Ankur Sharma

University of Southampton

FEEG3003 Individual Project

Image-based modelling of biological soft tissues – Extraction of microstructural features

ANKUR SHARMA

April 27, 2017

SUPERVISED BY DR. GEORGES LIMBERT

Word count: 9987

This report is submitted in partial fulfilment of the requirements for the MEng Mechanical Engineering degree programme, Faculty of Engineering and the Environment, University of Southampton

Declaration

I, Ankur Sharma declare that this thesis and the work presented in it are my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a degree at this University;
2. Where any part of this thesis has previously been submitted for any other qualification at the University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear
exactly what was done by others and what I have contributed myself;
7. Either none of this work has been published before submission, or parts of this work have been published as: NONE

Acknowledgments

I would like to thank my supervisor, Dr Georges Limbert for his guidance and support throughout this project. I would also like to thank the teaching staff at University of Southampton.

Table of Contents

Declaration 2
Acknowledgments 3
Abstract 6
1 Introduction 7
2 Objectives 7
3 Literature Review 8
3.1 Motivation for studying Collagen fibre orientation 8
3.2 Imaging techniques 9
3.3 Image Processing Concepts 9
3.3.1 Image Enhancement 9
3.3.2 Filtering 11
3.3.3 Segmentation 13
3.3.4 3D Reconstruction 19
3.4 Fibre orientation determination techniques 20
3.4.1 Gaussian orientation space filters 20
3.4.2 Fourier Transform based approaches 20
3.4.3 Hough Transform 22
3.4.4 Principal component analysis 23
3.4.5 Weighted orientation vector summation algorithm 24
3.4.6 Gradient orientation filters 25
3.4.7 Fibre tracking approaches – energy minimization algorithms 26
3.4.8 Chord Length transform 26
3.4.9 Summary 26
4 The Proposed Algorithm 27
4.1 Core algorithm 27
4.2 Regions of Interest (ROIs) 28
4.3 Speed performance and accuracy improvement 30
5 Implementation of other algorithms 33
5.1 Weighted Vector Summation method 33
5.2 Fourier Transform (Bandpass filter method) 36
5.3 Gradient Orientation filter 38
6 Discussion 40
6.1 2D algorithm analysis 40
6.2 3D algorithm analysis 43
6.2.1 Overall comparison 45
6.2.2 ROI window size effects 46
6.2.3 Fibre thickness effects 47

6.2.4 Fibre density effects 47
7 Example Application 49
8 Conclusion 51
9 Reference List 52
10 Appendix 55
10.1 Relevant code snippets 55
10.1.1 Moment of inertia 3D - Core algorithm 55
10.1.2 Moment of inertia 3D - ROI division 56
10.1.3 Weighted Vector Summation 2D – Algorithm 58
10.1.4 Weighted Vector Summation 3D – Algorithm 60
10.1.5 Fourier Analysis 2D – Algorithm 63
10.1.6 Gradient orientation filter 2D – Algorithm 66
10.1.7 Filtering and binarization – Core 67
10.2 2D Algorithms further analysis graphs 69

Abstract

The use of imaging techniques is abundant throughout the biomedical research industry. However, the obtained data must be processed to extract required information.

This report aims to study and develop a method for obtaining information about spatial orientation of microstructural features from image datasets. The report first summarizes the existing methods for microstructural feature identification and orientation calculation. It then discusses the development of an algorithm based on moment of inertia (implemented in Wolfram Mathematica) for finding spatial orientation of microstructural features. Initially, 2D images were considered and the algorithm optimized. It was then extended to 3D image datasets. Aspects such as filtering of noise and binarization were considered. The algorithm was compared to the other existing algorithms for speed performance, accuracy and applicability to different types of image datasets.

The initial application of the method is intended for biological soft tissue such as collagen fibres. However, the algorithm can be extended to other applications with some tweaks.

This document will report that the success in finding orientations of microstructural features accurately and quickly depends heavily on the dataset itself including microstructure feature density, shape, overlap, and noise levels. In general, the proposed algorithm performed better than other methods implemented for comparison such as weighted vector summation.

1 Introduction

Medical industry relies on imaging techniques for diagnosis and research. For example, Magnetic resonance imaging (MRI) is used extensively for neuroimaging and investigation of neurological cancers [1]. Another example is the use of Scanning Electron Microscopy (SEM) for imaging skin tissue for medical research and diagnosis.

These techniques can produce large datasets which may contain useful information. The data can be processed through manual examination. However automatic processing of the datasets is useful where the required information is not immediately clear, for e.g.- differences in orientation of collagen fibres in skin tissue scans.

2 Objectives

The overarching objective is to write an algorithm in Wolfram Mathematica for determination of fibre orientations from 3D image datasets. These 3D image datasets can ideally contain any fibrous structure, not just collagen fibrils. It is assumed that the images are 2 phases only (foreground/fibres and background/void). This report focuses on determining orientation data as a 3D vector field.

The 3D vector field representing the fibre orientations can then be passed onto external Finite Element Processing software for analysis of different mechanical and biological properties. This is outside the scope of this report.

The general outline of the steps involved in the process are shown in figure 2.1. The core of this report focuses on orientation algorithm. However, image acquisition, pre-processing and binarization are also discussed in some detail.

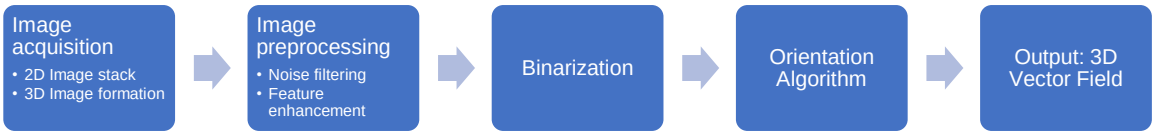


Figure 2.1 : The general outline of the steps involved in orientation determination of microstructural features.

3 Literature Review

3.1 Motivation for studying Collagen fibre orientation

Collagen is the most abundant protein in the human body. It is a large family of triple helical proteins that are found throughout the animal body and is important for a wide variety of functions including tissue scaffolding, cell adhesion, cell migration, cancer angiogenesis etc. However, it is most commonly known for being found in connective animal tissue such as skin, tendons, cartilage and bone where collagen fibrils are a major part of extracellular matrix (ECM) that provides structure to the tissue and increasing adhesiveness of cells [2].

Figure 3.1 shows an SEM image of elastic collagen fibres in skin [3]. Notice how the fibres are not all oriented in the same directions. Also, the scale of the fibres is in micrometres and requires advanced imaging techniques such as electron microscopy to capture details accurately.

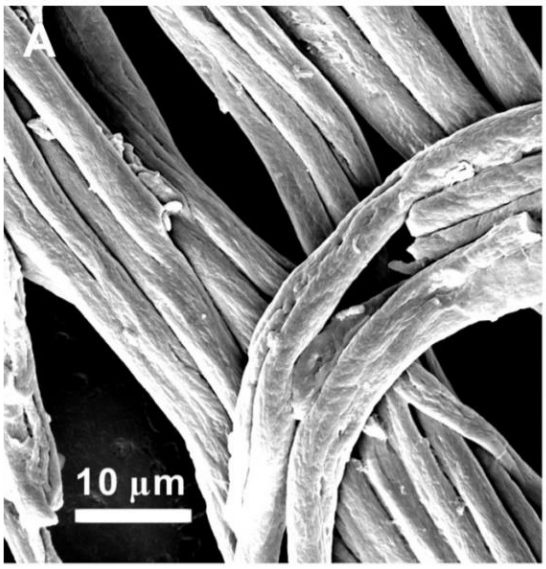


Figure 3.1: SEM image of elastic collagen fibres with diameters in the range of 3 to 5 µm [3]

The biological function of collagen lies predominantly in its mechanical properties. For e.g. - Youngs Modulus, peak tensile and shear stresses and viscoelastic properties [4]. These properties of collagen structures depend on the fibre orientations [5]. The collagen fibres are anisotropic even as single fibres, as they are formed of sub fibrils aligned along the fibre axis. [3] [4]

Therefore, information about collagen fibre orientations is essential in simulating mechanical behaviour of collagen rich tissues such as skin.

Apart from mechanical behaviour, determining collagen fibre orientations is a key aspect in many other areas.

Tissue type, age, health and hormonal status all affect the structure of collagen. Assessing collagen fibre orientation provides an indication of the functionality and performance of the structure [6]. For example – understanding of skin wrinkling with age can be predicted using changes in the fibre density and orientations [6].

Collagen fibres orientation also provides a potential biomarker for diagnosing diseased tissues, assessing response to treatments, localizing injuries or monitoring engineered

tissue development [7]. For example – Osteoarthritis can be identified in early stages by detecting damage in collagen network [7].

Another use of information about fibre orientation of collagen is in biomaterials. Knowing relation between mechanical properties and fibre orientation of collagen fibres can help optimize the biomechanical performance of these materials. [8] For example – optimizing specific heterograft biomaterials used in cardiac valve bioprostheses (BHV) applications [8].

3.2 Imaging techniques

A common imaging technique for skin tissue is electron microscopy. TEM (Transmission electron microscopy) involves detecting transmission of electron beam through a sample. It is used only for ultrathin sections as electrons must pass through the sample [9].

SEM (Scanning Electron Microscopy) is like TEM but the electron reflections and electron emissions are measured instead of transmitted electron beam.

The benefit of electron microscopy is that it offers a resolution of 0.2 nm compared to the 200-nm resolution of light microscopy [10] and is suitable for obtaining images of collagen fibre networks.

Serial block-face scanning electron microscopy (SBEM) is a technique to generate high resolution 3D images. It uses an ultramicrotome to cut away a very thin slice (typically around 30 nm) after each scan. [11]

Collagen fibre images maybe generated using SBEM, for example Gatan 3View System. This system has cut thickness between 15 to 200 nm (typically 25 – 50 nm). The images have a pixel density of up to 24k x 32k. Also, movement is sufficient to cover 1 x 1 mm specimen blocks. The 3View system can traverse through 600 µm in depth. [11]

3.3 Image Processing Concepts

Determining fibre orientation of microstructures such as collagen fibres involves use of several image processing techniques. A study of different concepts was done to aid in the process of developing a successful algorithm. The outline of these concepts is provided in this section.

3.3.1 Image Enhancement

Histogram transformation

Histogram of an image is the frequency or probability with which a certain intensity value occurs in an image. Generally, this intensity profile is not flat and perhaps might not cover the entire intensity spectrum [12].

Histogram equalization involves extending its range to all values between 0 and 1 for each colour channel (spreading the histogram). This is done by the transformation shown in Eq. 1

$$s = T(r) = \frac{1}{c}\int_0^r p_r(x)\,dx \qquad \textit{Eq. 1}$$

where r represents the intensity values of original image, p_r is the original probability of intensity values, s is the new intensity value to which p_r maps, and c is a constant [12].

The histogram is made flat via a process called histogram equalization with random additions whereby pixels are randomly redistributed across neighbouring grey values [12].

Spreading and flattening of histogram are demonstrated in the figure 3.2. Notice how the contrast of the image increases after histogram transformation. This contrast enhancement is helpful when segmenting the image into different components, for example fibres and the background.

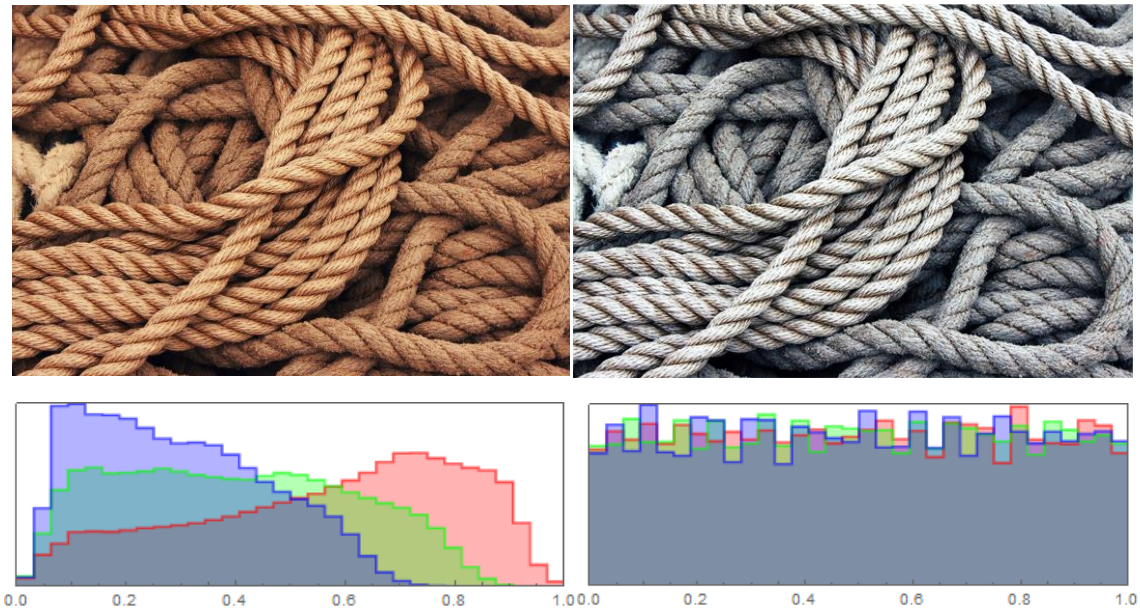


Figure 3.2: Left: A sample RGB image [13] and histogram distribution of its 3 channels, Right: The same image after histogram transformation and the new histogram distribution of the 3 channels

Principal Component Analysis (Karhunen-Loeve transformation)

If the image is not monochrome (grayscale) and contains multiple channels of data (e.g - RGB), principal component analysis can be used to maximize the information presented in a single new channel called the first principal component [12].

If RGB values are plotted on a RGB colour space, then the aim is to transform the axes to find a new set of axes where the data is uncorrelated [12]. Figure 3.3 shows how the image channels data with respect to the original axis is not totally uncorrelated (i.e. – there is a clear pattern that the other 2 channels' intensity also increases when either one of the 3 does). If the axes are placed such that one of the axis passes parallel and through the middle of the cluster, the data will become less correlated. The method involves constructing a transformation matrix using eigenvectors of autocorrelation matrix (3x3) of the RGB channels [12].

Figure 3.4 shows how the first principal component contains more information and has higher contrast than any of the individual red, green or blue components. This is useful in increasing accuracy when segmenting the image.

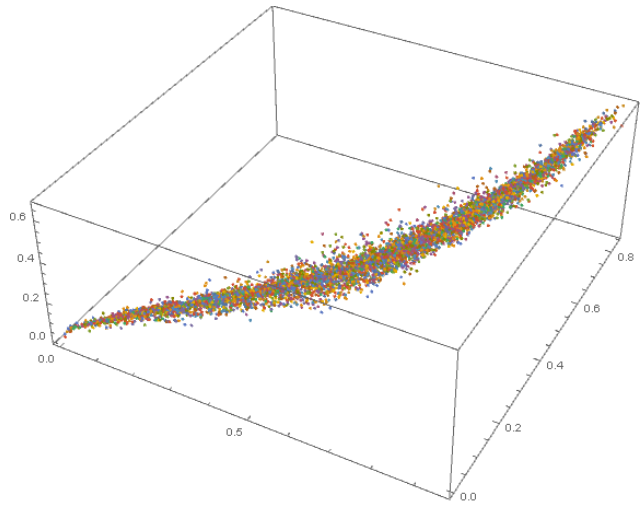


Figure 3.3: All the data points of the original rope image (figure 3.2) plotted on a RGB 3-axis graph (x axis is Red intensity, y-axis is Green intensity, z-axis is Blue intensity)

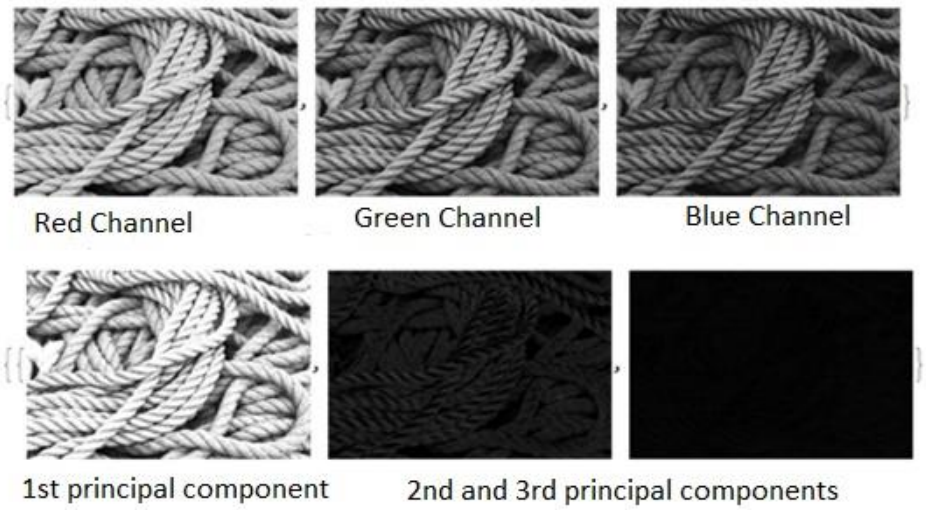


Figure 3.4: Top: Red, green and blue channels respectively of the rope image (figure 3.2), Bottom: 1st, 2nd and 3rd Principal components respectively of the same image

3.3.2 Filtering

Noise is the difference between the true image and the taken image. It modifies values of some of the pixels. Only additive Gaussian noise is considered since it is the most common with modern imaging equipment [12].

Discrete Fourier Transform (DFT) of a $M \times N$ 2D image is defined by Eq. 2

$$G'(u,w) = \frac{1}{\sqrt{MN}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} g(k,l) e^{-2\pi j[\frac{ku}{M}+\frac{lw}{N}]} \qquad \textit{Eq. 2}$$

DFT gives the frequency spectrum of the image and can be useful in identifying and filtering noise.

A crude example of how DFT can be useful in filtering high frequency noise is shown in Figure 3.5. Note how the reconstructed image (in most regions) does not have the same dotted pattern as the original image.

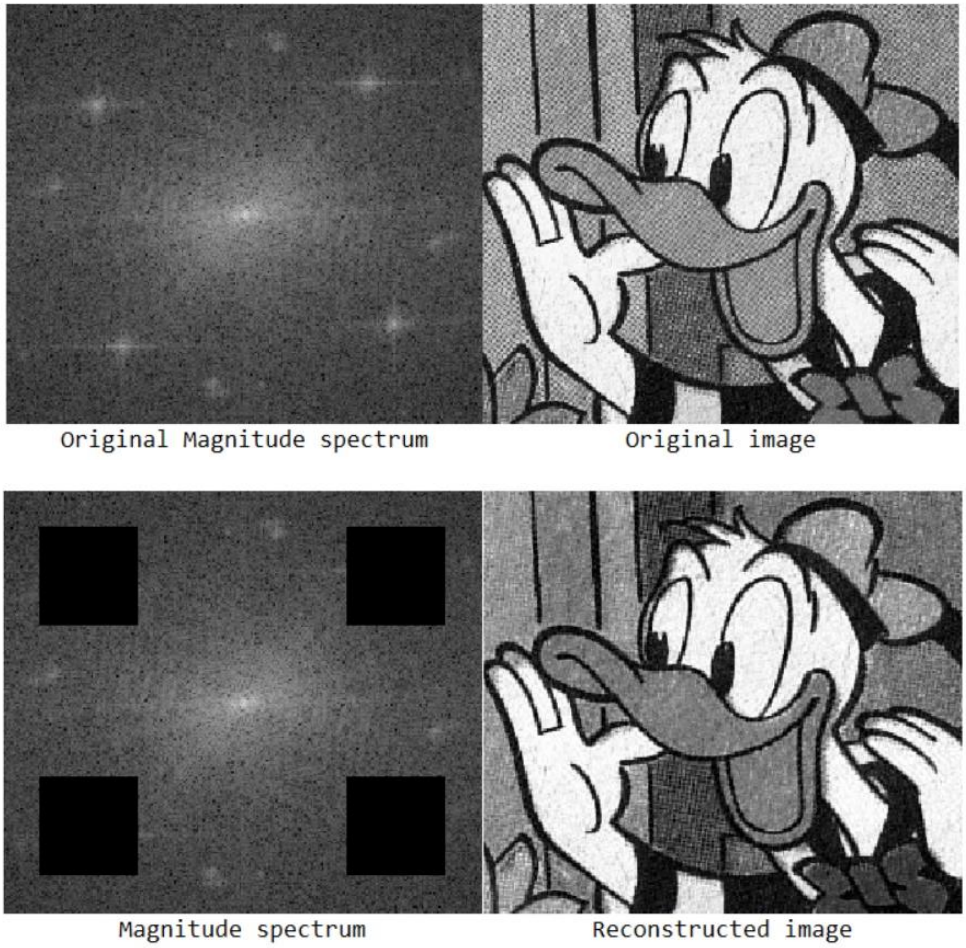


Figure 3.5: Top: A sample image [14] with high frequency noise and its frequency magnitude spectrum (calculated with 2D DFT) (Zero frequency at the centre, higher frequencies further away), Bottom: Same image with a modified frequency magnitude spectrum and the reconstructed image

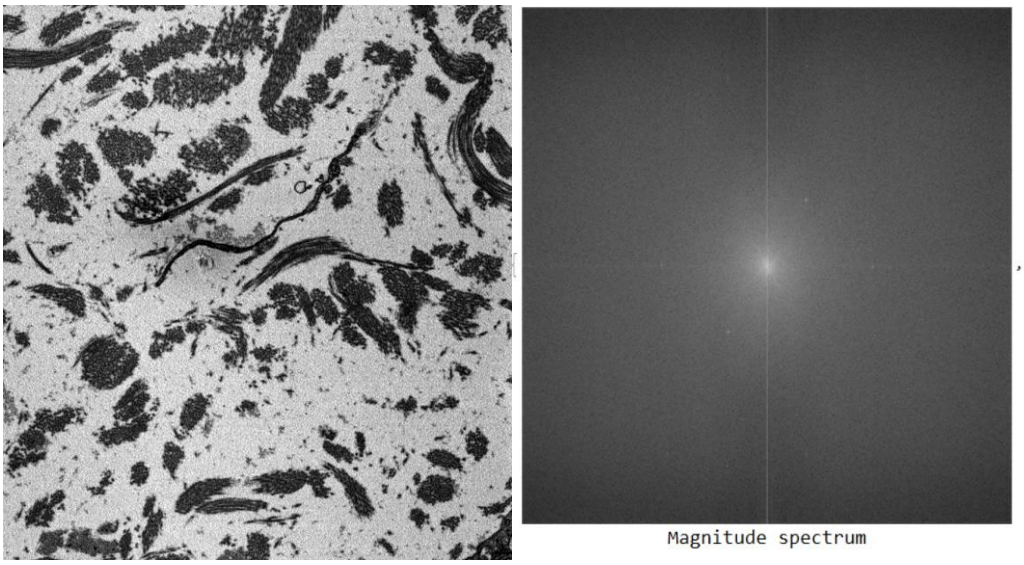


Figure 3.6: A sample image obtained from electron microscopy and its frequency spectrum (vertical and horizontal lines passing through the origin are due to the abrupt intensity change at the image boundary)

Figure 3.6 shows an image obtained from electron microscope and its frequency spectrum. There is comparatively smaller high frequency noise and it is not easy to filter using frequency space methods. Simpler methods such as convolution with filter kernels are more effective.

