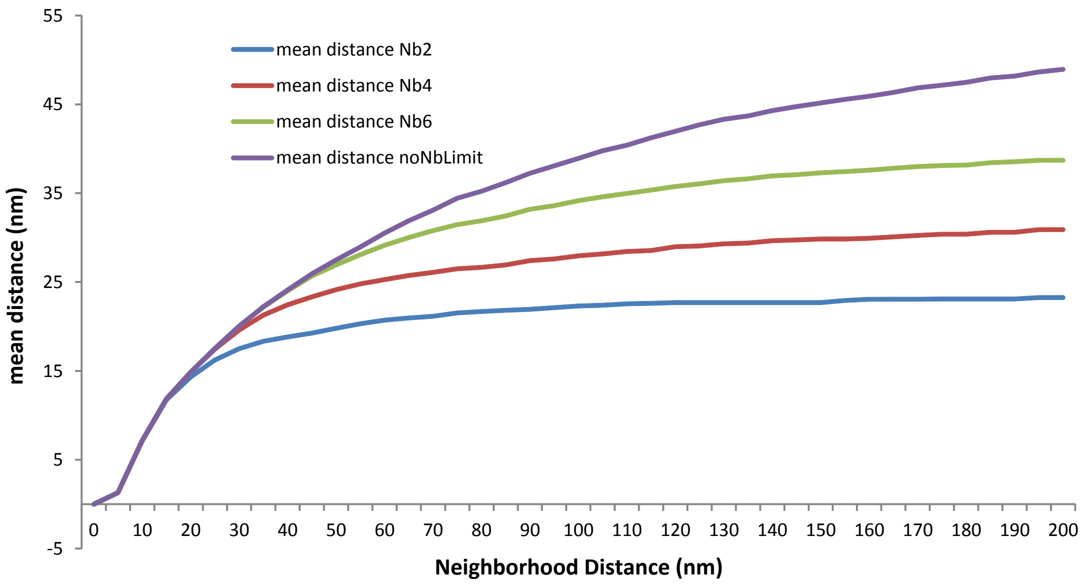


Figure 3.7: The figure shows the influence of different allowed neighbor values to the mean interfibrillar distance with respect to the neighborhood distance. The blue line shows the mean distance for maximal two neighbors, red for four, green illustrates six allowed neighbors while purple has no neighbor limit.

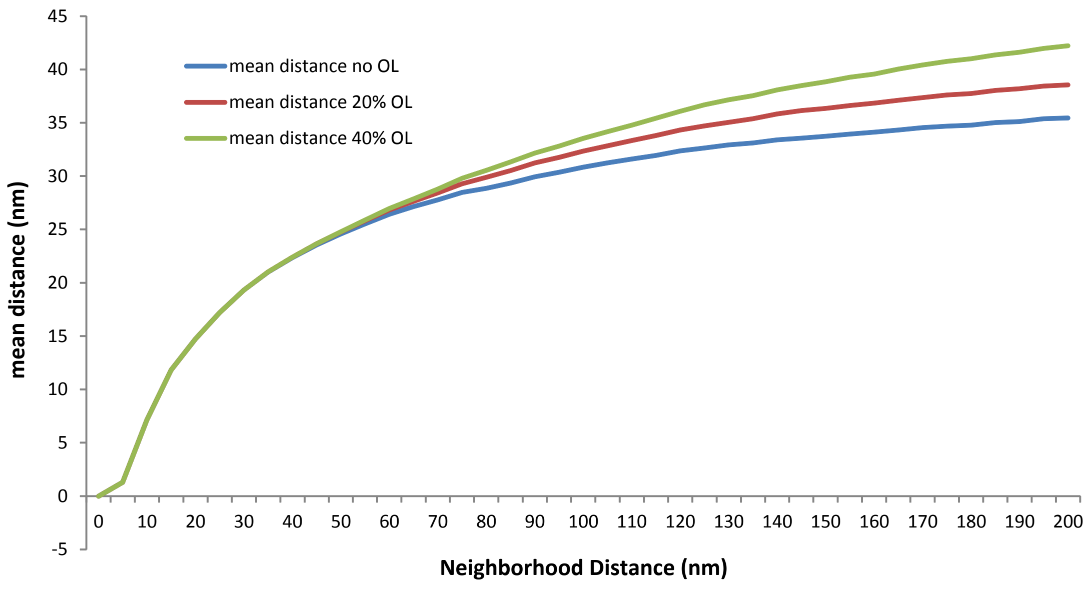


Figure 3.8: The effect of different overlap (OL) values with respect to the mean interfibrillar distance over the NbhD is shown.