Some of the common filters include Mean filter, Gradient filter, Sobel Masks and Canny Filters.

Mean Filter:

Convolution of image with kernel of the form:

$$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Gradient Filter:

Convolution of image with 2 kernels of the form:

$$\begin{bmatrix} -1 \\ +1 \end{bmatrix}, \begin{bmatrix} -1 & +1 \end{bmatrix}$$

Sobel Masks (for finding edges in low frequency noise):

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}, \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

Canny edge filter (finding edges in an optimal way assuming Gaussian noise):

It's a multistep algorithm which involves convolving image with a Gaussian filter to remove the noise, finding the intensity gradients of the image using Sobel masks or other similar kernels, applying non maximum suppression to get make the edges thinner (this involves comparing gradient values within each edge), filter out spurious edges which may be due to internal colour differences of components by applying a double threshold, and finally using Hysteresis (discussed in 3.3.3) to remove all edges that are not connected to strong edges. [12]

3.3.3 Segmentation

Segmentation involves dividing the image dataset into a discrete number of segments where each segment usually represents a feature or material. The simplest case of segmentation is binarization where the image dataset is divided into 2 components. Essentially, each pixel is given a value of 0 or 1 depending on which component it belongs to.

This report will focus on binarization methods only since it is assumed that there are 2 phases only – background medium (resin or void) and the solid component (collagen fibre).

Global thresholding:

An intensity value θ is chosen and binarization performed according to Eq. 3

$$g(X) = g(x,y,z) = \begin{cases} 0, & f(x,y,z) < \theta \\ 1, & else \end{cases} \qquad \text{Eq. 3}$$

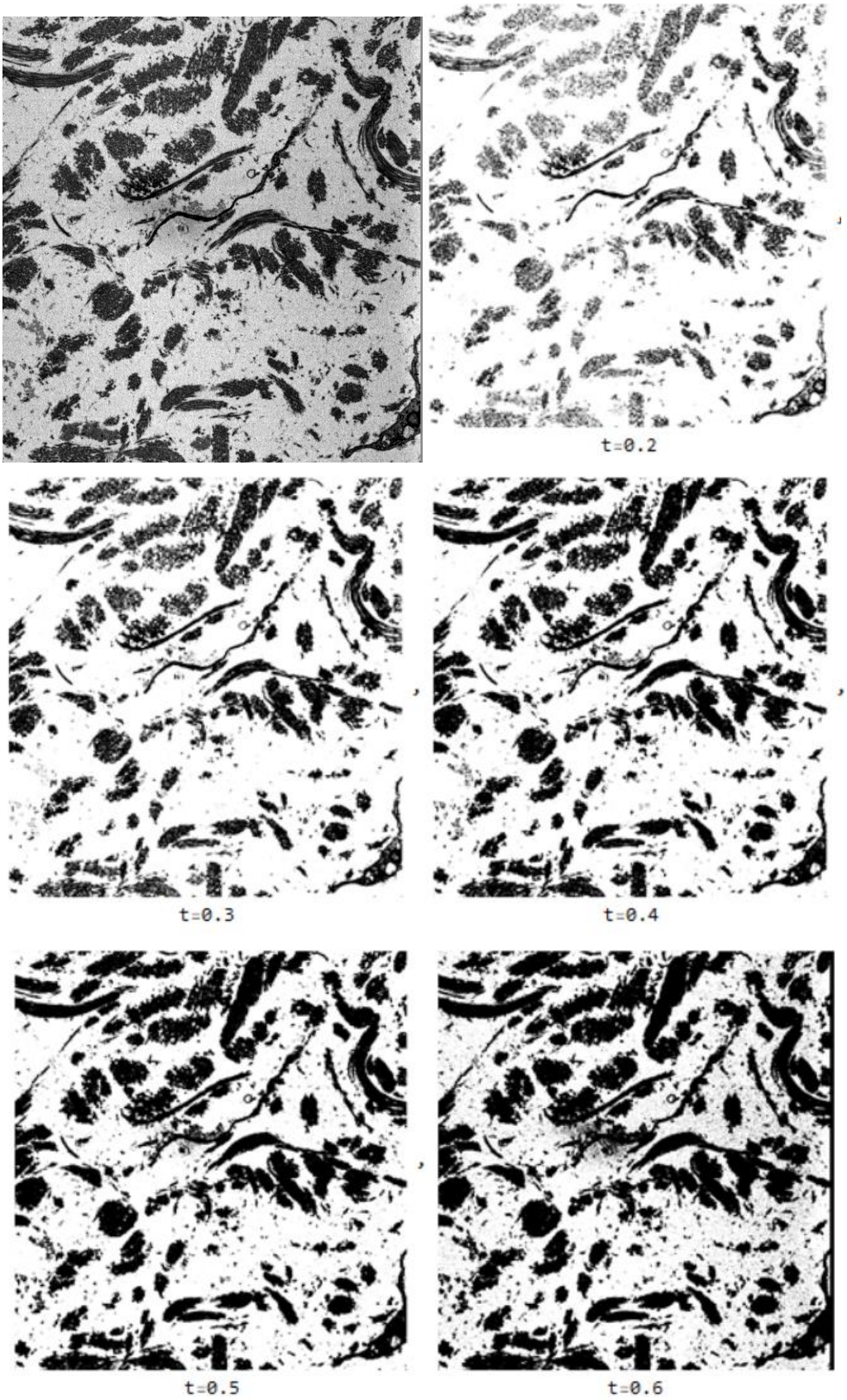


Figure 3.7: A sample raw grayscale image of biological tissue (top-left, same as figure 3.6) and binarized versions with global threshold $\theta = 0.2$, $\theta = 0.3, \theta = 0.4, \theta = 0.5$ and $\theta = 0.6$

$f(x, y, z)$ represents the image intensity at coordinate $X = (x, y, z)$ and $g(x, y, z)$ represents the binary image containing only 0s and 1s. The choice of threshold θ is crucial in this method and can change the resultant image significantly, as shown in figure 3.7.

Global thresholding is ineffective when signal to noise ratio is low or when the intensity values of the 2 phases overlap [15]. It is also ineffective when the background intensity values vary significantly throughout the volume [15].

Choosing θ is a challenge. There are several ways to do it-

Minimum error threshold method:

Threshold is chosen using prior knowledge of probability of a pixel to belong to the background or foreground [15]. It is not useful for this report as we don't have information about image beforehand.

Otsu's method

Histogram distribution of grey values is analysed and two separate peaks are identified, one for background, one for foreground [12] [15] The θ will be the minimum between these 2 peaks. The weighted sum of within class variances (background class and foreground class) is minimized [12] [15]. This is equivalent to maximizing interclass variance $\sigma_B^2(\theta)$ [12]. This method is widely used but assumes variances are well behaved as a function of θ [12].

Isodata Method

This method is like Otsu's method but searches locally instead of finding a global optimum. It is an iterative method– initial threshold θ_0 is set and mean background and foreground grey values ($\mu_0^{(0)}$ and $\mu_1^{(0)}$) are computed with that threshold [15]. Eq. 4 is used for next iteration and iteration is continued till convergence [15]

$$\theta_1 = \frac{\mu_0^{(0)} + \mu_1^{(0)}}{2} \qquad \textit{Eq. 4}$$

This method is widely used, but is known to be computationally expensive [15]

Bayesian thresholding

This method assumes Gaussian distribution of background and foreground grey values with different means but same variance ($\mu_0,\ \mu_1,\ \sigma^2$). It uses the probability expression $p(f(\mathrm{X})|j)$ where $j \in \{0,1\}$. It is asserted that ideal threshold will be when $p(\theta|0) = p(\theta|1)$ and θ is calculated, which results in Eq. 5 [15]

$$\theta = \left(\frac{1}{2}\right)(\mu_1 + \mu_0) + \frac{\sigma^2}{\mu_0 - \mu_1}\log\frac{p(0)}{p(1)} \qquad \textit{Eq. 5}$$

This method requires estimation of the mean of 2 classes and the standard deviation. It is not useful for images where this information is not known.

Local thresholding

If the intensity levels of an image vary depending on location, a shading correction is needed. In local thresholding, subtracting local mean intensity value from pixel values achieves that. Problem of local thresholding is of the type shown in Eq. 6 [15]:

$$g(X) = \begin{cases} 0, & f(\mathrm{X}) < \theta(\mathrm{X}) \\ 1, & else \end{cases} \qquad \textit{Eq. 6}$$

Niblack Thresholding

$$\theta(X) = Mean\{f(X')|\ \|X - X'\|_\infty \leq W\} + \lambda\sqrt{Var\{f(X')|\ \|X - X'\|_\infty \leq W\}} \quad \text{Eq. 7}$$

Mean and Variance around central voxel X are calculated and used as a local representation of image to get the threshold $\theta(X)$ as shown in Eq. 7 [15]. Here, W is the window size around voxel location X and λ is a parameter. In 3D, this approach works best when W chosen corresponds to size of objects [15]

Method fails at large low contrast areas. If W at X is a smooth area with little noise, then estimated variance will be low and low frequency variations or noise can cause false segmentations. [15]

The electron microscope images are generally uniformly lit in all areas and there is no need for a shading correction compared to macroscopic environment images.

Hysteresis binarization

A common problem in both local and global thresholding is noise. For example – If threshold was set too large to try to miss the bright noisy pixels, the segments identified might miss some of the parts of the true segment. A threshold too low would be include noise of the background also. [12]

Hysteresis thresholding uses two thresholds $\theta_1 > \theta_2$. Starting from voxels with f(x)> θ_1, voxels are iteratively assigned to the foreground if they fulfil the condition (x)> θ_2. This ensures that regions that are connected should be segmented as one more easily and noisy background voxels are suppressed. [12] [15]

In practice, this technique is not very computationally expensive and produces excellent results [12].

Region growing

Growing by grey value: While the region is growing, its current mean and standard deviation are computed, and new voxel is added if it lies within the range around region's mean [15].

Adams Seeded Region Growing : Like above method, but search for voxel is performed over all regions, not one region at a time [15]. Essentially, the different regions compete for each pixel.

These algorithms usually require initial seeds to grow from [15] (e.g. – the brightest pixels in the image which might be at the centre of each foreground component). It is difficult to determine the seeds automatically if no prior information about the image is known, so these methods are not always ideal for fully automatic algorithms.

Summary

Having tried a multiple of these methods, it was found that different binarization methods suit different types of images. The code for binarization implementation can be found in Appendix 10.1.7.

Overall, it can be concluded from figure 3.11 that applying gradient filters (Canny and Sobel) for edge detection produces good results when there are no internal boundaries within the microstructural features (fibres). The overlap of different fibres is also visible quite clearly. However, if the original image contains features with non-smooth & non-straight boundaries and erratic interiors (holes and noise) such as in figure 3.8, the gradient filtering and edge detection does not produce the best binary image (as seen figure 3.9)

On the other hand, hysteresis thresholding (which is essentially a combination of global thresholding and region growing) produces a good quality binary image with little to no holes and does not capture any of the undesired detail of the features within each microstructure. (figure 3.8). However, it does not capture the overlapping fibres as shown in figure 3.10. There are advantages and disadvantages to each method.

It was found that the binarization quality of the 3D images was very similar to the binarization quality of any of the constituents 2D image making up the 3D image. Therefore, before binarizing 3D data and expending time and computational power, it is useful to binarize a constituent 2D image and check if the result is desirable.

(Note: for figures 3.9 and 3.11, the algorithm for the left image is almost the same as the right image. The only difference between the algorithm on the left and the Canny edge detection on the right is the lack of applying non-maxima suppression and the initial convolution being with mean filter instead of Gaussian filter. This is why the only visual difference between the 2 binarized images is the thickness of lines)

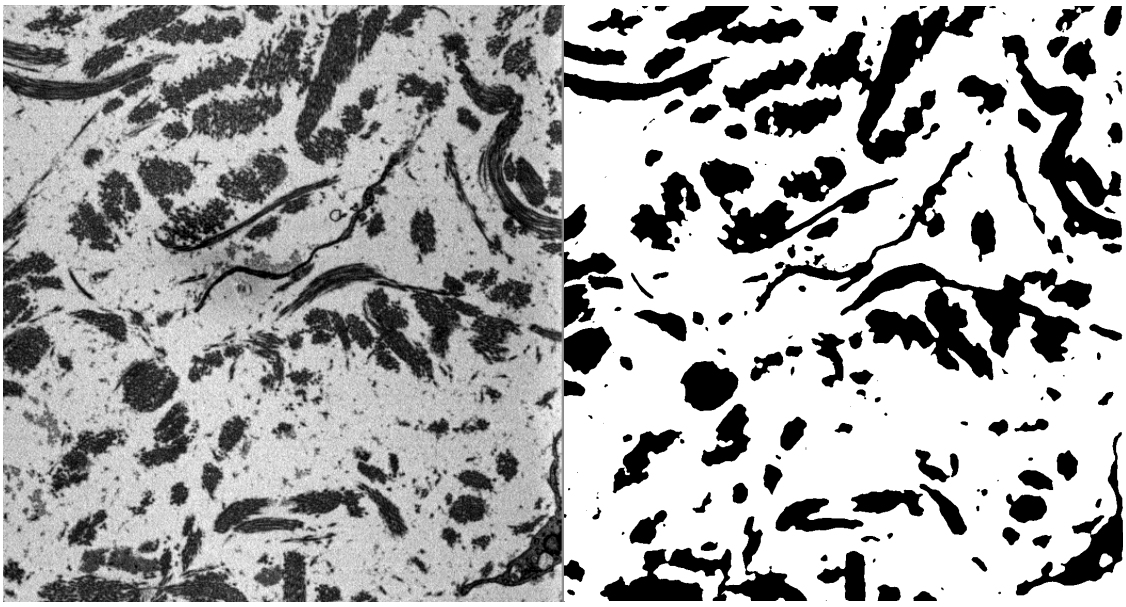


Figure 3.8: Left: The original grayscale image (same as figure 3.6), Right: the image binarized via the following sequential steps 1) Mean Filter with window radius 5, 2) Optimal value of upper threshold found using Otsu's method 3) Hysteresis segmentation to binarize with upper threshold from 2) and lower threshold of 0.4

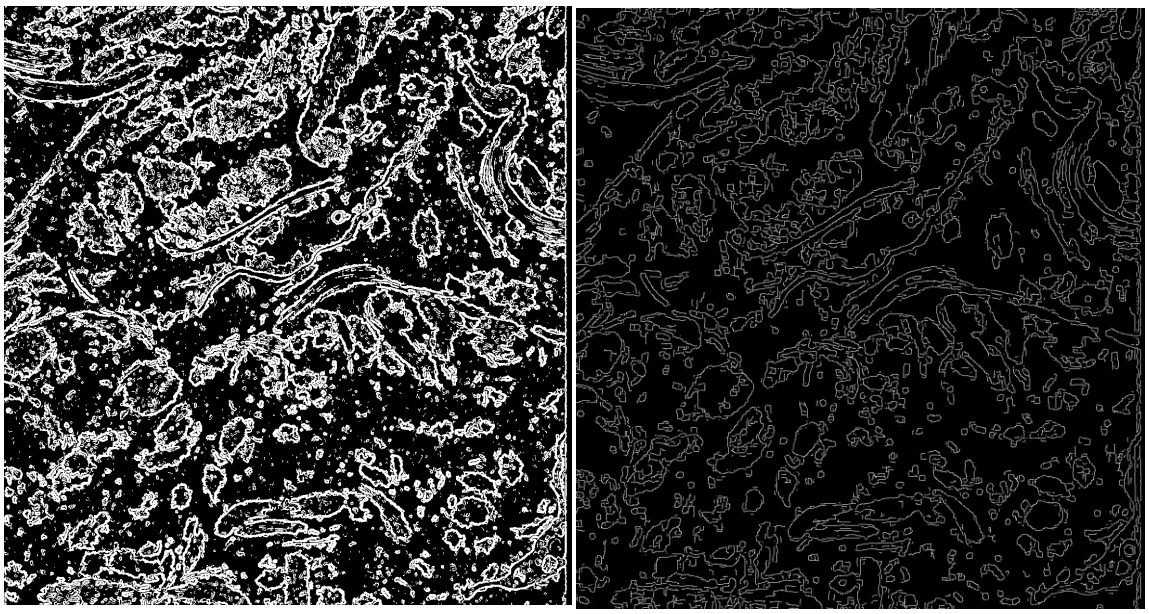


Figure 3.9: Binarization of the original image in figure 3.8 with 2 different methods, Left: Convolving with mean filter and Sobel filter in that order before binarizing using Otsu's global threshold, Right: Canny edge detection algorithm applied to the gaussian filtered image

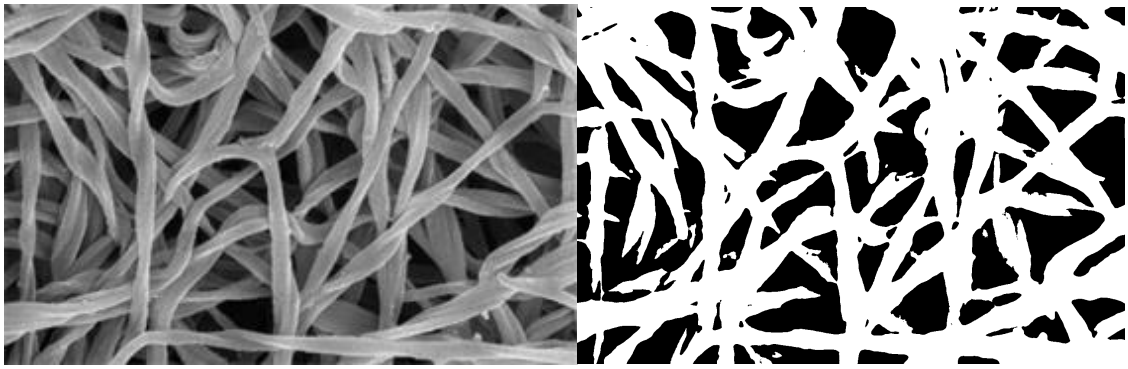


Figure 3.10: Left: The original grayscale image [16], Right: image binarized via the following sequential steps 1) Mean Filter with window radius 5, 2) Optimal value of upper threshold found using Otsu's method 3) Hysteresis segmentation to binarize with upper threshold from 2) and lower threshold of 0.4

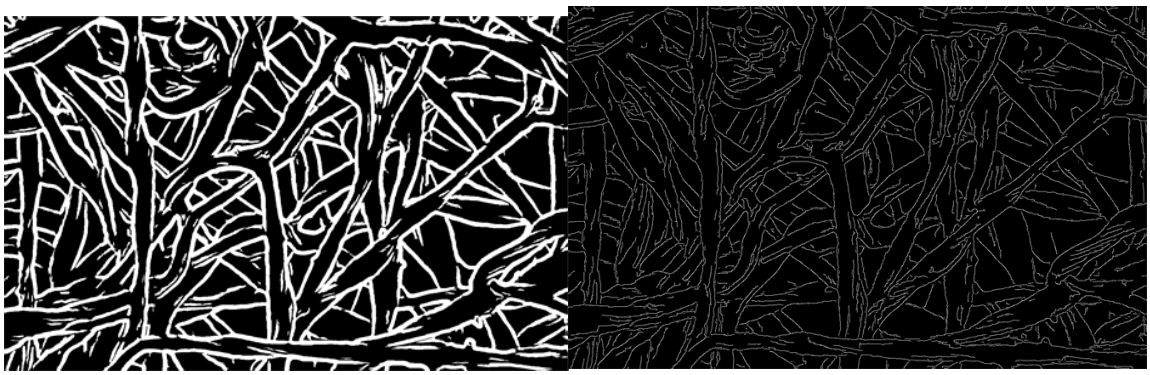


Figure 3.11: Binarization of the original image in figure 3.10 with 2 different methods, Left: Convolving with mean filter and Sobel filter in that order before binarizing using Otsu's threshold, Right: Canny edge detection algorithm applied to the gaussian filtered image

3.3.4 3D Reconstruction

Most often, the dataset will consist of a stack of images each of which represents a layer. Construction of 3D image from the 2D image stack can present challenges.

Gatan provides a 3D visualization software called *DigitalMicrograph* [11]. It allows simple stacking of images to construct a 3D model. Similar functionality is available in Mathematica also, using Image3D function which allows stacking of 2D images to form a 3D dataset [17]. The latter was used in this report.

However, if the distance between the slices is significantly greater than voxel size, 3D data between any 2 layers will have to be reconstructed.

Intensity value interpolation

There are several ways in which this can be done. Simplest technique is a linear or quadratic or cubic interpolation of intensity. So given two intensity values $f1$ and $f2$ at same point (x, y) but different z-layer, a linear interpolation of the intensity varying linearly from $f1$ to $f2$ could be made. So at a location exactly between the 2 layers, the intensity will be $(f1 + f2)/2$

This is available in Wolfram Mathematica through the inbuilt *ImageResize* function and *Resampling* option [17].

This technique is best suited when the histogram distribution of intensity is uniform. If there are multiple distinct phases in the image, then interpolation of intensity might not be an accurate way to estimate in between layers because it is likely that the in-between intensities do not represent either object clearly.

Features mapping

Another technique is to segment the 2D layers first and try to map the components between different layers.

Different segments can be identified in each layer. The segments which correspond to the same component/feature in different layers will have to identified. Essentially, segments will have to be mapped from one layer to another. Mapping segments in different layers can be done using Scale Invariant Fourier Transform (SIFT) or Speeded up robust features (SURF) techniques [18]. Scale invariant transform can map the images of same scene taken from different depths regardless of scale or rotation [18].

Once the mapping between different layers is found, the slices between can be interpolated. The interpolation here will be of the position of the features instead of the intensity values of individual voxels.

There exists a array of commercial voxel reconstruction software which can be used to interpolate the images between 2 layers. For example – Imaris, BioVis3D and Vaa3D [19] [20] [21]

For the purposes of this report, it will either be assumed that the there is enough data in z-direction so a simple stacking would suffice.

3.4 Fibre orientation determination techniques

Fibre orientation has been a topic of interest for decades in different fields including material sciences (e.g. – for optimizing mechanical properties of fibrous composite materials [22]) and micro CT scans [23]. Therefore, several methods have been developed which will be summarized and discussed.

The state of the research suggests that orientation of fibres in 2D images has been very well studied and successful methods found [24]. Usually, these involve Fourier or Hough transforms. 3D orientation determination algorithms are a fairly new area and are not so well studied [24].

3.4.1 Gaussian orientation space filters

A set of discrete points in the orientation space are chosen and the corresponding Gaussian filters are determined. Each gaussian filter corresponding to a discrete point represents a specific orientation [25]. Gaussian filters are of the form shown in Eq. 8.

$$g(x) \propto \exp\left(-\frac{1}{2}x^t \alpha^{-1} x\right) \qquad \text{Eq. 8}$$

α is a covariance matrix that describes the shape and orientation of the filter [25].

For each voxel, the Gaussian filter that provides the maximum filter response is determined and the corresponding orientation is designated as the local orientation for that voxel [25].

However, the shortcomings of this method include the fact that it can produce severe edge effects and image often needs to be cropped [25]. Also, the fibre radius is assumed to be constant and the parameters of the anisotropic filter need to be changed depending on the fibre radius [26].

3.4.2 Fourier Transform based approaches

Fourier transform gives the frequency space behaviour of the image. This was already seen in section 3.3.2. This is useful for extracting orientation information that is not readily available in the real domain. There are several methods to do this.

Bandpass filter on power spectrum

This technique emphasises filtering of the frequency in a very narrow passband range corresponding to the fibre thickness. The data in 2D Fourier frequency space is converted into a single radial frequency and a corresponding phase angle by transformation shown in Eq. 9, Eq. 10 and Eq. 11 [27].

$$u_r = u^2 + v^2 \qquad \text{Eq. 9}$$

$$\theta = \tan^{-1}\left(\frac{v}{u}\right) \qquad \text{Eq. 10}$$

$$F(u,v) = \hat{F}(u_r,\theta) = F(u_r \cos\theta, u_r \sin\theta) \qquad \text{Eq. 11}$$

$F(u,v)$ is the original Fourier response of the 2D image and $\hat{F}(u_r,\theta)$ is the transformed response.

The power of the DFT is calculated using Eq. 12 and angular amplitude is calculated using Eq. 13 [27].

$$P(u_r,\theta) = \left|\hat{F}(u_r,\theta)\right|^2 \qquad \text{Eq. 12}$$

$$A(\theta) = \sum_{u_r=f_1}^{f_2} P(u_r, \theta) \qquad \text{Eq. 13}$$

The bandpass frequencies f_1 and f_2 are given by Eq. 14 [27].

$$f_c = Max(R, C)/(2 * t), \qquad f_1 = 0.9 * f_c, \qquad f_2 = 1.1 * f_c \qquad \text{Eq. 14}$$

t is the fiber thickness, R is the number of rows in the image and C is the number of columns in the image.

Clearly, each orientation is weighted by the power of the filtered frequency response corresponding to that orientation.

This method is suitable for finding overall distribution rather than local vectors in the image [27]. However, applying to smaller sub images/regions of interests (ROIs) can be used to generate more detailed orientation information with increase in computational time. This method is only applicable to 2D Images.

Intensity averaged over orientation

This is a very similar approach to the bandpass filter approach. Instead of applying a bandpass filter, a simple analysis of the amplitudes of 2D frequency spectrum in any orientation can also be done [28]. The amplitude frequency spectrum needs to transformed so that the origin at the centre of the image represents the zero frequencies.

For each orientation in the frequency spectrum, the average intensity value is found along the line passing through the centre (zero frequency) and gradient corresponding to the orientation in question [28]. A line plot of average intensity value vs orientation can be produced. This gives the fibre orientation distribution throughout the image and the peak values can be determined [28]. As with other Fourier analysis methods, it is not useful for local orientation information within the image unless applied iteratively to small regions of interests (ROIs).

Filter Bank

The image is divided into different Regions of interests (ROIs). Fibre orientation within each ROI determined by filter bank method.

Filter bank contains various 3D orientation filters in the Fourier space i.e. - each filter corresponds to an orientation [29]. These are implemented in form of separable 1D filters. The frequency space is iteratively decomposed into a 2-band tree structure [29]. Coarse to fine approach is used in deciding the filters. The analysis is started with a small number of orientations and after each iteration, the possible orientation range is fine tuned. (2 band tree structure) [30].

If a ROI does not have enough pixels, it is classified as dark. If all the filter banks produce similar output, the ROI is isotropic. [29]

Fourier Transform of each ROI is compared with the filter banks by calculating correlation of each filter with the FT. The 3D orientation with maximum correlation is designated to the ROI [29].

Complexity of image is significantly important in determining whether the desired results will be produced [24].

The size of ROI represents a challenge as it needs to be ensured that enough local details are captured without sacrificing too much computational power and time.

Fourier zeroth-order maximum analysis with Gabor filter

This technique combines Fast Fourier Transform (FFT) and Gabor filters. After the pre-processing steps, Gabor filters with different orientations are applied are applied to the image [31].

Gabor filters are sinusoidal modulated Gaussian in the spatial domain. They are used to detect and highlight fibre edges for any orientation. They have applications in image detection as well as texture recognition [31].

Gabor filters for a small set of discrete directions such as 0°,45°,90° and 135° are applied to the image. This semi orthogonal set is chosen for quantifying the integrity of the basket weaving texture of the fibres, as that is a good indication of the skin health and age [31].

Windowing is performed on the resultant images, to remove the vertical and horizontal lines that appear in the frequency domain because of the vertical and horizontal discontinuities at the edge of a standard image [31].

The FFT on these Gabor filtered results usually produces elliptical shape in 2D frequency domain [31]. Taking the major and minor axis of these different FFT images, an orientation index can be calculated as in Eq. 15 [31].

$$N\ (Orientation\ index) = \frac{Max\ n}{Min\ n}, \qquad n = 1 - \frac{Short\ Axis}{Long\ Axis} \qquad \text{Eq. 15}$$

The orientation index closer to 1 means the basket weave structure is not strong but a higher value means that the basket weave structure integrity is strong and the skin age/ health can be classified accordingly [31].

Note that this method is not useful in determining local orientations in the image. Instead it is used to assess the overall orientation characteristic of the image. Therefore, this is not suitable for generation of the vector field, which is the focus of this study.

3.4.3 Hough Transform

Hough transform iterates through the image space and for each pixel in foreground iterates through the set of possible lines [32].

The lines are parameterized by the angle θ and the radius ρ as shown in Eq. 16 [28] [32].

$$\rho = x\cos\theta + y\sin\theta \qquad \text{Eq. 16}$$

It computes the parameters of the line (angle θ and radius ρ) with maximum occurences (equivalent to finding peaks in the Hough domain θ, ρ). The peaks represent the identified lines in the image [32].

It is often useful to first detect edges in the image before applying Hough transform [32]. Hough transform is only suited for finding straight lines through the image which is not necessarily the case for collagen fibre images. However, the transform can be done on smaller windows of interest such that the fibre can be estimated to be straight [28].

For each window, the peak value in the Hough domain θ, ρ is found and the fibre orientation for the sub image is assigned that resulting θ value [28].

3.4.4 Principal component analysis

Structure tensor

Structure tensor at each pixel of an image is defined using Eq. 17 [33].

$$S_{ij} = G_\sigma \left(\frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} \right) \qquad \text{Eq. 17}$$

where G_σ is a Gaussian function, f is the image and x_i and x_j represent the coordinate directions (2 in case of 2D images and 3 in 3D images)

Structure tensor and its eigenvalues and eigenvectors can be calculated at each pixel. The eigenvector corresponding to the largest eigenvalue (in case of 2D image) or corresponding to the smallest (eigenvalue (in case of 3D) indicate the direction of main axis/orientation of the pixel [33].

This is valid near the surface or edges only because it uses the gradients along the x and y directions before calculating the principal component.

Moment of Inertia

A similar approach uses the moment of inertia tensor (where inertia corresponds to the pixel intensity values). For each sub image or region of interest (ROI) this tensor is calculated. The main principal axis of the resultant moment of inertia tensor represents orientation of the ROI [34].

In general, the moment of inertia of a body is defined using Eq. 18

$$I = \sum_{i=1} m_i r_i^2 \qquad \text{Eq. 18}$$

where m_i is mass of the ith particle and r_i is the distance from centre of mass. In images, this mass m_i corresponds to the intensity value of the ith pixel.

The 3D inertia tensor is given by Eq. 19 [35].

$$I = \sum m \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} = \sum m \begin{bmatrix} y^2 + z^2 & -xy & -xz \\ -xy & x^2 + z^2 & -yz \\ -xz & -yz & x^2 + y^2 \end{bmatrix} \qquad \text{Eq. 19}$$

The eigenvectors of this tensor are the principal components of the image and the eigenvector corresponding to the largest eigenvalue is the desired orientation of the image [33].

Principal component analysis has limitations due to shape, thickness and density of fibres. This is because the ROIs where the principal components are determined should ideally cover as much of a single fibre as possible (while it is relatively straight) but not multiple fibres. The division of image into appropriate ROIs is therefore challenging. If the ROIs are chosen to be too small, the principal component of a ROI which solely contains fibre pixels would present no information about fibre orientation (assuming square ROI).

Changing fibre radius (thickness) leads to inaccuracies due to inappropriate ROI window size selection [34]. In this paper, a method for overcoming this weakness will be suggested.

3.4.5 Weighted orientation vector summation algorithm

2D

The core of this algorithm is measuring intensity variations across a fibre in different directions. The variations in intensity are more abrupt while travelling across the fibre diameter, but subtle along the direction of a fibre. Therefore, lack of variance in pixel values in a direction means that the fibre is likely to be oriented that way [36].

A small window is defined (such as 7x7 pixels) which is used to detect intensity variations. For each pixel location, all the possible vectors passing through the centre and 2 other pixels symmetric to the centre pixel are determined [36], as shown in figure 3.12.

These vectors need to be weighted by the inverse of their length because otherwise the vectors which are longer would influence the results more than the ones which are shorter in magnitude. A vector weighting of the form $W_1 = \frac{1}{L}$ (a representation of this weighting for the 7x7 window is shown in figure 3.12) is applied to the original vectors to make all their lengths equal [36].

The variance across each of these vectors is calculated. A second weight W_2 is defined which is a representation of the lack of variance across the vector (shown in Eq. 20) [36].

$$W_2 = \sqrt{\frac{1}{3}} - \sqrt{\sum_{i=1}^{3} (\bar{a} - a_i)^2} \qquad \textit{Eq. 20}$$

For each vector at each pixel in the image, weights W_1 and W_2 are calculated to find how much that vector is likely to be the orientation of the pixel. The cumulative sum of all the weighted vectors at a certain pixel gives the average vector orientation at that pixel [36].

Compared to orientational filters and edge detection algorithms (such as Sobel or Canny operators) this technique has improved accuracy [36].

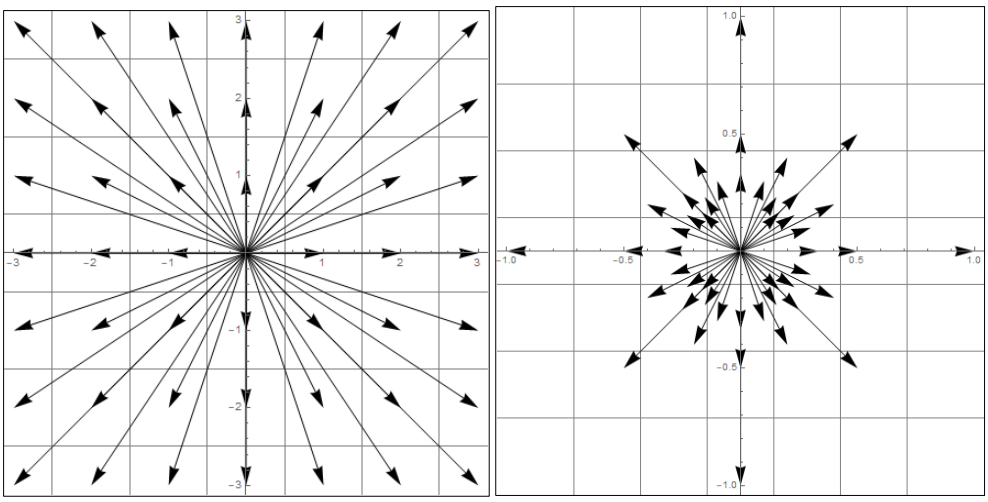


Figure 3.12: Left: A 7x7 window and all the possible vectors passing through the centre pixel of the window. Right: A representation of the weightings ($W_1 = \frac{1}{L}$) for each vector. Multiplying each of the corresponding weighting and the original vector results in all the vector lengths becoming equal.

3D

The 3D weighted vector summation is very similar to the 2D technique. The difference is that the window is a cube and the possible vectors in the windows are 3D. These 3D vectors can be defined using 2 parameters θ and φ which are the angle between vector's projection onto the xy plane and the x-axis, and angle between the vector and the z axis, respectively. These are shown in figure 3.13 [24].

However, to make calculations simpler, 2 additional angles are defined. These are angle between vectors projection on zx plane and x-axis (β) and angle between vectors projection on yx plane and the y-axis (γ) [24] (shown in figure 3.13). The transformation between the 2 representations can be done using Eq. 21

$$\tan^2 \varphi = \frac{1}{\tan^2 \beta} + \frac{1}{\tan^2 \gamma} \qquad \textit{Eq. 21}$$

To simplify the problem and save computational steps, the 3D image is projected onto three orthogonal 2D planes in each window. The angles θ, β and γ can be calculated using the 2D algorithm for each of these three 2D projections [24].

It should be noted that the 2D images (projections) used for the orientation calculation are averages of the parallel layers in the 3D window [24]. This makes sure all the data in the window is considered.

This algorithm is one of the latest developments in the field (having been published in 2015) and is able to acquire the orientation with higher accuracy than 2D stack analysis techniques [24].

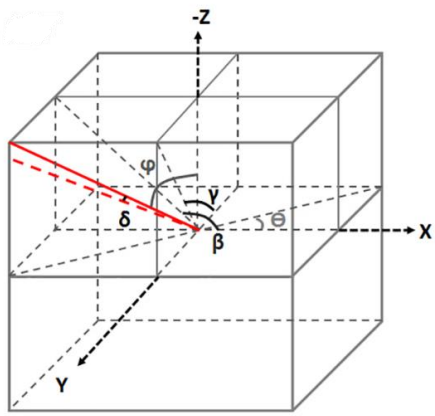


Figure 3.13: Representation of the angles use to define a 3D vector. Either θ and φ can be used or θ, β and γ [24]

3.4.6 Gradient orientation filters

Canny and Sobel filters discussed in section 3.3.2 are used for edge detection. They calculate the gradient. However, they are separable filters and calculate gradient in x and y direction independently [12]. Therefore, along with the gradient magnitude, information about gradient orientation can also be retrieved by applying these 2 filters [12].

The local orientation is calculated only for the edge pixels. This information can be interpolated to the interior of the fibres. However, for short fibres that have sharp changes in orientation, this can result in inaccurate orientation results.

These can be applied in 3D images also. However, it was found with experimentation that the smoothness of edges is very important for good accuracy of the results from this

algorithm. The results often tend to be inaccurate if the resolution of the image is poor as the edges are not smooth, but instead rugged.

3.4.7 Fibre tracking approaches – energy minimization algorithms

Fibre tracking approaches are popular in retrieving information about fibre positions and orientations from diffusion MRI data [37].

Algorithm tracks fibre bundles by calculating the local orientation at a voxel using principal eigenvectors of tensors or using the probability density of diffusion orientations [37]. It moves to the next voxel based on the orientation of the previous voxel and continues to do so iteratively [37].

This is prone to creating short and fragmented fibres due to either a wrongfully identified isotropic voxel or generating incorrect directions which go nowhere [37]. Moreover, the traditional methods cannot determine which direction to follow in a region where 2 fibres are crossing [37]. However, a solution to this has been suggested in the Fibre Assignment by Continuous Tracking (FACT) algorithm [38].

This approach is iterative and time consuming, and performance highly dependent on the complexity of images [24]. It is less feasible when applied to dense fibrillar objects [24]. And some level of user interaction is required [24].

3.4.8 Chord Length transform

Chord length transform maps each pair of (point, direction) to a length of the chord [39]. Performing the transform with a discrete set of directions, the result is a stack 2D images, which can be considered as a 3D image [39].

Orientation of a foreground pixel is designated as the one corresponding to the longest chord length that fits through its surroundings [39].

This transform is particularly useful when dealing with 2D images where the fibres are crossing because it reformulates the crossing of fibres 2D image into a segmentation problem of a 3D image [39].

The biggest limitation is that it works best only when the fibres are nearly straight and of significant thickness [39]. Also, the crossing angles should not be too small and this method works on binary images only [39].

3.4.9 Summary

Of the methods discussed, all are applicable or extendable to 3D images except some of the Fourier transform based approaches. However, very few have been traditionally used for 3D analysis because of accuracy and performance reasons. 3D weighted vector summation is one of the latest developments in the field so this will be used as a comparison point in this study.

4 The Proposed Algorithm

The algorithm proposed in this paper is based on principal component analysis of moment of inertia (see section 3.4.4). It is not a drastically new approach. However, it has been implemented in a way to reduce computational times and increase applicability to different situations.

The algorithm was first developed and studied for 2D images before extending it to 3D images.

4.1 Core algorithm

The 2D implementation is discussed first.

Let the image of size $r \times c$ be defined by the matrix I and function $i(row, column)$ as shown in Eq. 22

$$I = \begin{pmatrix} i(1,1) & \cdots & i(1,c) \\ \vdots & \ddots & \vdots \\ 1(r,1) & \cdots & i(r,c) \end{pmatrix} \qquad \text{Eq. 22}$$

$$A = vec(I^T) \qquad \text{Eq. 23}$$

vec in Eq. 23 represents vectorization of the matrix into a single column. A represents a vector of image data with dimensions $(rc, 1)$. A will be of the form shown in Eq. 24

$$A = [\, i(1,1) \quad i(2,1) \quad \dots \quad i(r,1) \quad i(1,2) \quad \dots \quad i(r,2) \quad \cdots \quad i(1,c) \quad \dots \quad i(r,c)\,]^T \qquad \text{Eq. 24}$$

Similar transformation was done to the pixel coordinates data. P in Eq. 25 represents a vector of dimension $(rc, 1)$ containing coordinates of all the points

$$P = [\, (1,c) \quad (1,c-1) \quad \dots \quad (1,1) \quad (2,c) \quad \dots \quad (2,c) \quad \cdots \quad (r,c) \quad \dots \quad (r,1)\,]^T \qquad \text{Eq. 25}$$

P can be represented as two separate matrices P_x and P_y both of dimensions $(rc, 1)$ as shown in Eq. 26

$$P = (P_x, P_y) \qquad \text{Eq. 26}$$

Centre of mass (Centroid, P_c) of the image is found as shown in Eq. 27

$$\begin{aligned} P_c &= (x_c, y_c) \\ x_c &= \frac{A^T P_x}{\sum A} \\ y_c &= \frac{A^T P_y}{\sum A} \end{aligned} \qquad \text{Eq. 27}$$

L is defined as a vector containing all the same rc coordinate points as P but transformed to be centred around the centroid, as shown in Eq. 28

$$\begin{aligned} L_X &= (P_x - x_c) \\ L_y &= (P_y - y_c) \\ L &= (L_x, L_y) = (P - P_c) \end{aligned} \qquad \text{Eq. 28}$$

The transformed coordinate points L are now weighted by the corresponding pixel intensity values to give the weighted matrix D, as shown in Eq. 29

$$\begin{aligned} D_x &= L_x A^T \\ D_y &= L_y A^T \\ D &= (D_x, D_y) = LA^T \end{aligned} \qquad \text{Eq. 29}$$

The inertia tensor is calculated by iterating through matrix D and calculating the elements of inertia matrix using Eq. 19 and summing them to the overall inertia tensor.