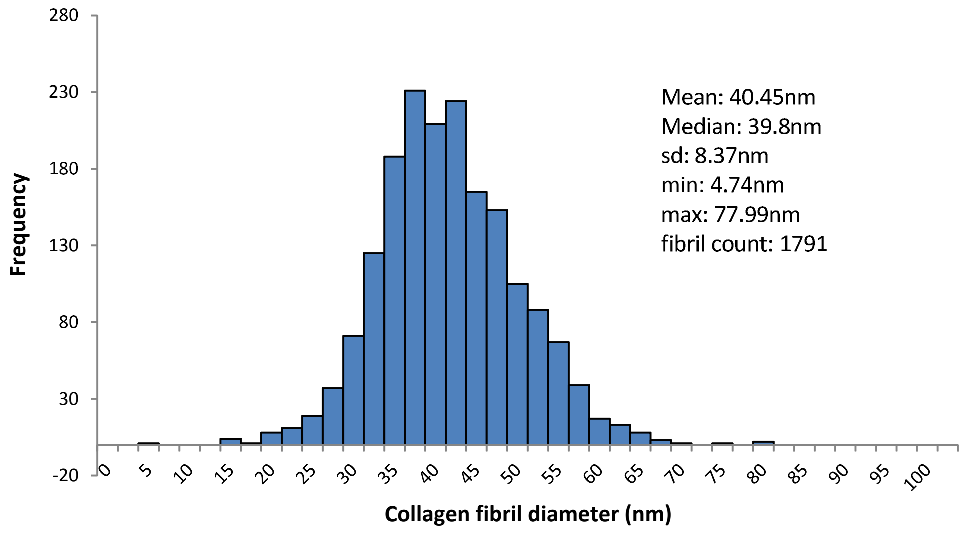


Figure 3.9: The histogram shows the measurements of 1791 fibril diameters of the unloaded porcine tissue sample extracted from TEM images.

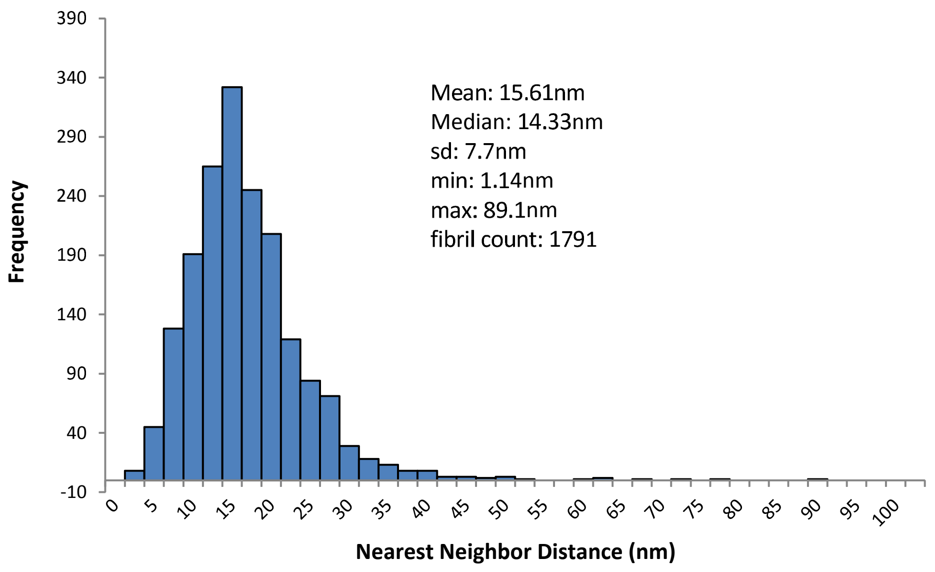


Figure 3.10: The figure shows the nearest neighbor distance distribution of the porcine tissue sample extracted from TEM images.