The eigenvectors of this tensor represent the principal components and the required orientation vector could be found.

The core of the algorithm is similar in 3D analysis.

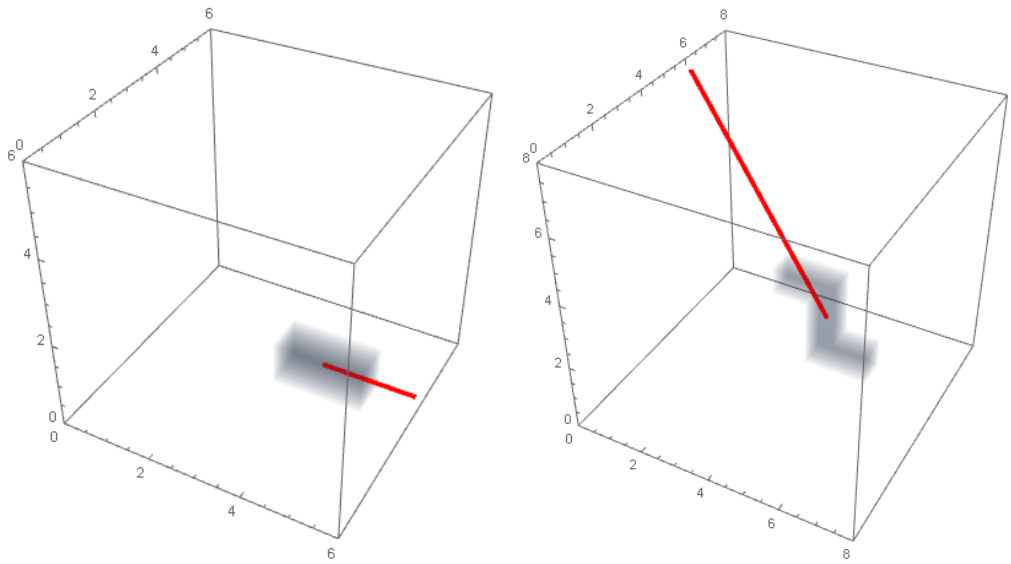
In 3D analysis, the vectors A and P include the intensity values and coordinates of pixels in all 3 dimensions instead of 2 (addition of z dimension). The inertia tensor is 3×3 instead of 2×2 and three principal components are found instead of two.

The relevant Mathematica code for the 3D implementation can be found in Appendix 10.1.1.

4.2 Regions of Interest (ROIs)

The core algorithm explained in section 4.1.1 must be applied to all sub images or ROIs to capture the required local orientation information.

The vector results depend on the ROI window size. Position and amount of fibre in each ROI affects the moment of inertia and hence the orientation result. For example - figure 4.1 shows fibre in the same region of an image captured differently in 2 different ROI window sizes. The resulting orientation is different for the two cases. Hence, choosing appropriate ROI sizes is the main challenge when applying this method.



A B

Figure 4.1: Application of the core algorithm in section 4.1.1 to 2 different ROIs. A captures only a part of the structure captured in B. The red line represents the calculated orientation. A:6x6x6 window, B: 8x8x8 window

Another challenge is when there are more than one fibre present in the same window. For example- figure 4.2 shows how the calculated orientation would be completely incorrect if the 2 disconnected fibres were captured in the ROI.

These challenges were overcome in this implementation. One feature which makes the algorithm unique is that it can distinguish between components in the ROI. The ROIs are scanned for existence of disconnected components, like 2 separate fibres in figure 4.2. If more than 1 disconnected components exist in that ROI, it is divided further into parts so that each ROI only contains 1 fibre and the orientation vector that is found is not incorrect. The improved result for the orientation of the fibres is shown in figure 4.3.

This allows the ROI window size to reduce for capturing more intricate details in the image and the algorithm results are not dependent on just a single window size inputted by the user. This implementation is unique as far as the author is aware.

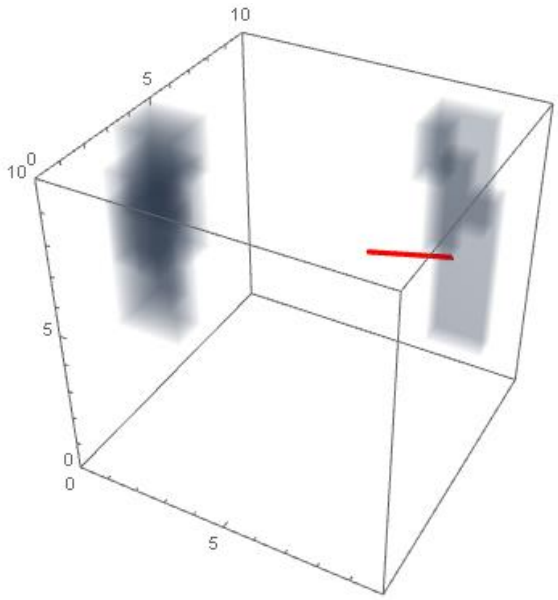


Figure 4.2: Application of the core algorithm in section 4.1.1 to a sample ROI containing two disjointed components. The calculated orientation is represented by the red line.

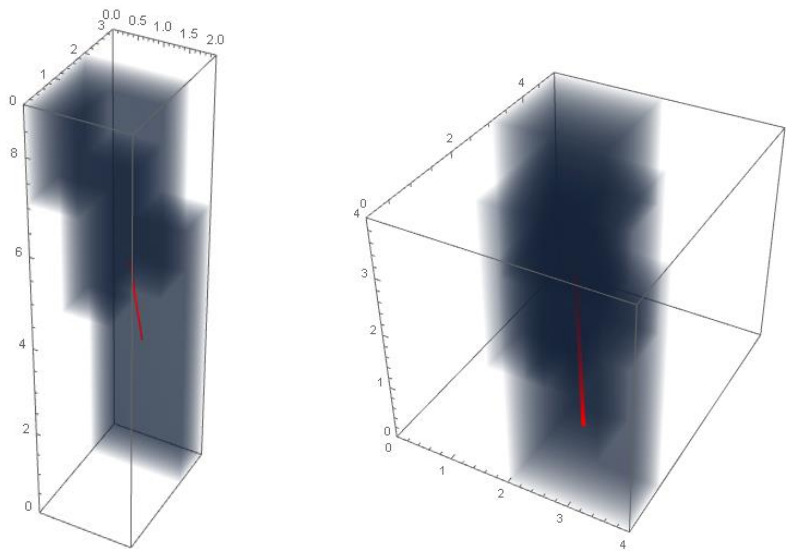


Figure 4.3: Division of the ROI in figure 4.2 into 2 parts. Application of the core algorithm presented in section 4.1.1 to each ROI. The calculated orientation is represented by the red line.

An iteration factor is the main user input which determines the base ROI window size to use. Iteration factor is the number of divisions of each dimension of the parent image to get the required ROIs. For example – a 50x50x50 image with an iteration factor of 5 will divided into 125 ROIs each with dimensions 10x10x10.

4.3 Speed performance and accuracy improvement

The suggestion for reducing the computation time is based on assigning the same orientation to all pixels in the window instead of only assigning it to the centre pixel, (latter is the case in the weighted vector summation method). This reduces the computational times by a factor of ROI window volume (typically between 5^3 and 17^3) [24].

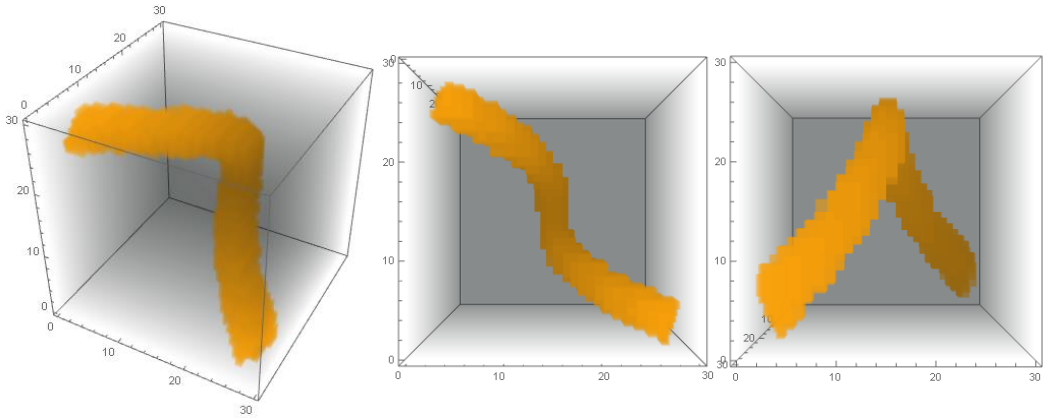
To further reduce computational times, ROIs containing more than 99% foreground pixels, more than 99% background pixels or structure of cube shape are assigned the vector $(0,0,0)$ because the orientation found will mostly be arbitrary in these cases. After the results were computed, the orientation of the foreground pixels that were assigned $(0,0,0)$ was changed to be same as the nearest non-zero orientation. Implementation of this can be found in Appendix 10.1.2.

As discussed in section 4.2 and shown in figure 4.1, the ROI window grid size influences the resulting vector field by significant margins. This could give inaccurate results.

To combat this, the algorithm was modified to compute the vector field for a range of window sizes and the resulting vector fields were averaged. This could be done without major loss in performance as it only increases the computational times linearly on average.

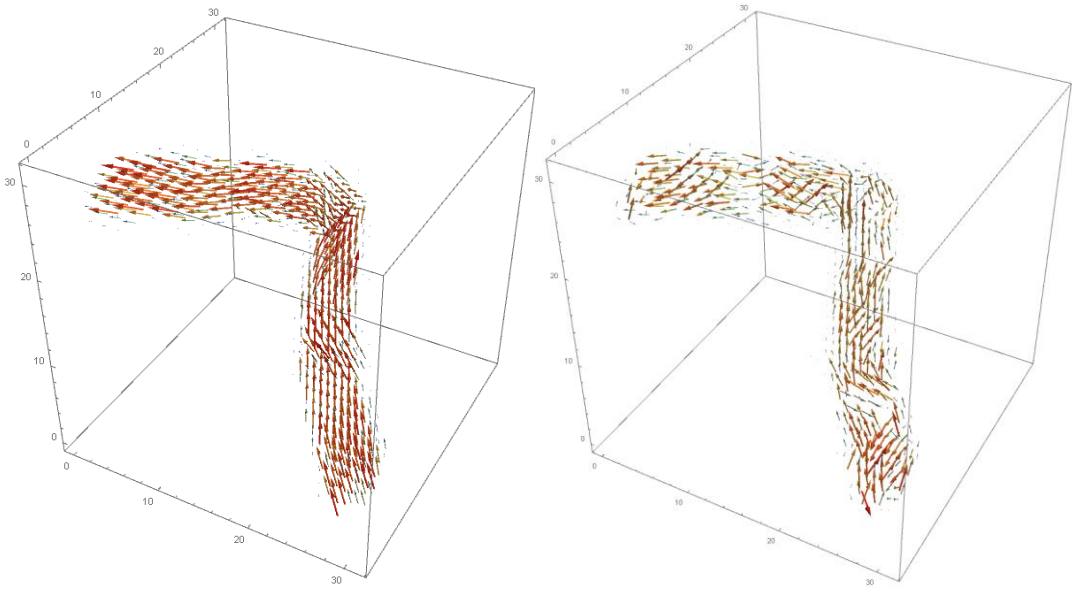
A sample 3D image containing fibre is shown in figure 4.4. The algorithm was applied using 4 different ROI window sizes. The results are shown in figure 4.5.

To show the applicability to 2D images, figure 4.7 shows the algorithm applied to a sample fibrous 2D image (shown in figure 4.6).

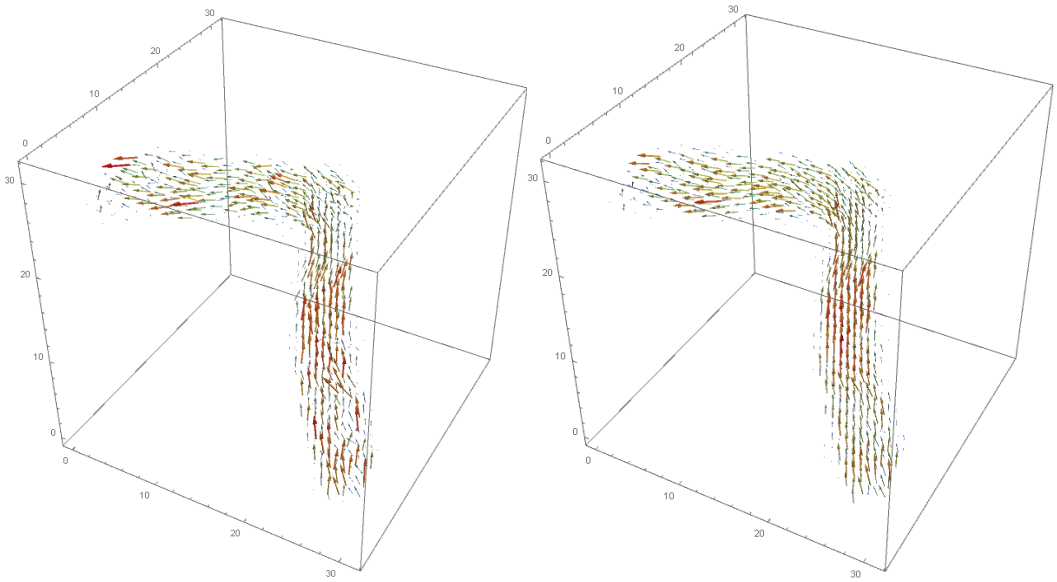


A B C

Figure 4.4: A sample binarized 3D image of size 30x30x30 containing a fibre, shown from A: Default perspective, B: Front perspective, C: Top perspective

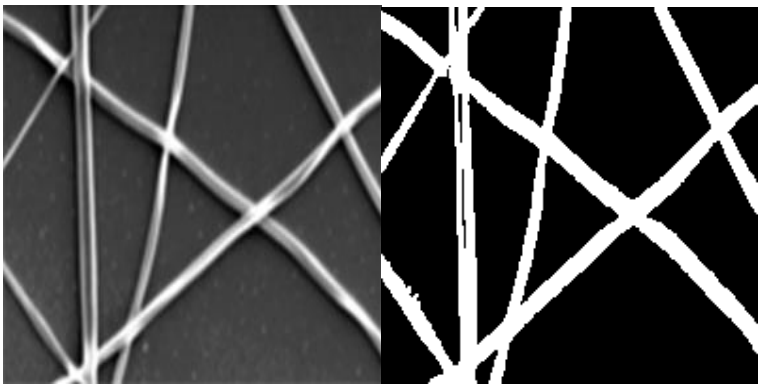


A: Iteration factor = 4, ROI size =8x8x8 B: Iteration factor = 10, ROI size = 3x3x3



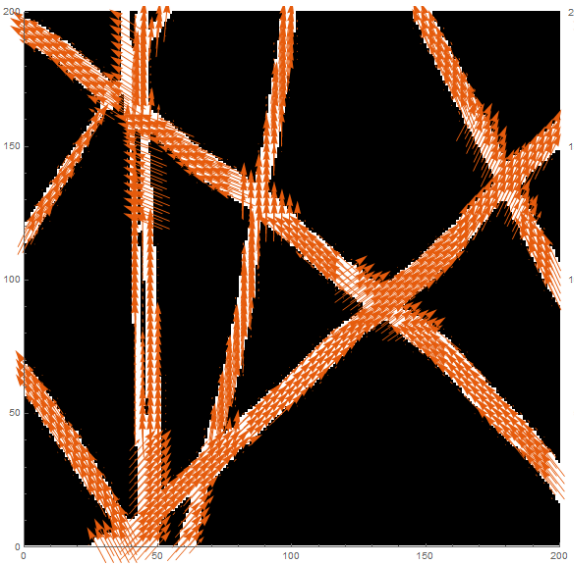
C: Averaged, iteration factors = 4,5,10,13 D: Mean Filter convolution on C (radius 1)

Figure 4.5: 3D vector fields calculated by applying the algorithm with different iteration factors to the 3D image shown in figure 4.4

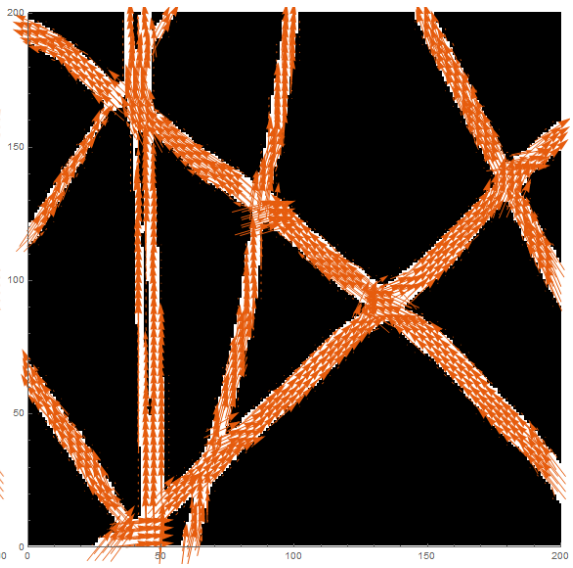


A B

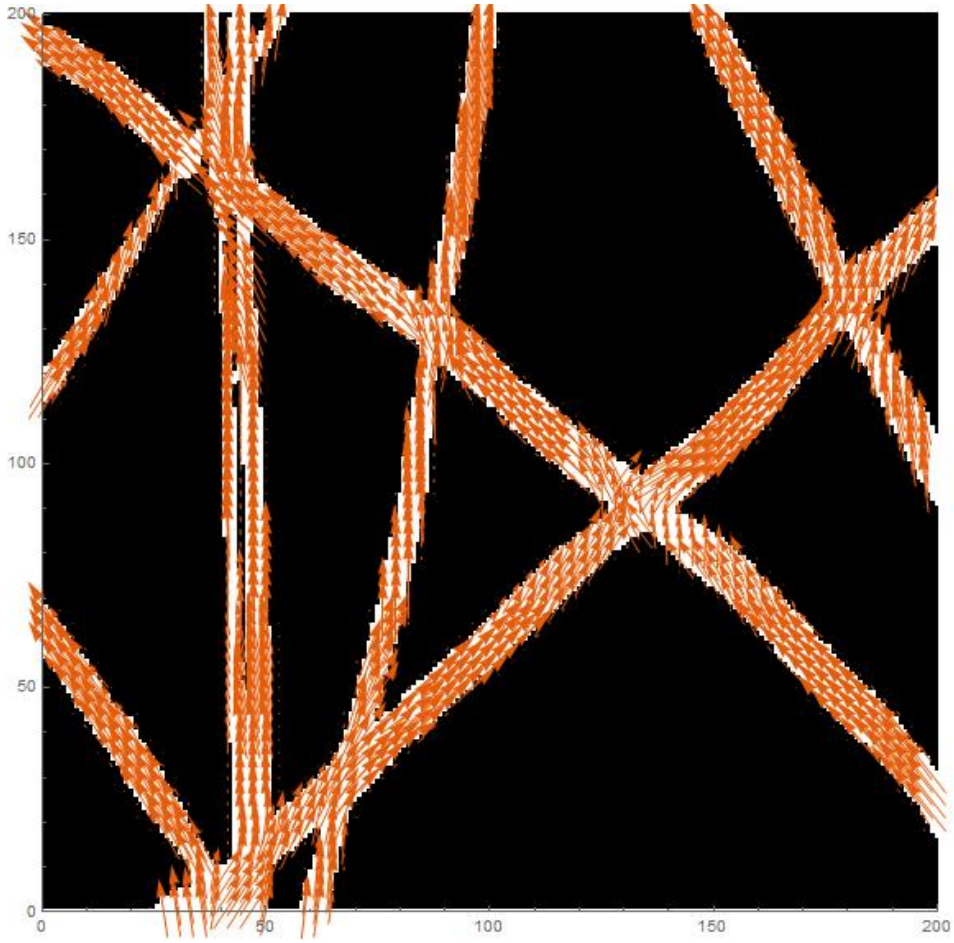
Figure 4.6: A: sample fibrous 2D image of size 200x200, B: After noise filtering using gaussian convolution filter and Binarized using Hysteresis and Otsu's thresholding



A



B



C

Figure 4.7: 2D vector field for image in figure 4.6 found using the proposed 2D Moment of inertia algorithm with A: window size 5, B: window size 17, C: Mean of vector fields with window sizes: 3,5,11,17 and smoothened using mean convolution filter

5 Implementation of other algorithms

To compare performance and accuracy of the proposed algorithm, several other algorithms were implemented in Mathematica.

5.1 Weighted Vector Summation method

This implementation was based on section 3.4.5.

The core of the implementation lies in defining all the possible vectors in the window size and weighting them for each pixel based on variance across. The 2D Implementation involved looping through all the vectors and for each vector looping through all the pixels of the image to calculate value of variance and add the weighted vector to the cumulative vector field. The conversion of circular orientation values (0° to 360°) to axial orientation values (0° to 180°) was also implemented [36]. Figure 5.1 shows the results of applying the method to the sample image shown in figure 4.6. The orientation at the centre of fibres was not found because the intensity variance is same in all directions at such points, so the result will be arbitrary.

An interpolation was included in the algorithm. Fig 5.2 shows the resultant vector field after taking the mean of vector fields calculated with different window sizes and applying interpolation of the vector field to interior pixels of fibres.

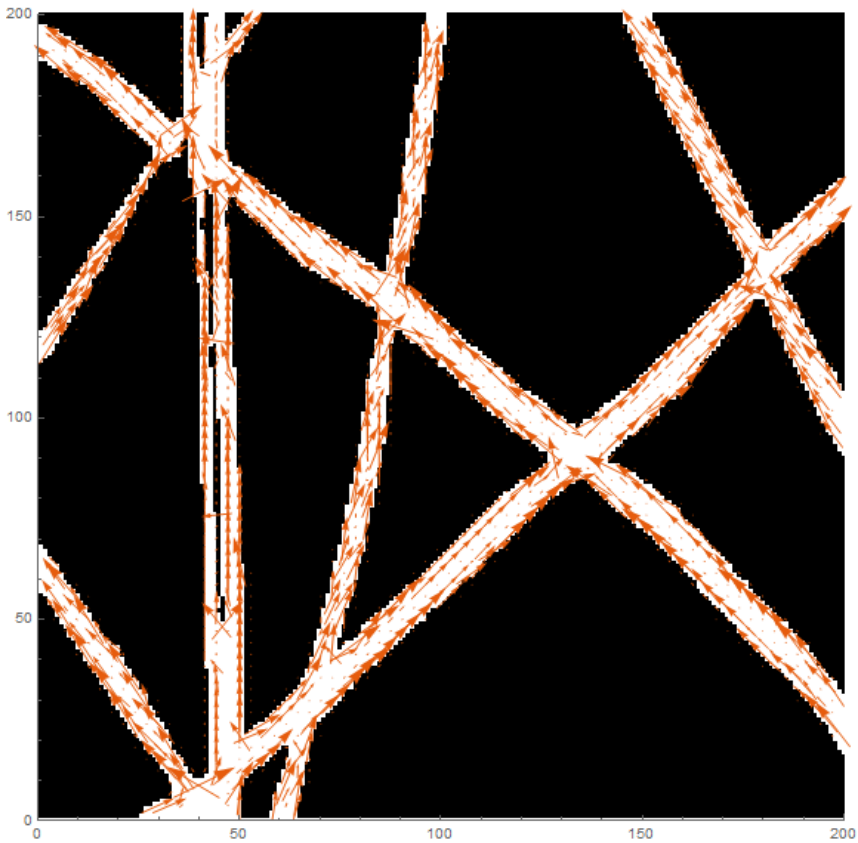


Figure 5.1: Resulting vector field after applying 2D vector weighted summation algorithm on image in figure 4.6 with window size 5x5

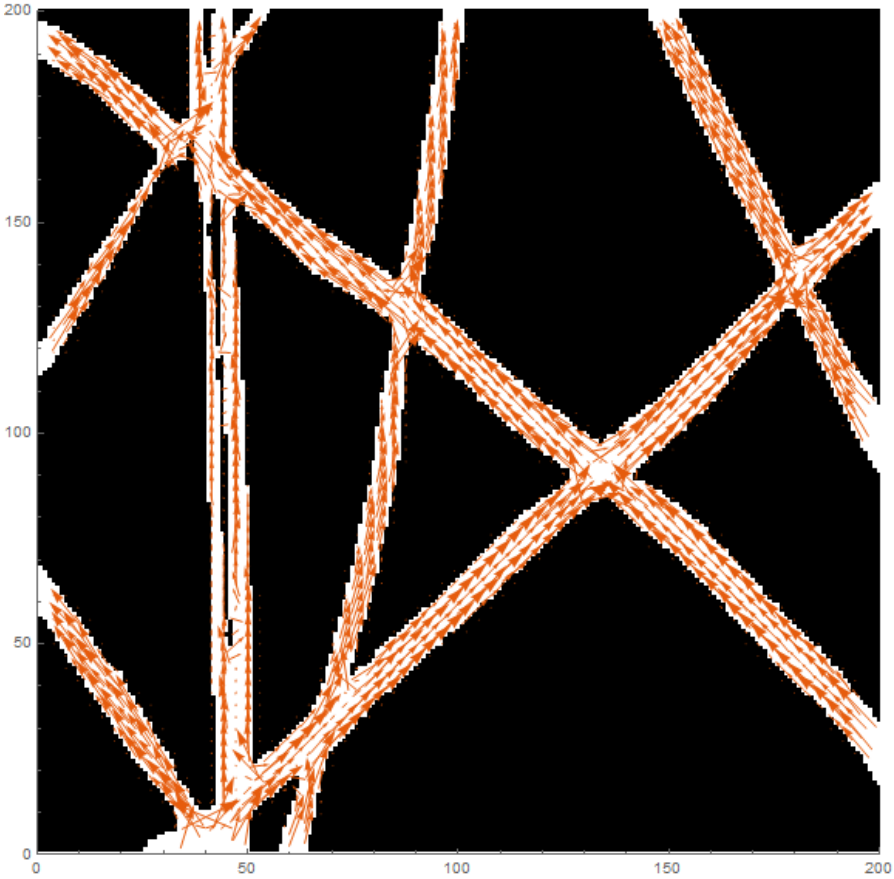


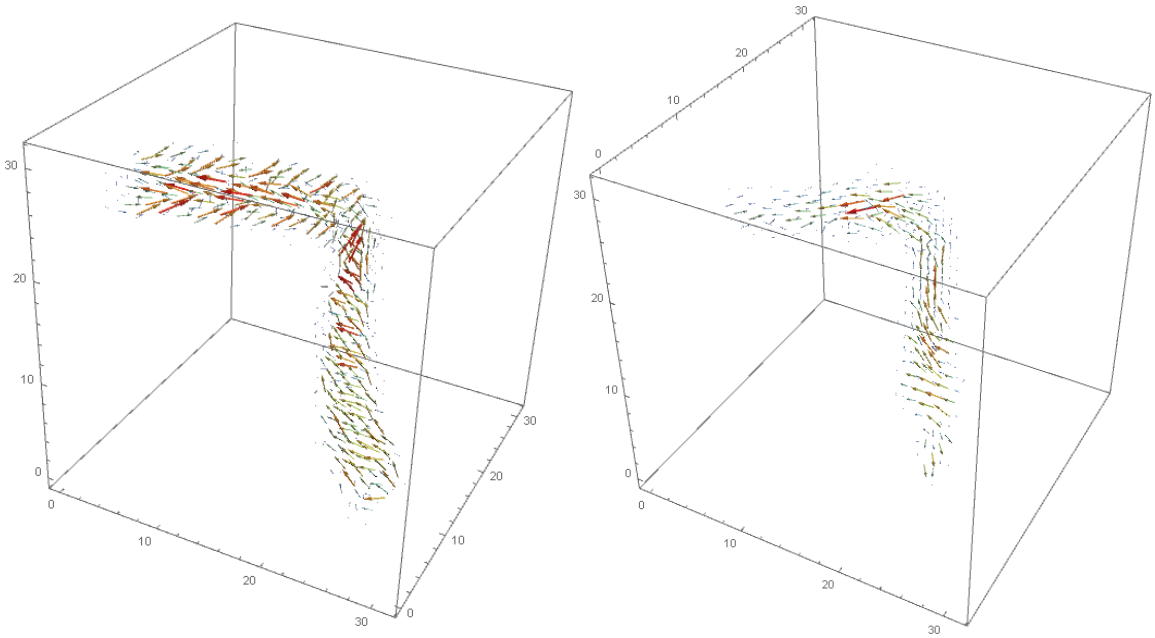
Figure 5.2: Resulting vector field after applying 2D vector weighted summation algorithm on image in figure 4.6 with window sizes 3x3, 5x5, 7x7, 9x9 – (Normalized mean and interpolated to interiors of fibres)

In 3D, the implementation involved averaging layers in the window in each x,y and z direction separately to essentially create 3 copies of the 3D image. Each layer in these copies was passed through the 2D algorithm to retrieve the orientations in that plane. Therefore, 3D algorithm for a 30x30x30 image will be about 30+30+30=90 times more computationally expensive compared to the 2D image of size 30x30. Essentially, 3 different 3D vector fields were found corresponding to each plane. These were added together to give the total vector field.

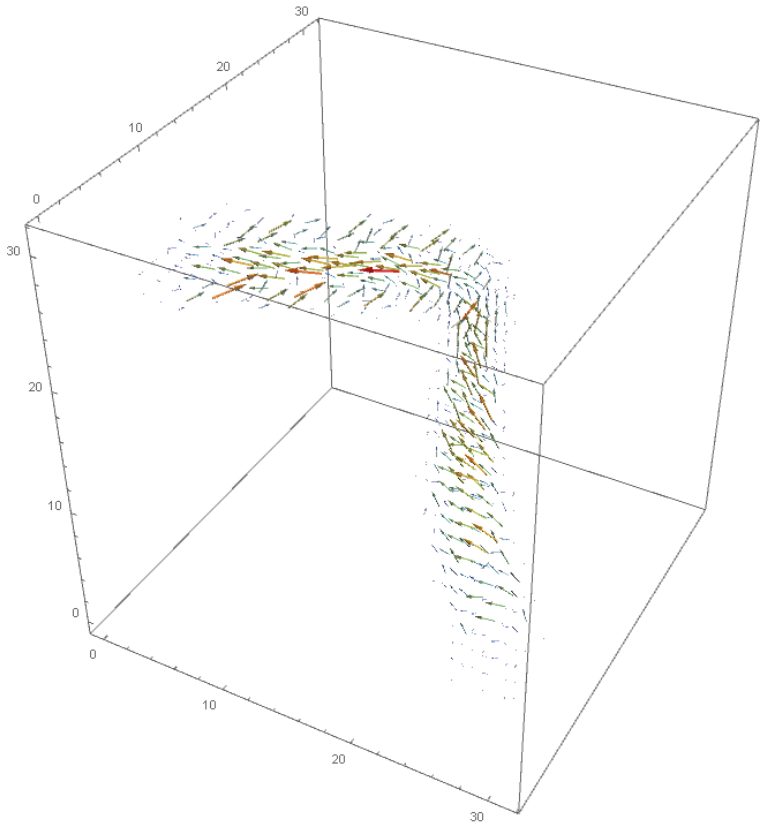
Figure 5.3 shows the application of the algorithm to the sample 3d image shown in figure 4.4.

The vectors extremely close to the edges of the image should be ignored in this implementation because of limited variance information in these regions.

The code for this method can be found in Appendix 10.1.3 and Appendix 10.1.4. It should be noted that the performance would be improved by parallelization on multiple processor cores, something which was not implemented here.



A B



C

Figure 5.3: 3D vector field for image in figure 4.4 found using 3D weighted vector summation with A: window size 5x5x5, B: window size 11x11x11, C: Mean of vector fields with window sizes: 3^3,5^3,7^3,11^3 and smoothened using mean convolution filter

5.2 Fourier Transform (Bandpass filter method)

Fourier transform for orientation distribution was implemented for 2D images only. This is a direct implementation of the *Bandpass filter on power spectrum* method presented in section 3.4.2.

The steps in algorithm are shown in figure 5.4 through figure 5.6

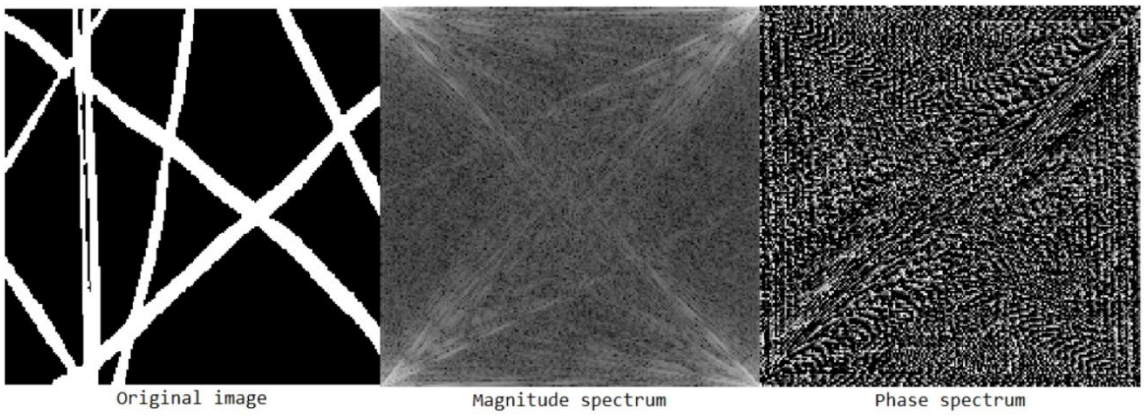


Figure 5.4: Application of 2D Discrete Fourier transform on the binary image. Magnitude spectrum, with origin at top left corner representing zero frequency

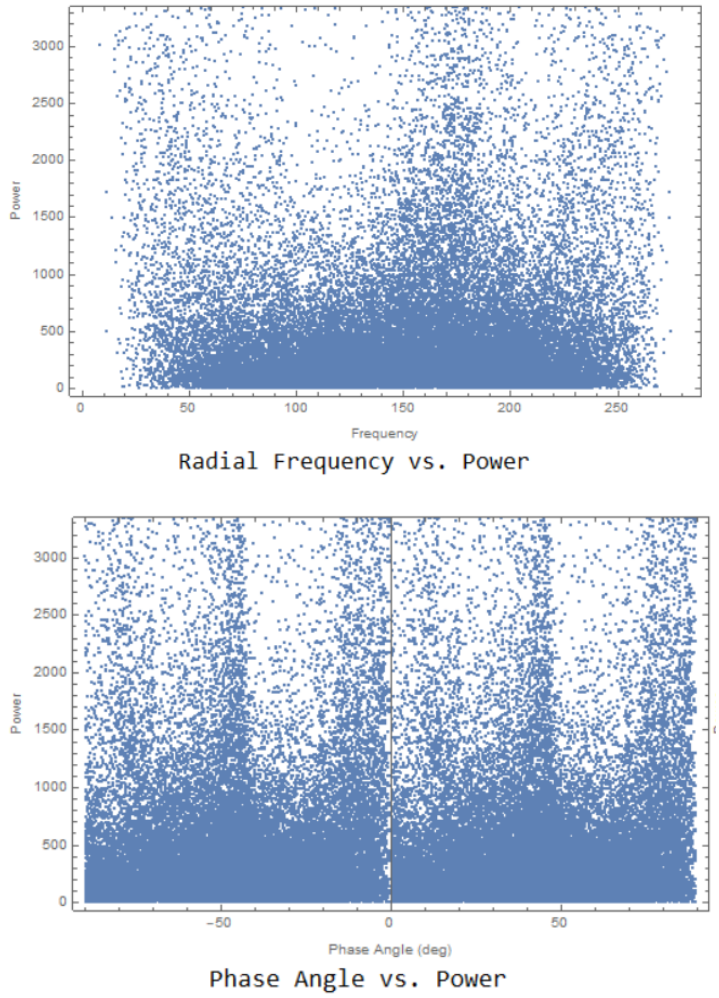


Figure 5.5: Scatter plot of Power vs Radial frequency and Power vs Phase angle respectively for the magnitude spectrum presented in figure 5.4.

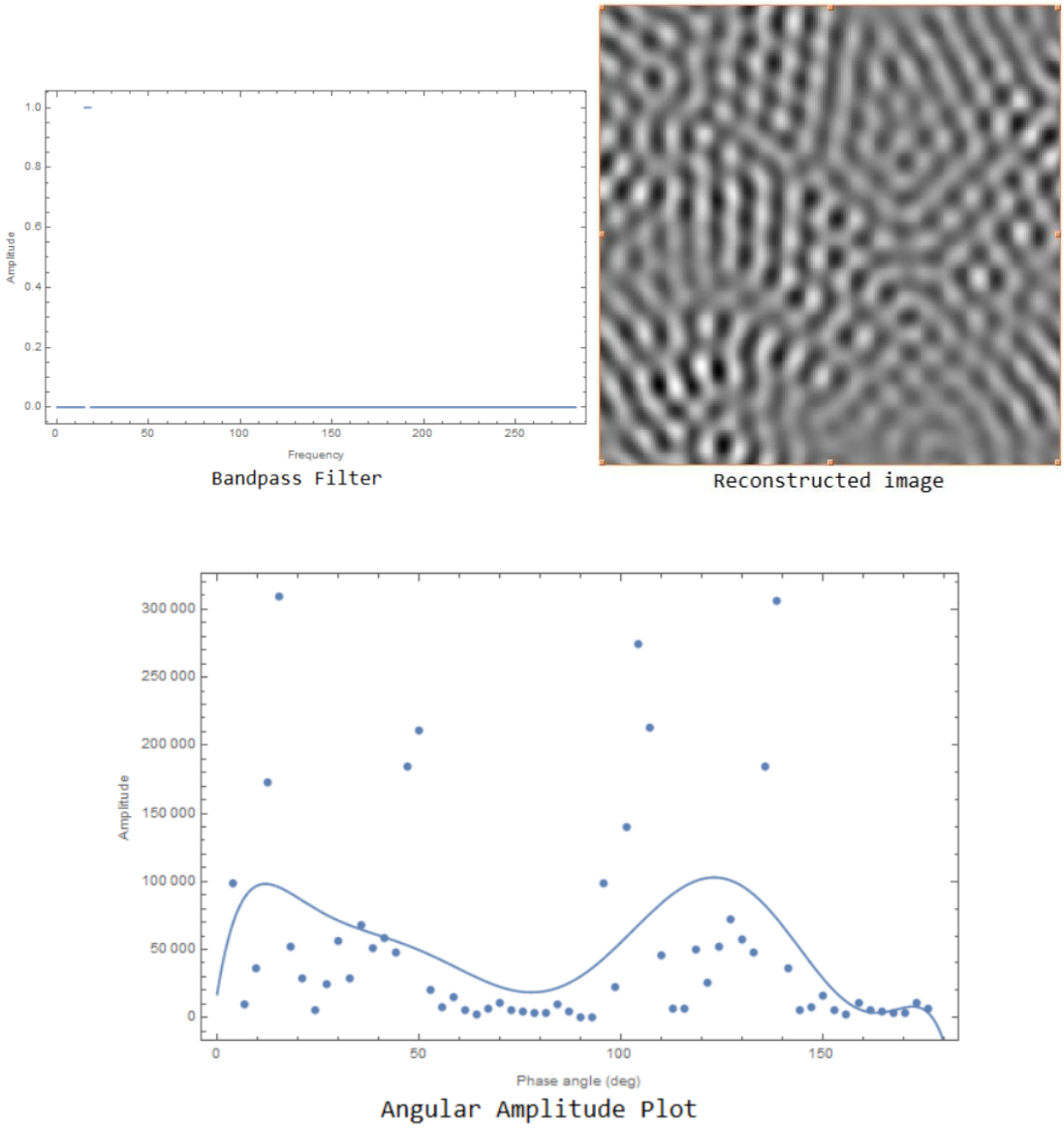


Figure 5.6: Application of a bandpass filter with frequencies corresponding to the fibre thickness. Reconstructed image after having applied bandpass filter. Angular amplitude plot after having applied bandpass filter showing approximate calculated orientation distribution.

The resultant orientation distribution is highly dependent on the image input and the bandpass filter frequencies. A uniform fibre thickness is required for the method to work perfectly and this lead to inaccuracies in orientation distribution found in figure 5.6.

This was not extended to the 3D images because:

i) The information retrieved is best suited for orientation distribution calculations of the fibres rather than for local orientation information.
ii) Applying it to on small windows would limit the frequency values to a small number of discrete possibilities which would produce distorted results
iii) Requires nearly uniform fibre length and user to input this estimated fibre thickness

Nevertheless, this was done in 2D to give a performance and accuracy comparison point as it is of one of the key methods used in orientation analysis of SHG images [36].

Appendix 10.1.5 shows the Mathematica code implementation for this Fourier analysis method.

5.3 Gradient Orientation filter

Gradient orientation filter was implemented using the Sobel kernels. This produced good results in 2D (figure 5.7). There were some points which produced undesirable orientations as shown in figure 5.7. This is because of rugged nature of lines which affects the gradient calculation heavily. This effect is more significant in application to 3D images. The vector field is initially calculated only at the edges and then later interpolated to all the foreground fibre pixels as shown in figure 5.8.

The results were completely inaccurate for 3D analysis. This is because the orientation of gradient is affected abruptly by small local features such as edge steps which were highlighted more in the smaller resolution 3D window compared to the better relative resolution in the 2D image. Figure 5.9 shows the resultant vector field when gradient orientation filter was applied. This shows why the simpler 2D analysis algorithms are not always easily extendable to 3D.

Appendix 10.1.6 shows the Mathematica code implementation of the gradient orientation filter in 2D.