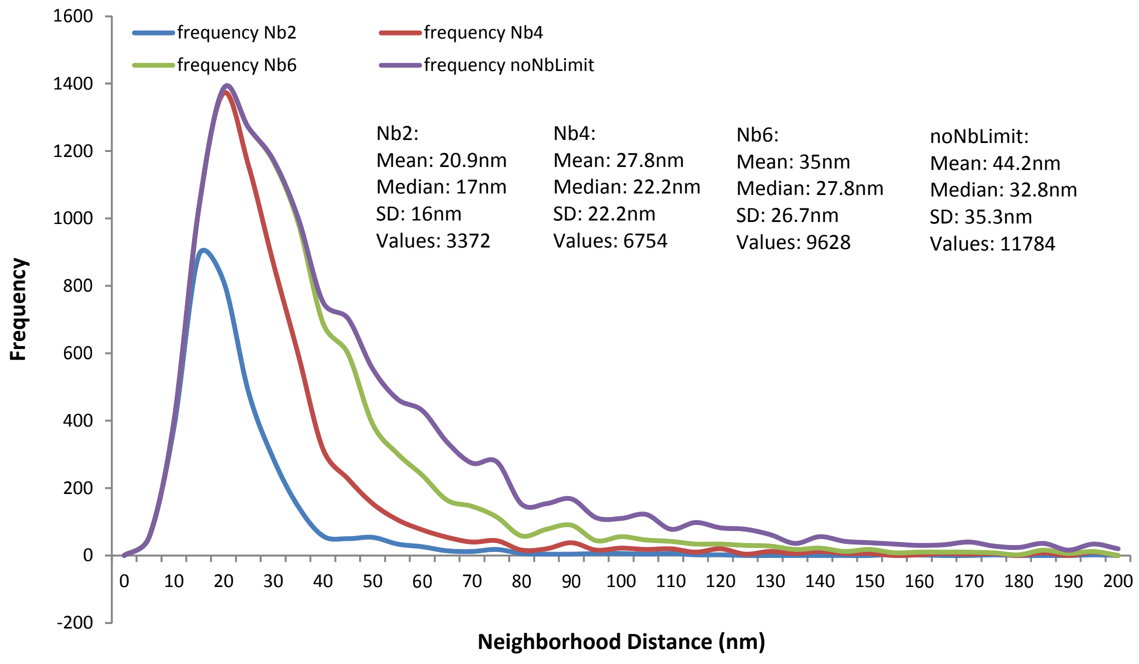


Figure 3.11: The figure shows a comparison between interfibrillar distance frequencies with alternating allowed neighbor quantities. The blue line represents an allowed neighbor count of two, the red line four, green stands for 6 neighbors and purple has no neighbor limit.

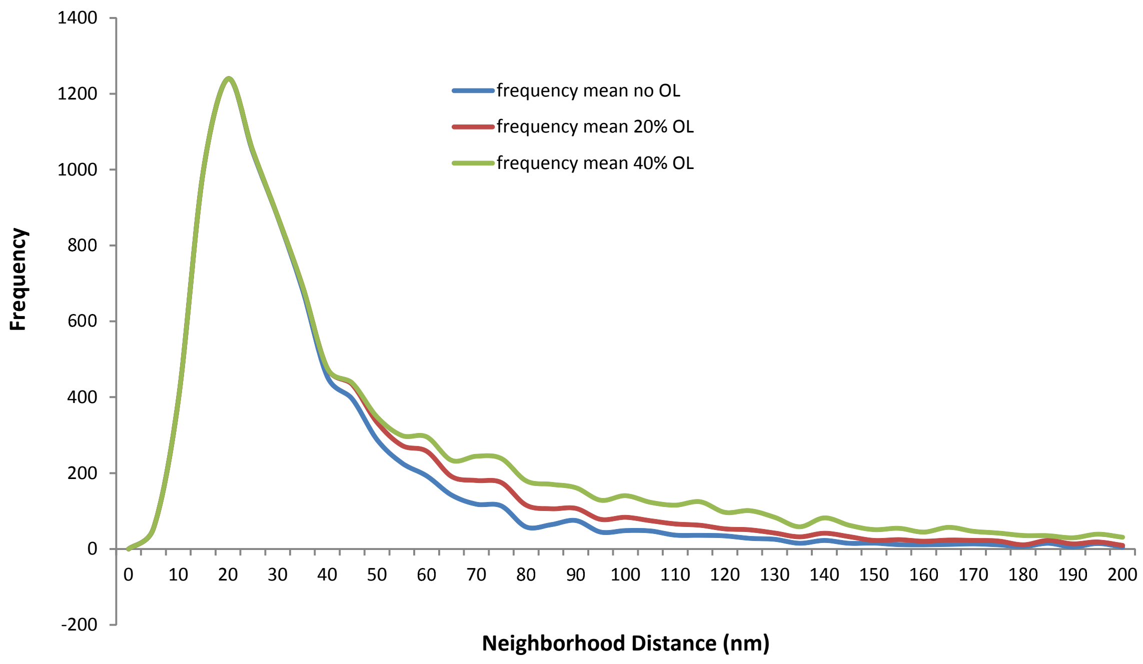


Figure 3.12: The average interfibrillar distance for each setting is shown with blue representing no overlap, red $20\ \%$ overlap and green $40\ \%$ overlap.

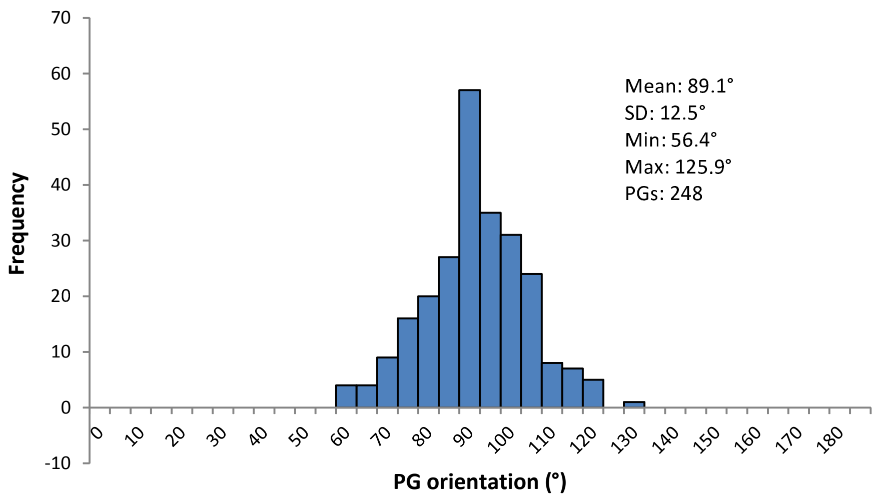


Figure 3.13: Distribution of proteoglycan orientation from the human tissue sample. $90\ ^\circ$ denotes the perpendicular orientation with regards to a collagen fibril.