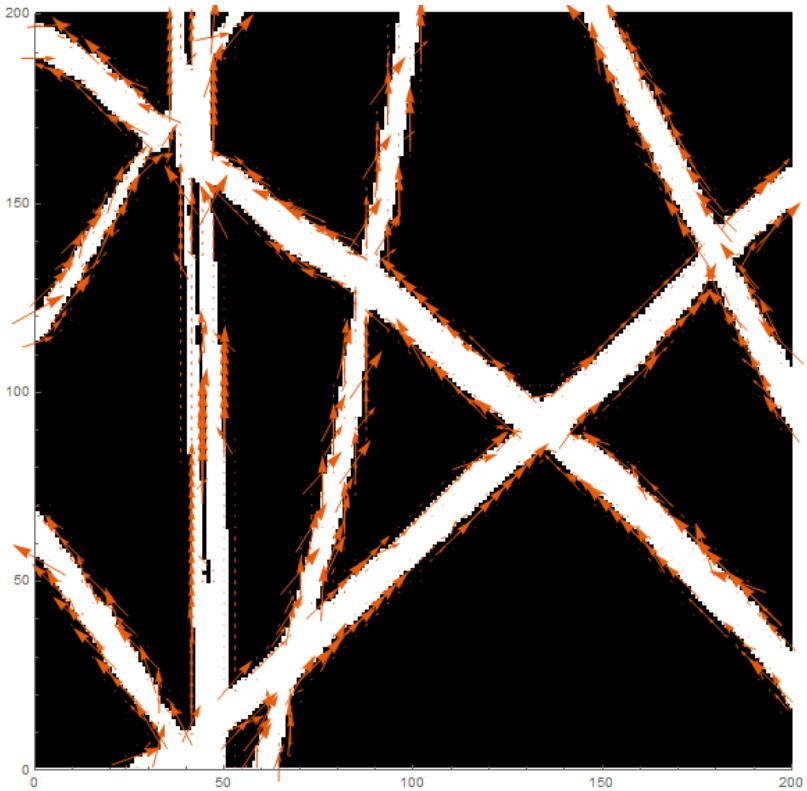


Figure 5.7: Resulting vector field after applying a separable Sobel kernel orientation filter on image in presented figure 4.6 with kernel radius of 1

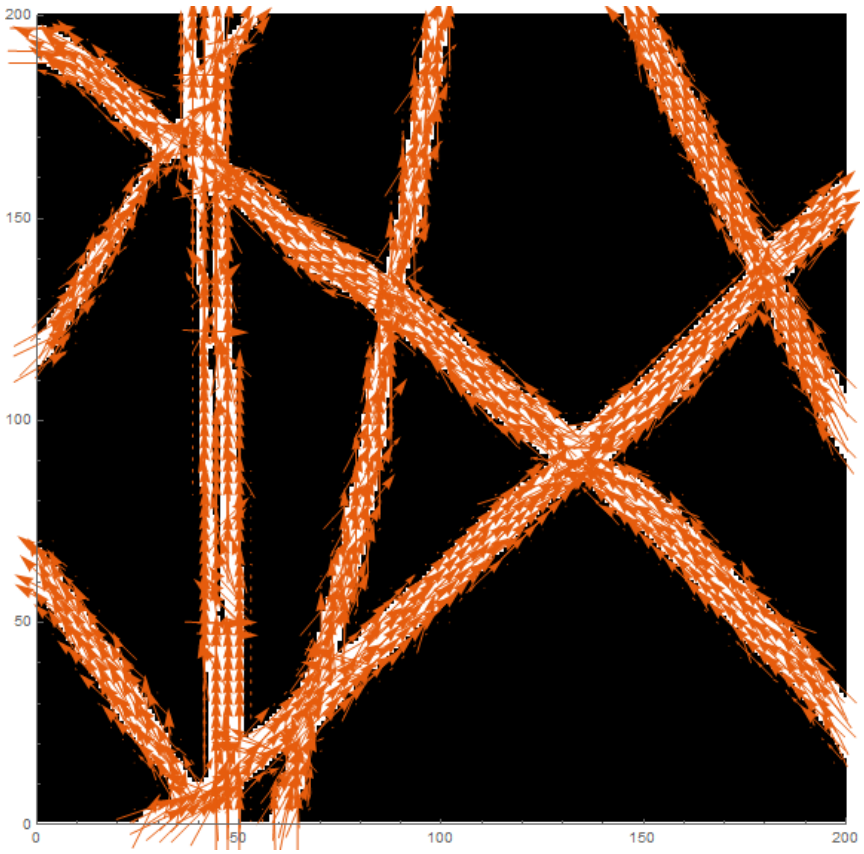


Figure 5.8: Resulting vector field after interpolation of vector field in figure 5.7 to interior of fibres

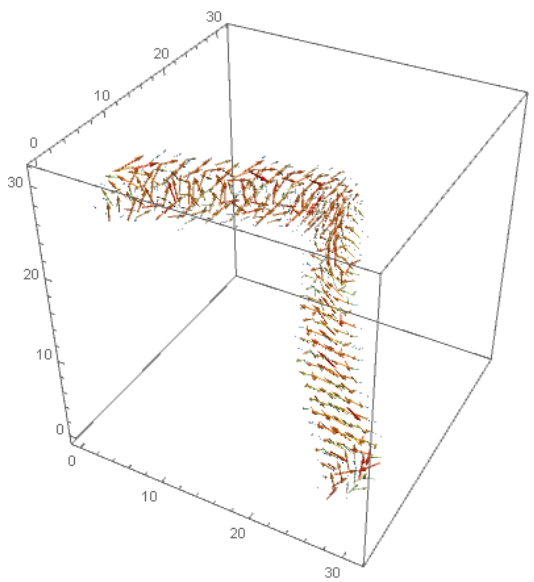


Figure 5.9: Resulting 3D vector field after interpolation of vector field calculated using 3D Sobel Gradient orientation filter applied to image presented in figure 4.4

6 Discussion

In this paper, the following algorithms were compared for accuracy, performance and suitability to different situations.

1. Proposed algorithm: Moment of inertia/ Principal component analysis (2D/3D)
2. Weighted Vector summation (2D/3D)
3. Fourier transform (2D only)
4. Orientation gradient filter (2D only)

6.1 2D algorithm analysis

The 2D algorithms were applied to computer generated images with known orientation distributions. Figure 6.1 shows one such image, along with the calculated vector field.

Orientation distribution was determined by converting each 2D vector into an angle and rounding it to a discrete value (based on required angle resolution). Figure 6.2 and figure 6.3 show 2 such angle distributions for the image in figure 6.1.

These orientation calculations are useful in calculation of error.

Average absolute difference between orientations of points in the ideal case and the experimental case can be calculated using Eq. 30 (X is a pixel location)

$$Error = \sum_{X} \left\| \theta_{ideal}(X) - \theta_{exp}(X) \right\| \qquad \text{Eq. 30}$$

This is an indication of the error and hence the accuracy of the algorithms.

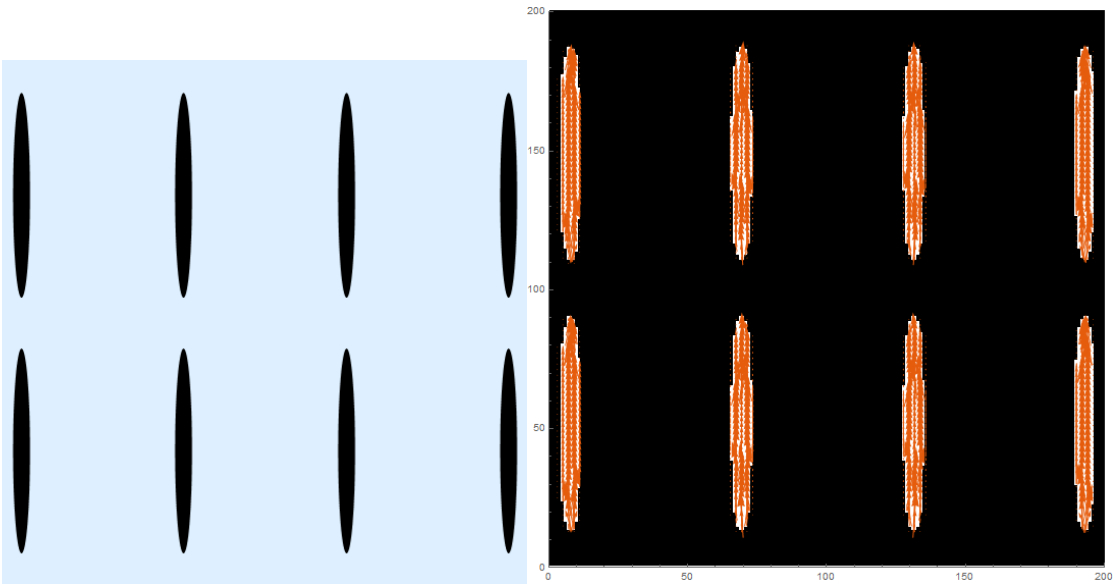


Figure 6.1: A sample computer-generated image with known orientation distribution (peak at 90 degrees, 0 elsewhere) and the vector field (right side) calculated using 2D moment of inertia algorithm

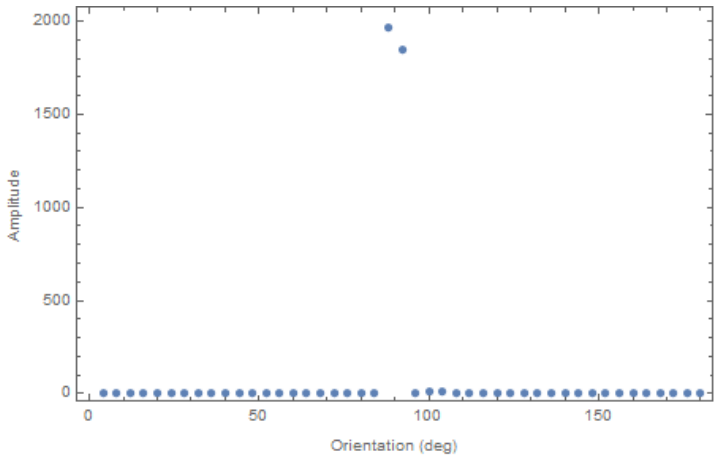


Figure 6.2: Orientation distribution of the image shown in figure 6.1, (2D Moment of inertia algorithm)

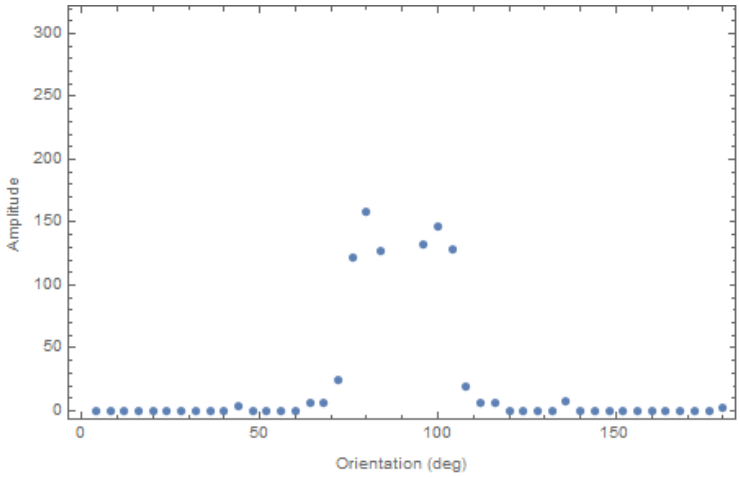


Figure 6.3: Orientation distribution of the image shown in figure 6.1, (2D Vector Summation method)

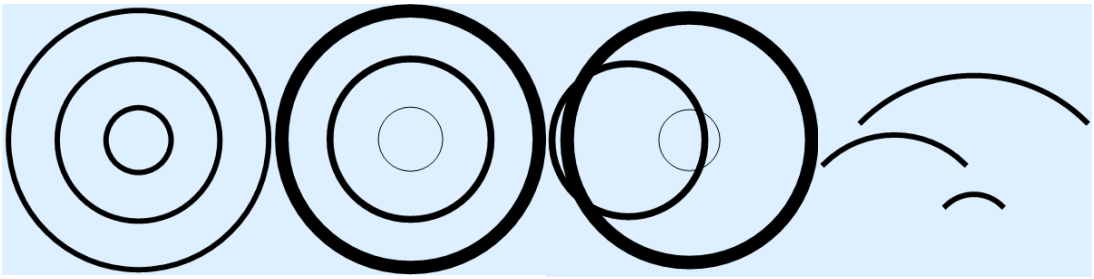


Figure 6.4: Sample computer generated images. From left to right: Uniform thickness circles, non-uniform thickness circles, overlapping circles, Uniform thickness arcs

The algorithms were applied to different images shown in figure 6.4. The resulting speed performance and accuracy graphs can be seen in figure 6.5 and figure 6.6 respectively

The proposed algorithm has the lowest average error among the methods investigated. (figure 6.5). Even the vector summation 2D method was not as accurate. In terms of speed performance, the proposed moment of inertia-based algorithm was the 2nd best, after gradient filter. However, the speed was still significantly better (as much as 10 times faster) than both Vector Summation and Fourier analysis. This was expected because of the performance tweaks explained in section 4.2

The Fourier transform algorithm was also found to be very inaccurate for anything which was not a uniform orientation distribution, for example the arcs (shown in figure 6.4). Also, the speed performance of this method was as much as 9 times worse than the proposed algorithm. (figure 6.6)

The gradient filter had the poorest accuracy, even after experimenting with different kernel radii. The average error for gradient filter was as much as 22.5 degrees (which is 4.5 times worse than the proposed algorithm) (figure 6.6)

Overall, the proposed algorithm performed well both in terms of accuracy and speed.

For more graphs depicting effects of ROI sizes, fibre thickness, image size and fibre thickness on the accuracy and speed performance of the proposed 2D algorithm and comparisons with 2D weighted vector summation algorithm, please see Appendix 10.2.

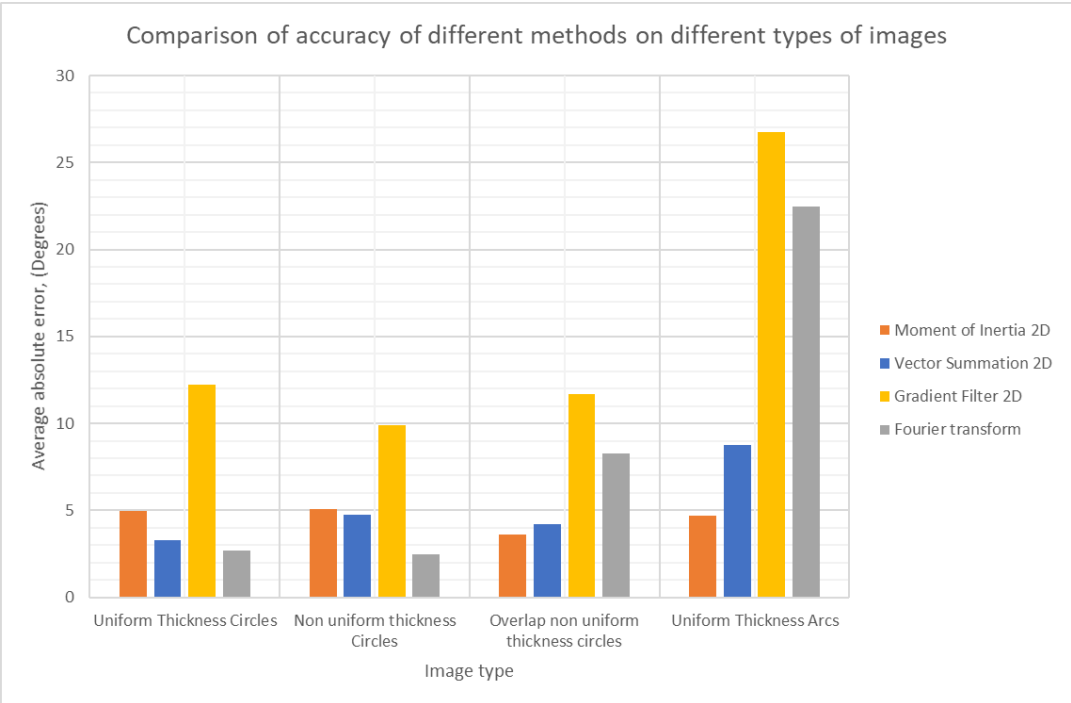


Figure 6.5: Comparison of orientation distribution accuracy of different methods, on different types of images (shown in figure 6.4), (Lower error value represents better accuracy)

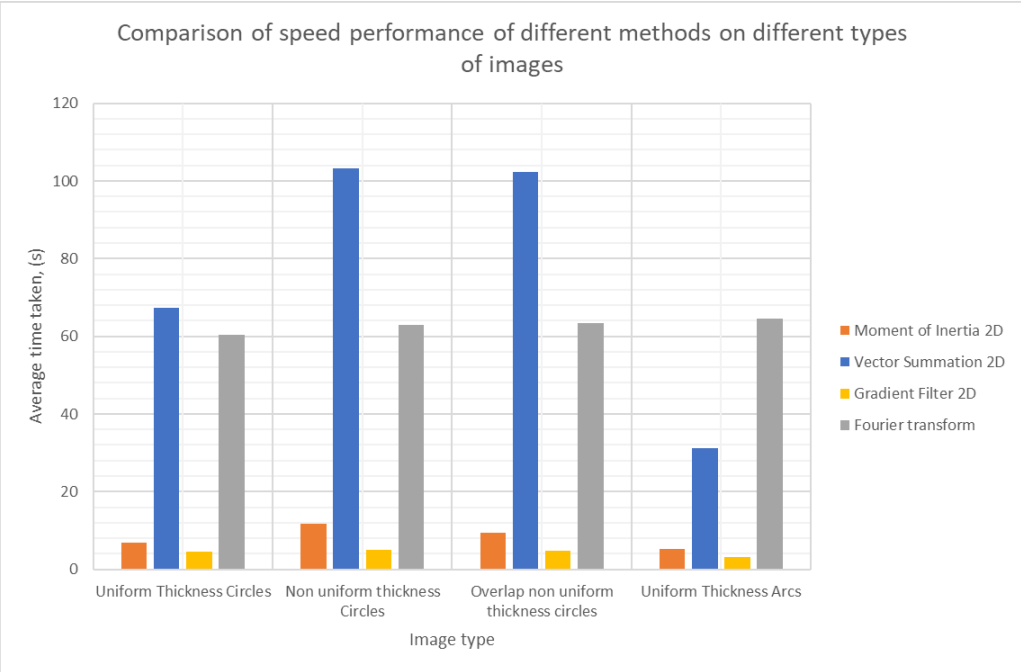


Figure 6.6: Comparison speed performance of different 2D algorithms, on different types of images (shown in figure 6.4), (Lower time value represents better speed performance)

6.2 3D algorithm analysis

Sample images with known orientation distribution and varying fibre thicknesses, varying fibre densities and varying total volume were generated (examples shown in figure 6.7 and figure 6.8).

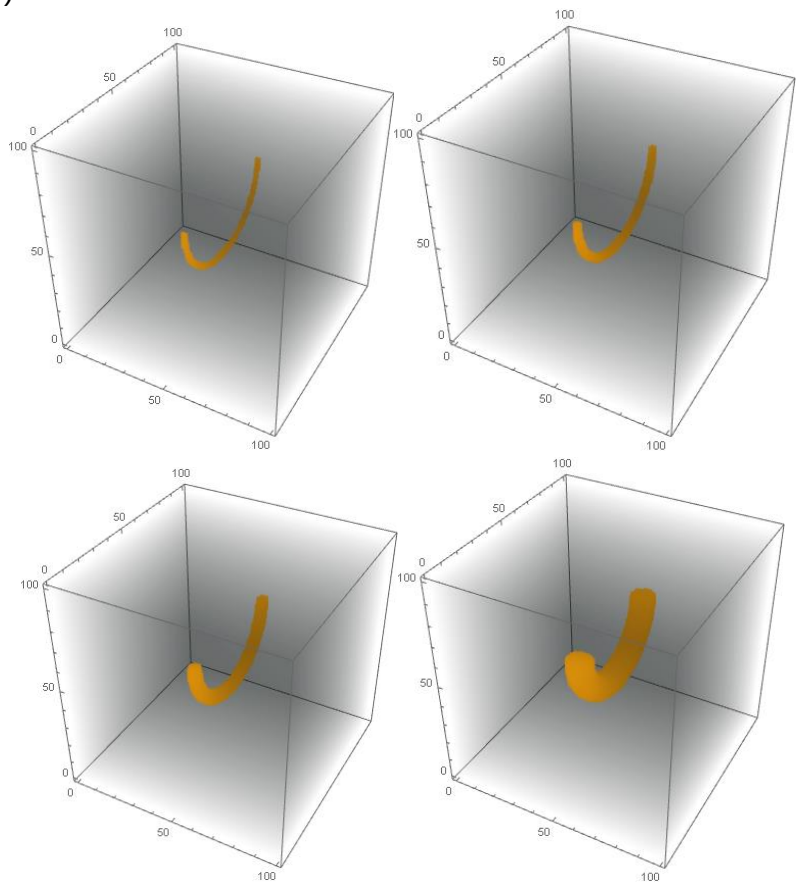


Figure 6.7: Sample computer-generated 3D images of a fibre with varying thicknesses (image size 100x100x100)

Orientation in 3D is a combination of 2 angles (θ, φ), as discussed in section 3.4.5. These were calculated from each 3D vector and the orientation distribution plotted on a spherical surface 3D plot. An example of this is shown in figure 6.9. These orientations are useful in calculating the error (using Eq. 30)

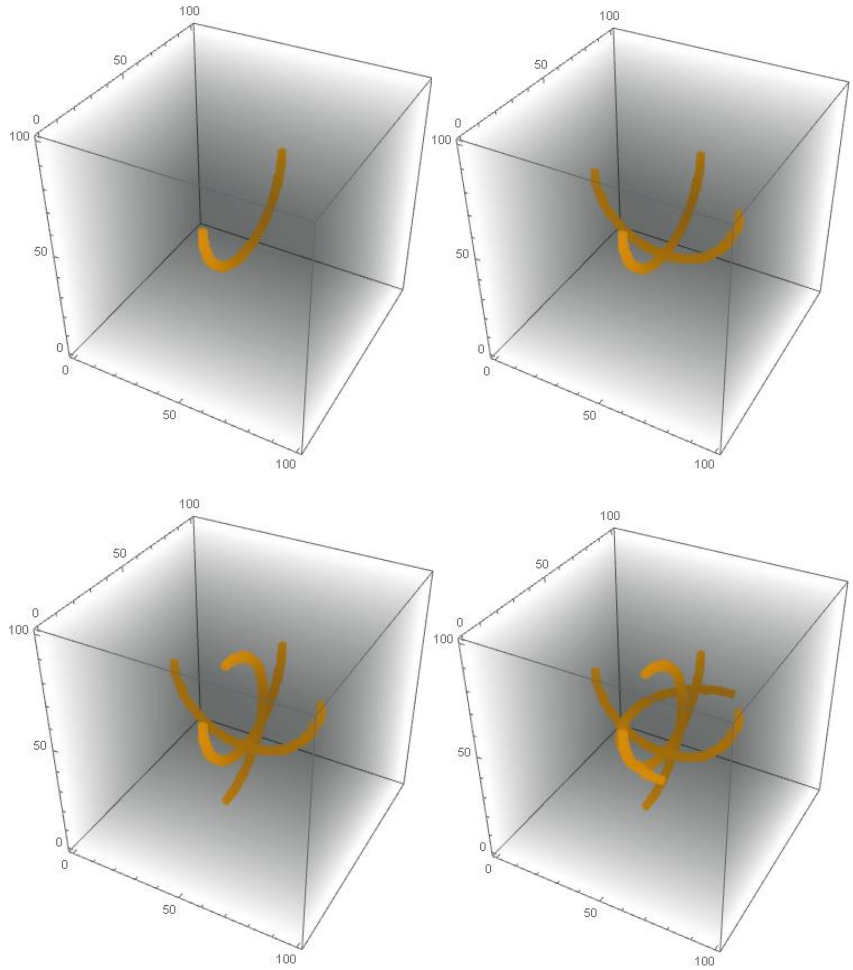


Figure 6.8: Sample computer-generated 3D images containing different number of fibres oriented differently (image size 100x100x100)

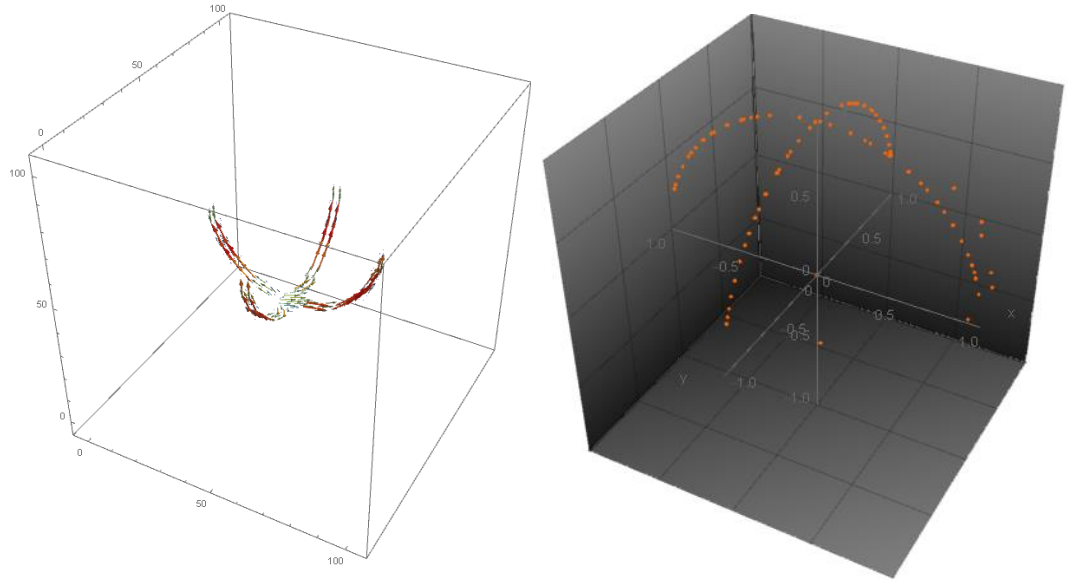


Figure 6.9: (Left) 3D Vector field generated for image in figure 6.8 using proposed moment of inertia algorithm, (Right) Orientation distribution of the vector field plotted on a sphere surface

6.2.1 Overall comparison

Accuracy of the proposed moment of inertia algorithm is shown in comparison with weighted vector summation 3d method, as a function of image volume, in figure 6.10. The error is lower for the proposed algorithm which underlines its effectiveness.

Similarly, the speed performance was plotted as a function of image volume and compared with that of weighted vector summation 3D (figure 6.10). Again, the proposed algorithm was as much as 10 times faster than the weighted vector summation method. This was expected because of the tweaks made to the algorithm to improve the performance, as discussed in section 4.2.

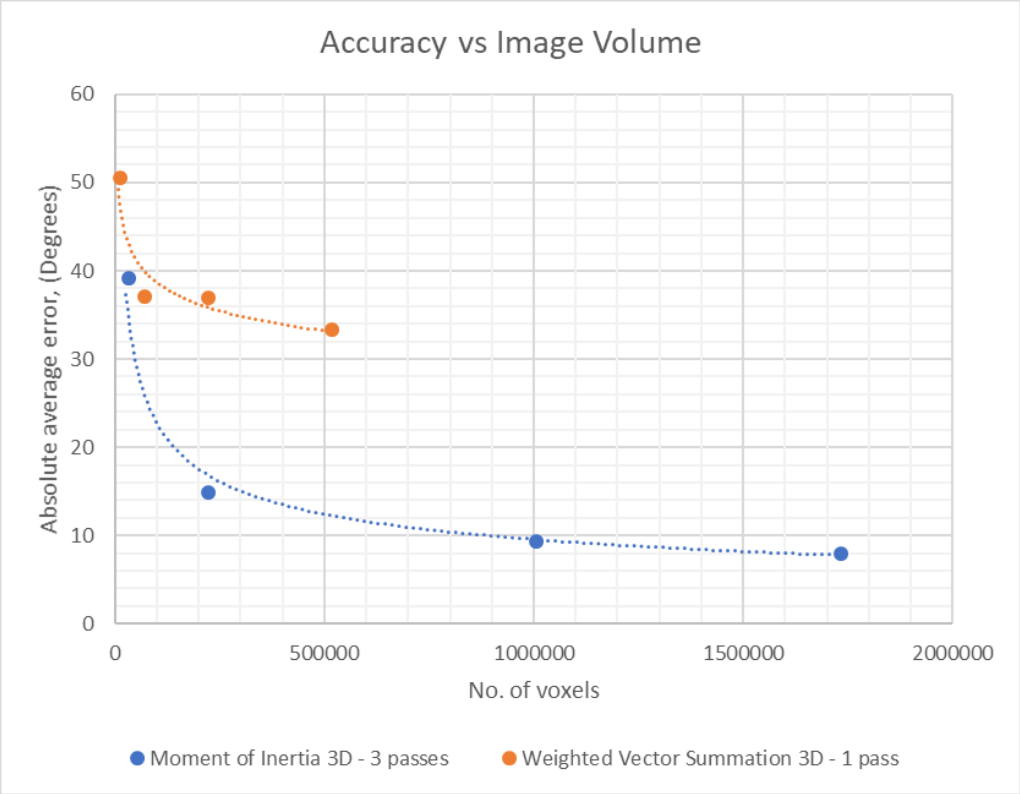


Figure 6.10: Plot of orientation distribution accuracy with respect to image volume, calculated using proposed moment of inertia 3D algorithm and the weighted vector summation algorithm 3D. (Lower error value represents better accuracy)

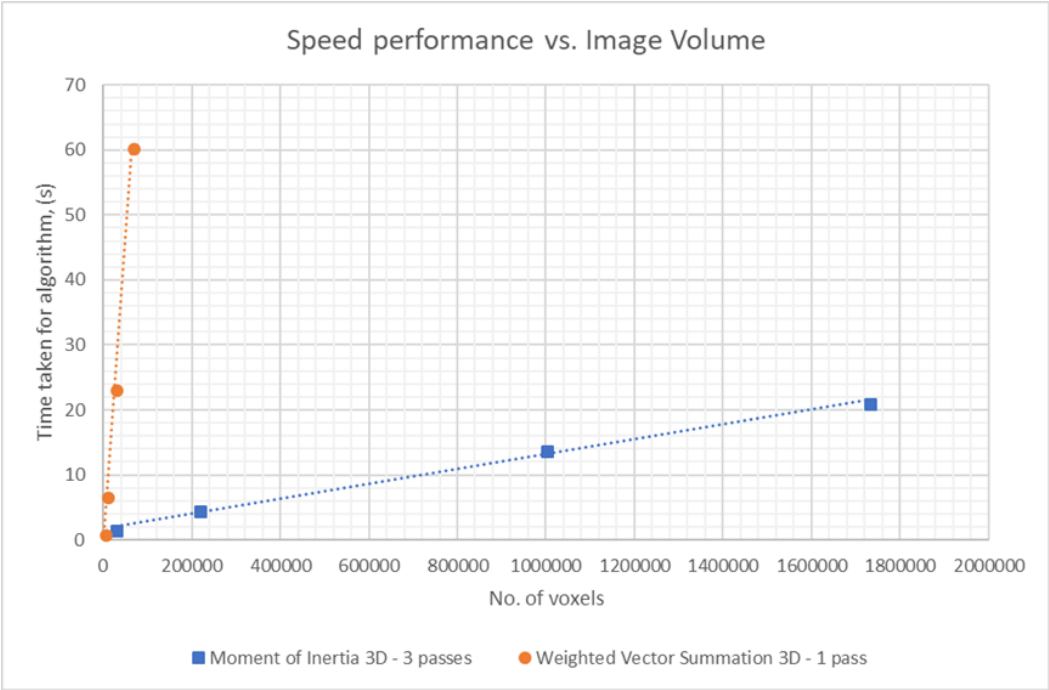


Figure 6.11: Plot of speed performance with respect to image volume, calculated using proposed moment of inertia 3D algorithm and the weighted vector summation algorithm 3D. (Lower time value represents better speed performance)

6.2.2 ROI window size effects

The ROI window size is controlled using an iteration factor, as explained in section 4.2.2. This iteration factor is inversely proportional to the ROI window size.

From figure 6.12, we can conclude that a lower iteration factor, or a larger window size, is better for the speed performance of the algorithm. This was not totally expected because it was expected that a larger window size would mean bigger matrices being used in the moment of inertia calculation and hence worse performance. Evidently, this effect does not overpower the effect due to reduced number of iterations.

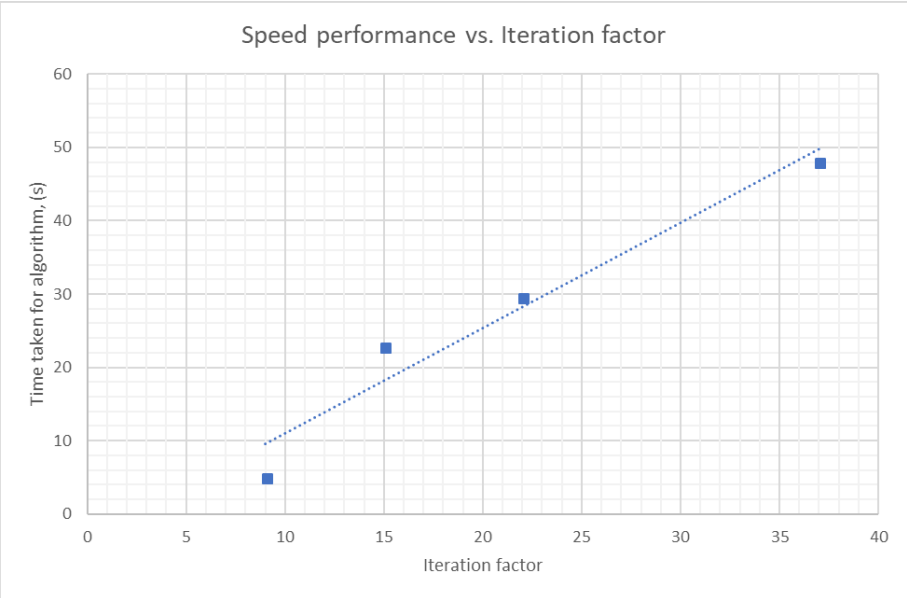


Figure 6.12: Plot of speed performance with respect to image volume, calculated using proposed moment of inertia 3D algorithm (Lower time value represents better speed performance)

6.2.3 Fibre thickness effects

Figure 6.13 shows the effects of fibre thickness on accuracy. The accuracy improves with thinner fibres because moment of inertia is more well defined for elongated objects rather than rounder objects.

Surprisingly, the speed performance does not change with increasing fibre thickness, possibly because the increased number of foreground pixels is not significant enough for a small 3d image of size 100x100x100.

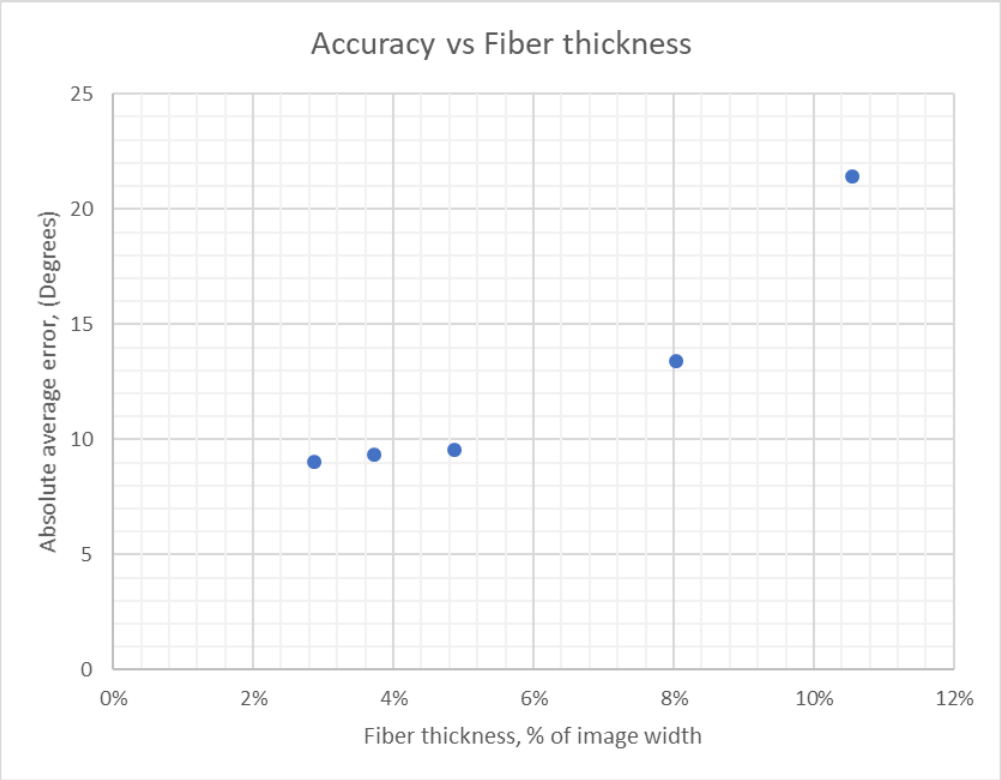


Figure 6.13: Plot of orientation distribution accuracy with respect to fibre thickness, calculated using proposed moment of inertia 3D algorithm (Lower error value represents better accuracy)

6.2.4 Fibre density effects

From figure 6.14, it can be concluded that the speed performance does become worse with increasing number of fibres. However, this increase is linear and significantly small as shown by the line of best fit with a low gradient of 1 sec/fibre

Therefore, the proposed algorithm will produce results at good speed regardless of the fibre density in the image.

Figure 6.15 shows the effects of fibre density on accuracy. Clear, the error value increases with increase in fibre density. This was expected because the overlap between fibres also increases (as seen in the images of figure 6.8)

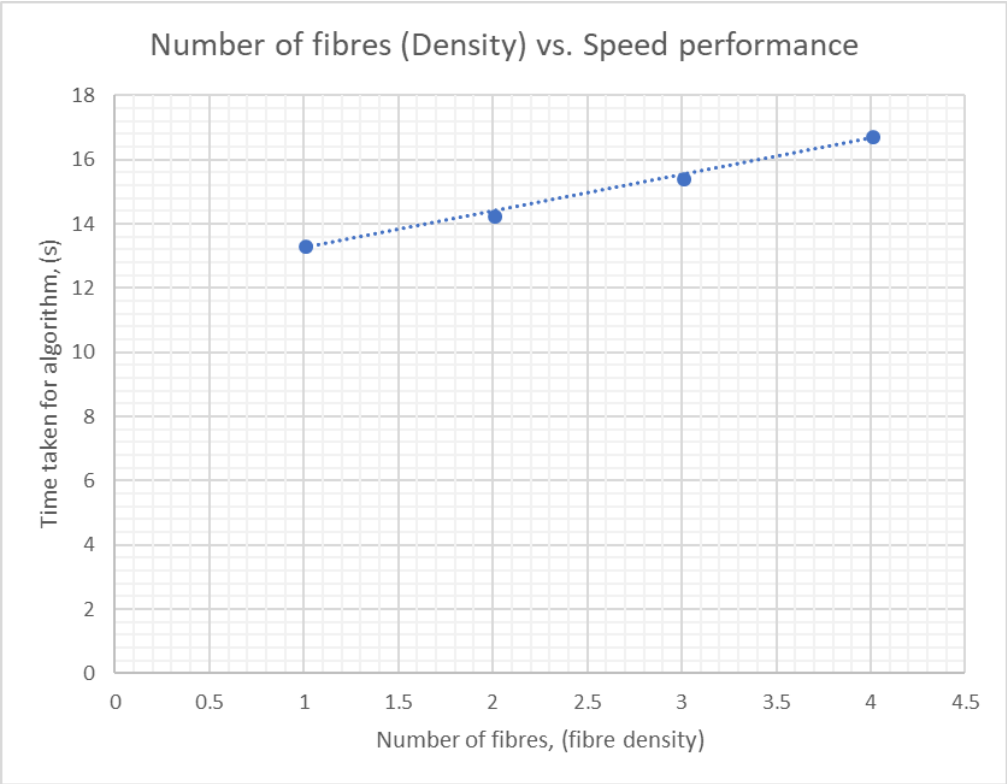


Figure 6.14: Plot of speed performance with respect to fiber density, calculated using proposed moment of inertia 3D algorithm (Lower time value represents better speed performance)

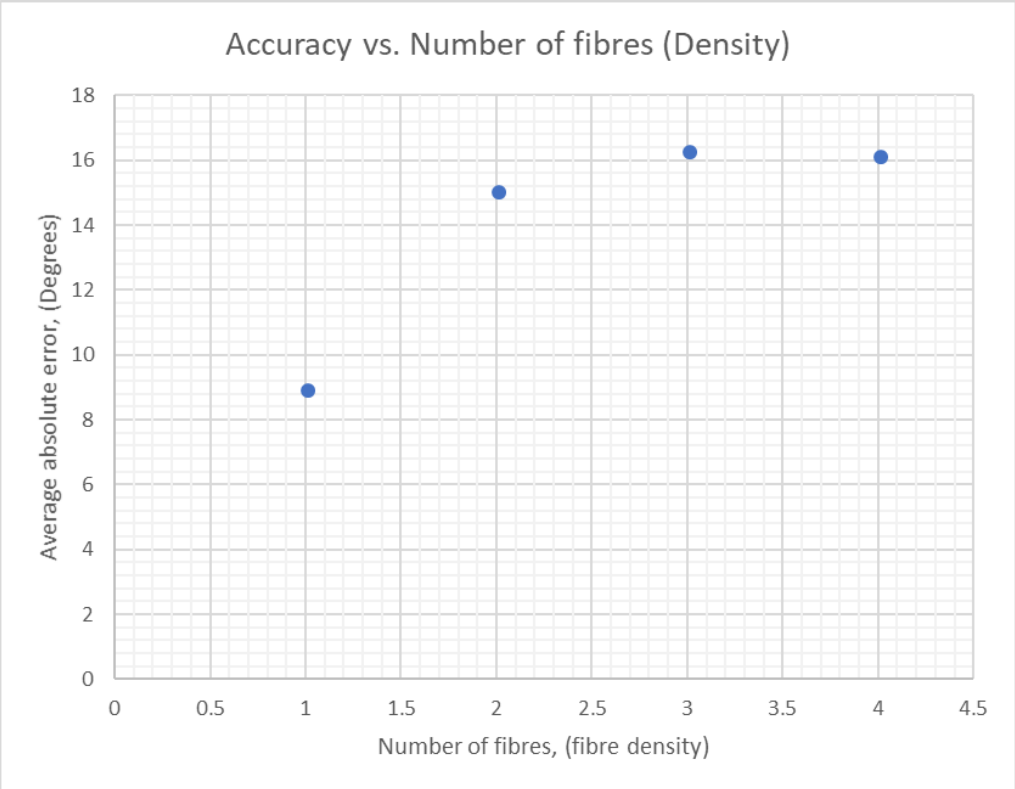


Figure 6.15: Plot of orientation distribution accuracy with respect to fibre density, calculated using proposed moment of inertia 3D algorithm (Lower error value represents better accuracy)

7 Example Application

A multi-image tiff file (~570 Mb) containing the optical section series of central nervous system (dendritic tree structure, medium spiny neurons) of a mouse was used as a sample dataset to apply the algorithm on. The original volume of this dataset was $1440 \times 1782 \times 233$ voxels [40].