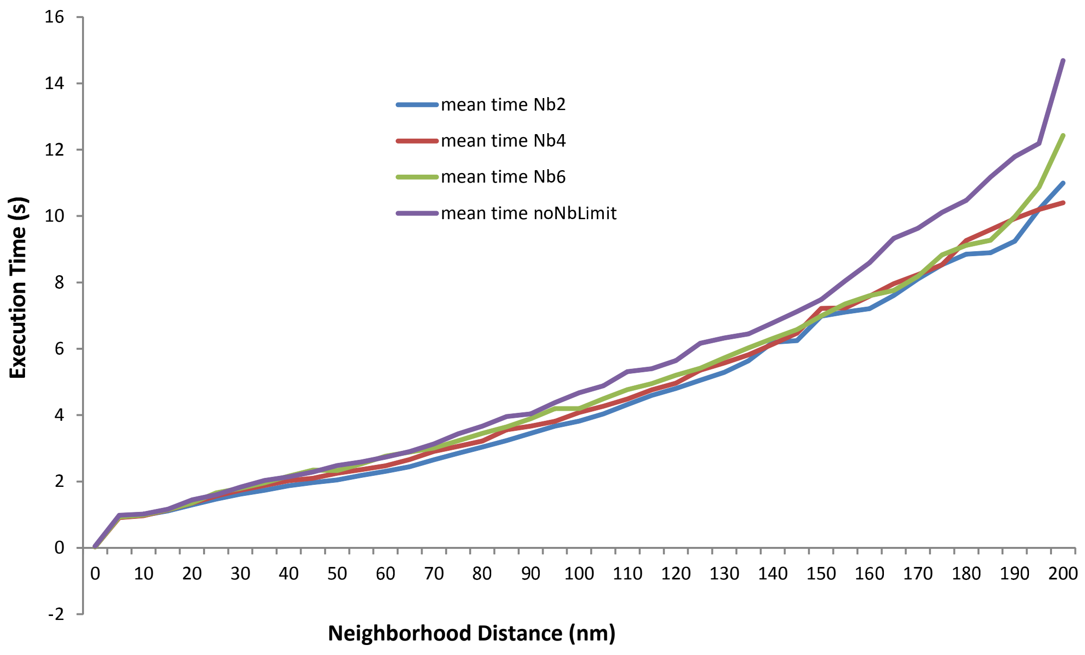


Figure 3.14: Execution time vs NbhD with different maximum neighbors. The blue line shows the behaviour for two neighbors, red for four, green represents six allowed neighbors and purple has no neighbor limitation.

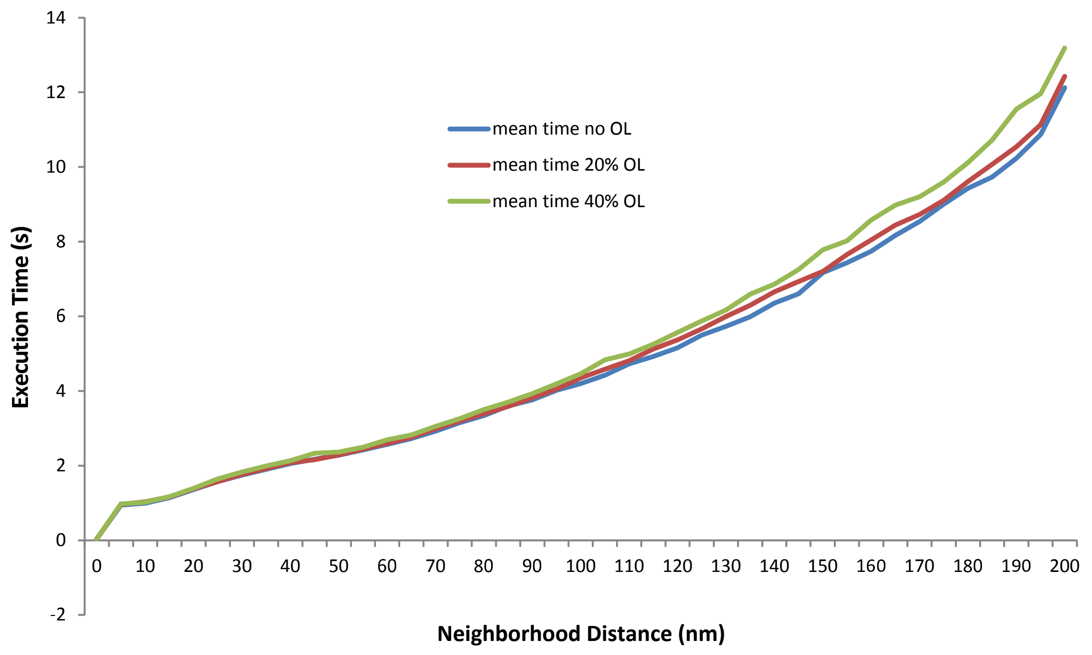


Figure 3.15: Execution time vs NbhD using different overlap settings. The blue line is without overlap whereas red and green are $20\ \%$ and $40\ \%$ overlap, respectively.

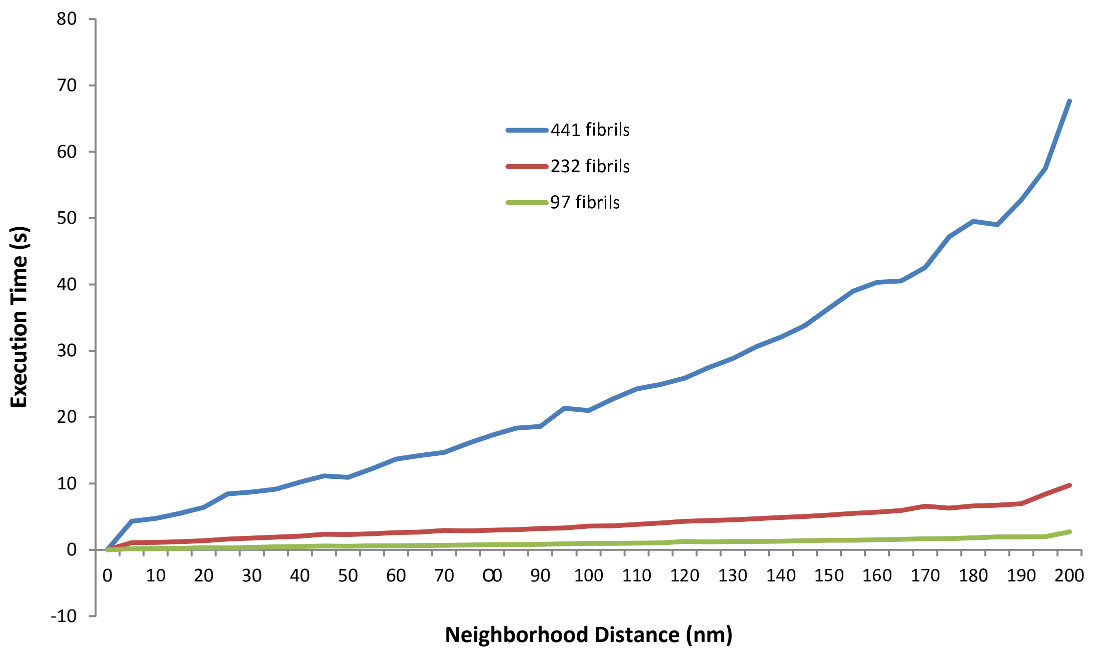


Figure 3.16: Execution time vs NbhD with regards to the measured fibril or particle count in the image.

4 Discussion

4.1 Notes regarding the plugin and add-ons

With regards to the fact that this plugin development was the first encounter in the field of Java programming and due to the big variety of different data types, storage variables, classes etc. available it is certain that performance improvements can be made in future versions. Many different add-on ideas arose while the development of this plugin, which could be implemented in future releases such as:

- Automation process starting from the binarised image (automatically call *Analyze Particles* with required options)
- Automatic computation and graph creation of the spatial distribution from the particles of interest
- Automatic curve fitting of the *Neighborhood Distance* to *mean / median distance* relation
- Automatic curve fitting of a distance histogram per *Neighborhood distance* and *Neighbor Range* cycle
- In case of many open tables or figures, a standard ‘save to...’ path may be given so that everything can get saved automatically
- Implementation of an automatic proteoglycan orientation algorithm

4.2 Plugin model output

At the lowest NbhD value of $15\ px$, shown in Fig. 3.1, no difference can be seen between the maximum Neighbor Count (NbC) and different overlap values due to the very tight NbhD. Compared to Fig. 3.2, it can be seen that in the case of a maximum NbC of three the interconnections between the particles don’t change because every particle is already saturated or is not allowed to have additional neighbors. For example in picture A at the top, in the middle a small particle is located, which only has one neighbor because all potential ‘partners’ in its vicinity are already saturated with closer particles. In the other cases, it is demonstrated that the overall interconnection number increases with growing neighbor counts and overlap values.

As for the remaining example, Fig.. 3.3 shows an even denser interconnection network, especially when no NbR limit is present and the overlap's value is $50\ \%$. Again, in the case of a maximum NbC of three, only minor changes can be seen (for example in A the small particle at the top, in the middle gets a second neighbor at a overlap value of $25\ \%$).

4.3 Plugin performance

The plugin was developed and tested on a desktop computer, with the following specifications:

- CPU: AMD Opteron 165 Dual Core @ $1.8\ GHz$
- Working memory: $4\ Gb$ DDR Ram
- Windows 7® 32-bit Edition

This hardware is nowadays a rather low-end machine, which might be a factor explaining the long plugin execution times, although, as was already stated previously, the performance within the source code lacks of optimization. Another issue that occurred several times, was the lack of available working memory for ImageJ. At least in a 32-Bit environment, the standard memory allocation to the program was set to $640\ Mb$ which can be altered to whatever amount is necessary as long as it stays within specific boundaries. The maximum allowed memory allocation was in the current hardware configuration $1600\ Mb$. One may think that this amount of memory should be more than sufficient to provide a stable plug runtime but as was seen the program ran several times out of memory and stopped the running process. This was especially often observed when both parameters, the *Number of Neighbors* in conjunction with the *Neighborhood Distance Range* were used to analyze large amounts of particles (> 400) at once. Additionally any running background processes or programs may reduce the available process power or memory therefore it is advices to close any resource consuming applications before running this plugin for analysing many particles at once. It is further important to note that it is not applicable to set the memory allocation in advance to an arbitrary high number, because under the circumstance that more than the standard memory value is applied to ImageJ, it may happen (as it did during the testing phase) that ImageJ itself won't start again after closure. To bring it back to life, ImageJ has be loaded at its default memory configuration which can be achieved by deleting the config file *ImageJ.cfg* in the programs root directory. Therein the information about the altered memory allocation is stored. Therefore a quick note shall be given that these two parameters should be used with caution. An iterative process is suggested to gain proper information value vs. requiered computation time. If the iterations are set too high, it may be the case that the program suddenly stops because of memory issues and all calculated data to this stage would be lost. It is highly recommended to optimize the given source code to overcome this issue. The code itself should provide a good basis for further improvements and said add-ons although it is not yet object oriented written but the idea

behind the algorithm should be reasonable.
With regards to the presented performance results in Fig. 3.14 - 3.16 it can be shown, that the neighbor range option as well as the overlap function (Fig. 3.14 and Fig. 3.15) do not contribute significantly to the overall process time. The actual particle or fibril count, which are analyzed at once, has the biggest influence on the overall computation time, as can be seen in Fig. 3.16. Therefore, it is suggested to keep the particle count reasonably low so it won't act as a bottleneck.

4.4 Model parameter acquisition

4.4.1 Collagen fibrils

With regards to the mean interfibrillar distance over a specific neighborhood distance range, Fig. 3.7 illustrates that, depending on a given neighbor range, the mean distance reaches a plateau phase, were no more additional neighbors will be found, either due to the maximal allowed neighor restriction or due to fibrillar overlap.
In Fig. 3.8 it can be seen, that when the overlap option is used, more neighbors (even those farther away) will be found, hence the mean distance is higher than with lower or no overlap.

4.4.2 Proteoglycan orientation

In Fig. 3.13 it can be seen, that the majority of the proteoglycans is oriented perpendicularly to the collagen fibrils, as is expected in the unloaded state.
The manual acquisition process can indeed be very time consuming thus an automated solution is highly recommended but rather difficult to implement (at least if the fibrils are not straightened). A possible approach to this goal arose out of the *Skeletonize* operation but was found too late to try and develop the required algorithm.

4.5 Limitations

As a matter of fact, the results obtained within this work should be handled with care as they are restricted to certain limitations and boundaries which can occur.

4.5.1 Tissue fixation and staining

A very crucial step in gathering relevant and 'true' information about the conformation and structure of living tissue or their constituents is the choice of a proper fixation method. The need for knowing the underlying mechanics of certain structures was mentioned several times and therefore it is essential to preserve fresh tissue as closely to its native (in vivo) state as possible. The environment in which tissue resides inside living organisms has a number of factors that are vital for their survival and for their proper task execution. Such factors are for example temperature, pressure, nutrition, oxygen supply and so forth which are highly regulated by complex control systems. In order to obtain valuable and representative microstructural parameters, it is very important to restrict alteration of the nano scaled structures after and due to tissue extraction as much as possible (i.e. fix the structural composition in place). Despite the high technology instruments that are available for scientific research, it cannot be said for certain that the structures seen on transmission electron microscope images (like the ones in this thesis) are represented exactly as if they were in their native environment. Another important note mentioning is that a shrinking effect was observed due to fixation. This fact must be considered when working with the obtained results. If the shrinking factor to a given fixation method or tissue type is known, it is easy to subtract the error from the obtained values. This correction process could be implemented as another feature to the plugin, as noted above. To extract specific features out of images, they need to be distinguishable from their surrounding (background, other filaments, particles etc.).Therefore different staining methods and agents are at disposal so that the desired structures have a higher contrast to their vicinity. Ideally the staining agents bind very specifically to their target but in reality this is not always the case. The observed and analyzed structures and filaments are nicely distinguishable by their geometrical shape for the human eye, but as was already mentioned it is not as easy for machines. Without carefully carried out tissue staining, it might be as good as impossible to determine valuable and meaningful parameters.

4.5.2 Image acquisition

Different imaging techniques are available, from light over fluorescence to electron microscopy or even completely different methods like magnet resonance imaging or computer tomography. Depending on the kind of images that are desired or the structures to be analyzed, the best visualization technique has to be chosen carefully. For this work images were obtained using transmission electron microscopy, which is a specific method amongst others in the field of electron microscopy. Due to the fact that this imaging technique interferes directly (physically) by high energetic electrons with the sample tissue, one must consider the possibility that the specimen or its fine structures might get damaged during the exposure time to the electron beam.

4.5.3 Image processing and analysis

The image processing part is, when carried out manually vulnerable to false subjective observations and bad decision making. Starting with smoothing or sharpening methods over histogram equalization, thresholding, binary operation etc. there are many steps on the road to the final analyzable image were failure may occur. A good deal of experience in this field of expertise is recommended to yield good and valuable results. Thus a study in the field of digital image processing was conducted to obtain the required theoretical background to not blindly carry out operations and hope fingers-crossed to deliver usable material. On the contrary, automatic algorithms may help to find better or ‘the best’ threshold value for a given segmentation problem but in the end these automatic approaches are based on distinct developed algorithms, all with their pros and cons. Image analysis is far more time consuming when performed manually, depending on the amount of information that needs to get extracted. Further depending on how narrow the error margin of measurements is, it could get very tricky, difficult and hardly possible to measure with pixel or subpixel accuracy by hand. Here automatic processes and algorithms are of great use, assumed that the logic behind the code was carefully thought out and written (relatively) performantly. As can be seen especially by the computational time results in Fig. 3.16 one may assume that the given premise of performant and well structured code writing was not as well implemented as might be possible. At last the created plugin may not only be helpful in the field of biomechanics but everywhere where such determinations are of use.

4.5.4 Out-of-plane deviations

The presented results were all gathered from two dimensional image planes which may be problematic due to the fact that the measures structures are apparent in three dimensional space. This circumstance can produce significant measurement errors, particularly when quantifying the proteoglycan orientations. This is because PGs can protrude away from their fibrils in all directions to adjacent fibrils. Therefore some may travel out of or come into the plane of sight (Liao and Vesely, 2007). The occurring measurement error should reduce when an considerable amount of proteoglycan angles is acquired. In spite of this fact, Liao and Vesely (2007) found a statistically significant trend regarding their PG orientation measurements.

Bibliography

B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, editors. *Molekularbiologie der Zelle*. Wiley-VCH Verlag GmbH&Co, 2004.

D. Balzani, S. Brinkhues, and G. A. Holzapfel. Constitutive framework for the modeling of damage in collagenous soft tissues with application to arterial walls. *Comput. Meth. Appl. Mech. Eng.*, 213-216:139–151, 2012.

D. E. Birk and E. Zycband. Assembly of the tendon extracellular matrix during development. *J. Anat.*, 184:457–463, 1994.

D. E. Birk, M. V. Nurminskaya, and E. I. Zycband. Collagen fibrillogenesis in situ: Fibril segments undergo post-depositional modifications resulting in linear and lateral growth during matrix development. *Dev. Dynam.*, 202:229–243, 1995.

D. E. Birk, R. A. Hahn, C. Y. Linsenmayer, and E. I. Zycband. Characterization of collagen fibril segments from chicken embryo cornea, dermis and tendon. *Matrix Biol.*, 15:111–118, 1996.

D. E. Birk, E. I. Zycband, and S. Woodruff. Collagen fibrillogenesis in situ: Fibril segments become long fibrils as the developing tendon matures. *Dev. Dynam.*, 208:291–298, 1997.

F. T. Bosman and I. Stamenkovic. Functional structure and composition of the extracellular matrix. *J. Pathol.*, 200:423–428, 2003.

N. J. Boudreau and P. L. Jones. Extracellular matrix and integrin signalling: the shape of things to come. *Biochem. J.*, 339:481–486, 1999.

W. Burger and M. J. Burge, editors. *Digitale Bildverarbeitung - Eine Einführung mit Java und ImageJ*. Springer-Verlag Berlin Heidelberg, 1st edition, 2005. URL `http://www.imagingbook.com`.

T. Ferreira and W. Rasband. *ImageJ User Guide*, ij 1.46r edition, 10 2012.

C. Frantz, K. M. Stewart, and V. M. Weaver. The extracellular matrix at a glance. *J. Cell Sci.*, 123:4195–4200, 2010.

P. Fratzl. Collagen: structure and mechanics, an introduction. In P. Fratzl, editor, *Collagen: Structure and Mechanics*, pages 1–13. Springer, 2008.

T. C. Gasser, R. W. Ogden, and G. A. Holzapfel. Hyperelastic modelling of arterial layers with distributed collagen fibre orientations. *J. R. Soc. Interface*, 3:15–35, 2006.

R. C. Gonzalez, R. E. Woods, and S. L. Eddins, editors. *Digital Image Processing*. Gatesmark Publishing, 2nd edition, 2009.

H. K. Graham, D. F. Holmes, R. B. Watson, and K. E. Kadler. Identification of collagen fibril fusion during vertebrate tendon morphogenesis. the process relies on unipolar fibrils and is regulated by collagen-proteogylcan interaction. *J. Mol. Biol.*, 295:891–902, 2000.

D. F. Holmes, H. K. Graham, J. A. Trotter, and K. E. Kadler. Stem/tem studies of collagen fibril assembly. *Micron*, 32:273–285, 2001.

G. A. Holzapfel. Lecture at Graz University of Technology on 'Mechanik biologischer Gewebe' - 450.001, Wintersemester 2012/13, Institute of Biomechanics. 2012.

G. A. Holzapfel and R. W. Ogden. Constitutive modelling of arteries. *Proc. R. Soc. Lond. A*, 466:1551–1597, 2010.

G. A. Holzapfel, T. C. Gasser, and R. W. Ogden. A new constitutive framework for arterial wall mechanics and a comparative study of material models. *J. Elasticity*, 61:1–48, 2000.

ImageJWiki. Faq: Which file formats are supported by imagej. `http://imagejdocu.tudor.lu/doku.php?id=faq:general:which_file_formats_are_supported_by_imagej`, 2010. [Online; accessed 10-February-2013].

K. E. Kadler, D. F. Holmes, J. A. Trotter, and J. A. Chapman. Collagen fibril formation. *Biochem. J.*, 316:1–11, 1996.

S. H. Kim, J. Turnbull, and S. Guimond. Extracellular matrix and cell signaling: the dynamic cooperation of integrin, proteoglycan and growth factor receptor. *J. Endocrinol.*, 209:139–151, 2011.

M. Kroon and G. A. Holzapfel. A new constitutive model for multi-layered collagenous tissues. *J. Biomech.*, 41:2766–2771, 2008.

G. Landini. Auto threshold. `http://fiji.sc/wiki/index.php/Auto_Threshold`, 2012a. [Online; accessed 15-February-2013].

G. Landini. Auto local threshold. `http://fiji.sc/wiki/index.php/Auto_Local_Threshold`, 2012b. [Online; accessed 15-February-2013].

J. Liao and I. Vesely. Skewness angle of interfibrillar proteoglycans increases with applied load on mitral valve chordae tendineae. *J. Biomech.*, 40:390–398, 2007.

National Institutes of Health. Imagej disclaimer. `http://rsb.info.nih.gov/ij/disclaimer.html`, 2004. [Online; accessed 10-February-2013].

World Health Organization. Cardiovascular diseases (cvds) fact sheet n317. `http://www.who.int/mediacentre/factsheets/fs317/en/index.html`, 2012. [Online; accessed 25-February-2013].

K. M. Meek. The cornea and sclera. In P. Fratzl, editor, *Collagen: Structure and Mechanics*, pages 359–396. Springer, 2008.

P. P. Provenzano and R. Jr. Vanderby. Collagen fibril morphology and organization: Implications for force transmission in ligament and tendon. *Matrix Biol.*, 25:71–84, 2006.

TW. Ridler and S. Calvard. Picture thresholding using an iterative selection method. *IEEE T. Systems Man Cybernet.*, 8:630–632, 1978.

C. A. Schneider, W. S. Rasband, and K. W. Eliceiri. NIH Image to ImageJ: 25 years of image analysis. *Nat Methods*, 9:671–675, 2012.

A. J. Schriefl, G. Zeindlinger, D. M. Pierce, P. Regitnig, and G. A. Holzapfel. Determination of the layer-specific distributed collagen fiber orientations in human thoracic and abdominal aortas and common iliac arteries. *J. R. Soc. Interface*, 9:1275–1286, 2012.

A. J. Schriefl, H. Wolinski, P. Regitnig, S. D. Kohlwein, and G. A. Holzapfel. An automated approach for three-dimensional quantification of fibrillar structures in optically cleared soft biological tissues. *J. R. Soc. Interface*, 10, 2013.

F. H. Silver, J. W. Freeman, and D. DeVore. Viscoelastic properties of human skin and processed dermis. *Skin Res. Technol.*, 7:18–23, 2001.

S. Vesentini, A. Redaelli, and F. M. Montevecchi. Estimation of the binding force of the collagen molecule-decorin core protein complex in collagen fibril. *J. Biomech.*, 38:433–443, 2005.

J. E. Wagenseil and R. P. Mecham. Vascular extracellular matrix and arterial mechanics. *Physiol. Rev.*, 89:957–989, 2009.

I. T. Weber, R. W. Harrison, and R. V. Iozzo. Model structure of decorin and implications for collagen fibrillogenesis. *J. Biol. Chem.*, 271(50):31767–31770, 1996.

T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Commun. ACM*, 27:236–239, 1984.

Statutory Declaration

I declare that I have authored this Thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material, which has been quoted by the relevant reference.

date	signature