The image dimensions in x and y axis were reduced to 25% to make the volume $360 \times 446 \times 223$ voxels. This was done to reduce the time taken for algorithm and to make image have a relatively similar size in each dimension, so that the 3D effect can be more prominent. The 3D image representation is shown in figure 7.1.

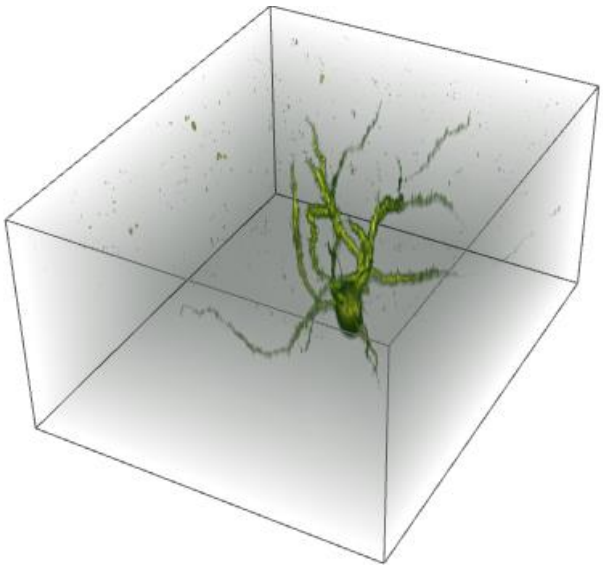


Figure 7.1: 3D image representation of 2D stack image data of mouse central nervous system [40]. Dimensions: $360 \times 446 \times 223$

The image data after noise filtering and binarization (mean filtered, and binarized using hysteresis with Otsu's thresholding) is shown in figure 7.2.

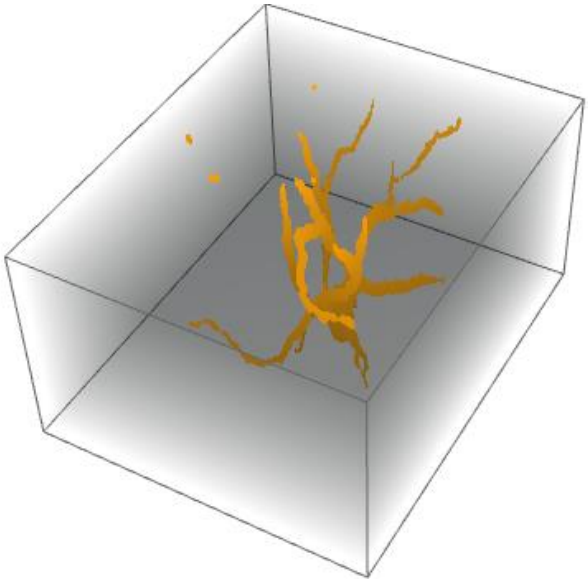


Figure 7.2: Noise filtered and binarized version of 3D image shown in figure 7.1

After applying a 4-pass moment of inertia algorithm with iteration factors (9,15,22,29), the resulting vector field is shown in figure 7.3

From visual inspection of the vector field, the orientations of the thin or elongated structures was captured relatively accurately. A clear orientation for the rounder central structure could not be identified, which is as expected. A possible improvement to this method would be to produce a confidence value for each vector produced in the image so that rounder structures of the image can either be ignored or at least the user is alerted of existence of possible spurious data in regions with low confidence value.

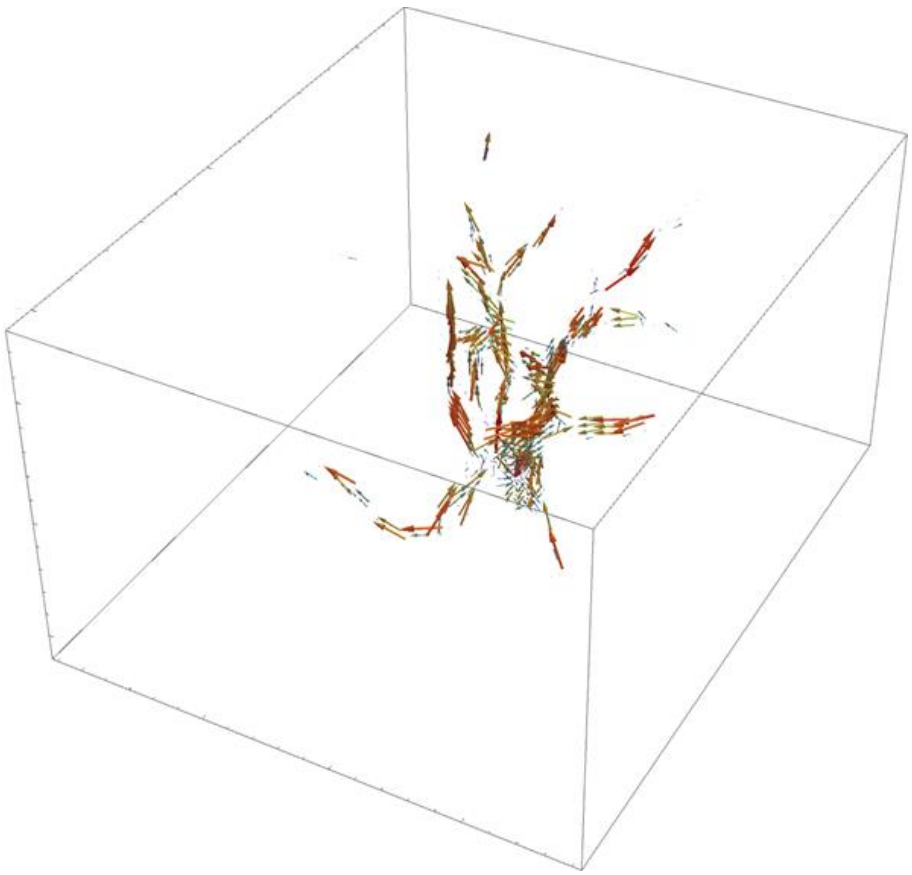


Figure 7.3: 3D vector field representing the orientations of the structures of the 3D image shown in Fig 7.1. Obtained using the proposed moment of inertia algorithm. (Vector points shown for 1/10th of the voxels only to make the visualization clearer)

8 Conclusion

An algorithm for orientation determination of microstructures in 3D images was developed. This was based on principal component analysis of moment of inertia tensor with tweaks for improved accuracy and better performance. The algorithm was analysed by application on computer generated images and compared with other common methods, especially an implementation of the weighted vector summation method. It was found that the proposed algorithm outperformed others both in terms of speed performance and accuracy.

It was reported that success in finding orientations of microstructural features accurately and quickly depends heavily on the image dataset itself including microstructure feature density & overlap, fibre thickness and image volume/resolution.

Prior to 3D algorithm formulation, the algorithm was implemented for 2D image applications and similarly found to be accurate and fast when compared with other methods.

Image filtering and binarization were also studied to make images fit for use with the algorithm and it was found that different methods are useful for different types of images with hysteresis and canny filter being especially accurate.

The method was implemented in Wolfram Mathematica and integration with other software systems could be considered for an improved workflow in biomedical research environments. Moreover, the resultant 3D vector field could be used in a FEA study to fully appreciate whether the algorithm is successful.

Lastly, more computing power is needed to apply the algorithm to larger datasets. This could also be done with an improved implementation, for example making use of parallelization, and is a good avenue for further research.

9 Reference List

[1] J. Kyu Gahm, N. Wisniewski, G. Kindlmann, G. L. Kung, W. S. Klug, A. Garfinkel and D. B. Ennis, “Linear Invariant Tensor Interpolation Applied to Cardiac Diffusion Tensor MRI,” 2012.

[2] K. E. Kadler, C. Baldock, J. Bella and R. P. Boot-Handford, “Collagens at a glance,” *Journal of Cell Science,* vol. 120, pp. 1955-1958, 2005.

[3] L. Yang, “Mechanical properties of collagen fibrils and elastic fibers explored by AFM,” 2009.

[4] M. P. E. Wenger, L. Bozec, M. A. Horton and P. Mesquida, “Mechanical Properties of Collagen Fibrils,” *Biophysical Journa,* vol. 93, pp. 1255-1263, 2007.

[5] A. van Doorn, P. H. M. Bovendeerd, K. Nicolay, M. R. Drost and J. D. Janssen, “Determination of Muscle Fibre Orientation using Diffusion-Weighted MRI,” *European Journal of Morphology,* 1996.

[6] C. Flynn and B. A. McCormack, “Simulating the wrinkling and aging of skin with a multi-layer finite element model,” *Journal of Biomechanics,* 2010.

[7] X. Bi, G. Li, S. B. Doty and N. P. Camacho, “A novel method for determination of collagen orientation in cartilage by Fourier transform infrared imaging spectroscopy (FT-IRIS),” *OsteoArthritis and Cartilage,* vol. 13, pp. 1050-1058, 2005.

[8] A. Mirnajafi, J. Raymer, M. J. Scott and M. S. Sacks, “The effects of collagen fiber orientation on the flexural properties of pericardial heterograft biomaterials,” *Biomaterials,* vol. 26, no. 7, pp. 795-804, 1 3 2005.

[9] University of Massachusetts Medical School, “What is Electron Microscopy?,” 2017. [Online]. Available: https://www.umassmed.edu/cemf/whatisem/.

[10] John Innes Centre, “Electron Microscopy,” 2017. [Online]. Available: https://www.jic.ac.uk/microscopy/intro_EM.html.

[11] Gatan Inc, “Serial Block-Face Imaging,” 2018. [Online]. Available: http://www.gatan.com/techniques/serial-block-face-imaging.

[12] M. P. B. Petrou, Image Processing: The Fundamentals, Chichester: John Wiley & Sons Ltd, 1999, pp. 1-323.

[13] Pixabay, “Free Stock Photos of Rope,” 2018. [Online]. Available: https://www.pexels.com/photo/rope-dew-brown-beige-39279/.

[14] Mathematica StackExchange, *How to use 2D Fourier analysis to clean the noise in an image,* 2016.

[15] O. Wirjadi, “Survey of 3d image segmentation methods,” vol. 123, 2007.

[16] G. Koenderink, “How do collagen networks protect biological tissues from damage?,” 2016. [Online]. Available: https://amolf.nl/news/collagen-networks-protect-biological-tissues-damage.

[17] Wolfram, “Wolfram Language and System Documentation Center,” 2017. [Online]. Available: http://reference.wolfram.com/language/.

[18] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision,* 2004.

[19] BitPlane (Oxford Instruments), “3D Reconstruction and Segmentation of Whole Slide Images,” 2014. [Online]. Available: http://www.bitplane.com/learning/3d-reconstruction-and-segmentation-of-whole-slide-images-an-expert-imaris-tip.

[20] Allen Institute for Brain Science, “Vaa3D,” 2017. [Online]. Available: http://www.alleninstitute.org/what-we-do/brain-science/research/products-tools/vaa3d/.

[21] Stanford Medicine, “3D reconstruction software list,” 2018. [Online]. Available: http://biocomp.stanford.edu/3dreconstruction/software.html.

[22] R. Blanc, C. Germain, J. P. Da Costa, P. Baylou and M. Cataldi, “Fiber Orientation Measurements in Composite Materials,” *Composites Part A: Applied Science and Manufacturing, pp.197-20,* vol. 37, no. 2, pp. 197-206, 2006.

[23] S. Mönnich, R. Glöckner and F. Becker, “Analysis of Fibre Orientation using µCT Data,” Darmstadt, 2012.

[24] Z. Liu, K. P. Quinn, L. Speroni, L. Arendt, C. Kuperwasser, C. Sonnenschein, A. M. Soto, I. Georgakoudi, S. G. A. D. M. Lullo, J. Sweeney, L. Körkkö, J. D. Ala-Kokko, A. San, 7. P. P. Provenzano, K. W. Eliceiri, J. M. Campbell, D. R. Inman, J. G. White and P. J. Keely, “Rapid three-dimensional quantification of voxel- wise collagen fiber orientation,” *J. Biol. Chem. Science Acad. Sci. U.S.A. J. Miner. Met. Mater. Soc. Biomech,* vol. 277, no. 3910, pp. 4223-4231, 2002.

[25] K. Robb, O. Wirjadi and K. Schladitz, “Fiber Orientation Estimation from 3D Image Data: Practical Algorithms, Visualization, and Interpretation,” in *Seventh International Conference on Hybrid Intelligent Systems*, Kaiserlautern, 2007.

[26] T. Breuel, O. Wirjadi, K. Schladitz and A. Rack, “Applications of anisotropic image filters for computing 2D and 3D-fiber orientations,” 2009.

[27] J. P. Marquez, “Fourier analysis and automated measurement of cell and fiber angular orientation distributions,” *International Journal of Solids and Structures,* 2006.

[28] C. Bayan, J. M. Levitt, E. Miller, D. Kaplan and I. Georgakoudi, “Fully automated, quantitative, noninvasive assessment of collagen fiber content and organization in thick collagen gels,” *Journal of Applied Physics,* vol. 105, 2009.

[29] R. H. Bamberger and M. J. T. Smith, “A Filter Bank for the Directional Decomposition of Images: Theory and Design,” *IEEE TRANSACTIONS ON SIGNAL PROCESSING,* vol. 40, no. 4, 1992.

[30] T. Y. Lau, R. Ambekar and K. C. Toussaint, Jr, “Quantification of collagen fiber organization using three-dimensional Fourier transformsecond-harmonic generation imaging,” *OPTICS EXPRESS 21821,* vol. 20, no. 19, 2012.

[31] O. S. Osman, J. L. Selway, P. E. Harikumar, C. J. Stocker, E. T. Wargent, M. A. Cawthorne, S. Jassim and K. Langlands, “A novel method to assess collagen architecture in skin,” *BMC Bioinformatics,* 2013.

[32] I. Papusha and M. Ho, “Hough Transform for Directional Orientation,” 2010.

[33] A. Miettinen, “Characterization of three-dimensional microstructure of composite materials by X-ray tomography X-Ray computed micro-and nano-tomography of metamorphic fabric analysis View project,” 2016.

[34] H. Altendorf, E. Decencière, D. Jeulin, P. D. S. Peixoto, A. Deniset-Besseau, E. Angelini, G. Mosser and M. C. Schanne-Klein, “Imaging and 3D morphological analysis of collagen fibrils,” *Journal of Microscopy,* 2012.

[35] J. Blow and A. J. Binstock, “Why the inertia tensor is the inertia tensor,” 2004. [Online]. Available: http://number-none.com/blow/inertia/deriving_i.html.

[36] K. P. Quinn and I. Georgakoudi, “Rapid quantification of pixel-wise fiber orientation data in micrographs,” *Journal of Biomedical Optics,* 2013.

[37] B. K. Raymond W Wong, T. C. M Lee Debashis Paul Jie Peng and D. Neuroimaging Initiative, “Fiber direction estimation, smoothing and tracking in diffusion MRI,” *Annals of Applied Statistics,* 2015.

[38] S. Mori, B. J. Crain, V. P. Chacko and P. C. M. van Zijl, “Three-Dimensional Tracking of Axonal Projections in the Brain by Magnetic Resonance Imaging,” *The American Neurological Association,* vol. 45, no. 2, p. 265–269, 1999.

[39] K. Sandau and J. Ohser, “The chord length transform and the segmentation of crossing fibres,” *Journal of Microscopy,* 2007.

[40] M. Martone and E. Bushong, *Mouse BIRN test data,* Cell Image Library, 2001.

10 Appendix

10.1 Relevant code snippets

10.1.1 Moment of inertia 3D - Core algorithm

```
findSym3d[img_, forceSingle_: False, graphEnabled_: False] :=
  Module[{imd, timg, nx, ny, nz, points, weights, mass, com,
    centroid, centroids, count, v1, v2, v3, vtemp1, vtemp2, vtemp3,
    vec, lengths, widths, pointsCom, diag, inertia, masks, shapes,
    subImgs, gr, graaf, multiEnabled = True},

   imd = Transpose@
     ImageData[RemoveAlphaChannel@ColorConvert[img, "Grayscale"]];
   {nx, ny, nz} = Dimensions[imd];
   weights = Flatten[imd];
   mass = Total[weights];
   If[mass == 0 || mass == 1,
    Return[{0, Null, Null, Null, Null, Null }]];
   If[mass/(nx*ny*nz) < 0.01,
    Return[{0, Null, Null, Null, Null, Null }]];
   If[mass/(nx*ny*nz) > 0.99,
    Return[{0, Null, Null, Null, Null, Null }]];

   count = Values[ComponentMeasurements[img, "LabelCount"]][[1]];
   centroids = Values[ComponentMeasurements[img, "Centroid"]];

   If[(! forceSingle) && multiEnabled && count > 1,
    (
     lengths = Values[ComponentMeasurements[img, "Length"]];
     widths = Values[ComponentMeasurements[img, "Width"]];
     masks = Values[ComponentMeasurements[img, "Mask"]];
     shapes = Values[ComponentMeasurements[img, "Shape"]];
     subImgs = Values[ComponentMeasurements[img, "Image"]];
     gr = drawGraph3d[img, Null, Null, graphEnabled];
     Return[{count, Null, gr, Null, masks, subImgs}];
     )
    ];
   points = Tuples[{Reverse@Range[nx], Reverse@Range[ny], Range[nz]}];
   com = Total[points weights]/mass // N;

   pointsCom = (Subtract[#, com] & /@ points) weights;
   diag = Total[(#.#) & /@ pointsCom];
   inertia =
    Table[KroneckerDelta[i, j] diag -
      Total[(#[[i]] #[[j]]) & /@ pointsCom], {i, 1, 3}, {j, 1, 3}];
   {v1, v2, v3} = Eigenvectors[inertia];
   {vtemp1, vtemp2, vtemp3} = changeAxis /@ {v1, v2, v3};
   vec = vtemp3;
   centroid = centroids[[1]];
   graaf = drawGraph3d[img, centroid, vec, graphEnabled];
   Return[{count, vec, graaf, centroid, Null, Null}];
   ];
```

10.1.2 Moment of inertia 3D - ROI division

```
momentOfInertia3D[img_, pixFactor_: 10] :=
  Module[{vectorField, row1, row2, col1, col2, sli1, sli2, nRows,
    nCols, nSlis, subIm, count, vec, graff, cen, masks, subImages, cnt,
ve,
    gra, ce, mks, sI, timeStart, timeEnd}, (
    timeStart = TimeUsed[];
    vectorField = resetVectorField3D[img];
    {nSlis, nRows, nCols} = Dimensions@ImageData@img;
    Do[
     row1 = Ceiling[(nRows/pixFactor)*(i - 1) + 1];
     row2 = Ceiling[(nRows/pixFactor)*i];
     col1 = Ceiling[(nCols/pixFactor)*(j - 1) + 1];
     col2 = Ceiling[(nCols/pixFactor)*j];
     sli1 = Ceiling[(nSlis/pixFactor)*(h - 1) + 1];
     sli2 = Ceiling[(nSlis/pixFactor)*h];
     subIm = ImageTake[
       img, {sli1, sli2}, { row1, row2}, { col1, col2 }
       ];
     {count, vec, graff, cen, masks, subImages} = findSym3d[subIm];
     If[count == 0, (
       Do[
         If[vectorField[[hi, k, l]] === {1, 1, 1},
           vectorField[[hi, k, l]] = {0, 0, 0}];
         , {k, row1, row2}, {l, col1, col2}, {hi, sli1, sli2}
         ];
       )];

     If[count == 1,
      If[! SameQ[vec, Null],
        (
         If[vec[[3]] < 0, vec = -vec];
         Do[
          If[vectorField[[hi, k, l]] === {1, 1, 1},
            vectorField[[hi, k, l]] = vec];
          , {k, row1, row2}, {l, col1, col2}, {hi, sli1, sli2}
          ];
         )
        ];
      ];

     If[count > 1, (
       Do[(
          {cnt, ve, gra, ce, mks, sI} =
           findSym3d[subImages[[m]], True];
          If[! SameQ[ve, Null],
           (
            If[ve[[3]] < 0, ve = -ve];
            Do[

             If[masks[[m, hi - sli1 + 1, k - row1 + 1,
                 l - col1 + 1]] === 1,
               vectorField[[hi, k, l]] = ve;
               ];
```

```
          , {k, row1, row2}, {l, col1, col2}, {hi, sli1, sli2}
          ];
         ), (
         Do[

           If[masks[[m, hi - sli1 + 1, k - row1 + 1,
               l - col1 + 1]] === 1,
             vectorField[[hi, k, l]] = {0, 0, 0};
             ];
           , {k, row1, row2}, {l, col1, col2}, {hi, sli1, sli2}
           ];
         )
        ];
       ), {m, 1, count}];
    )];

 Do[
  If[vectorField[[hi, k, l]] === {1, 1, 1},
    Print["Problematic point: sli = ", hi, ", row = ", k,
      ", col = ", l];
    ];
  , {k, row1, row2}, {l, col1, col2}, {hi, sli1, sli2}
  ];

 , {i, 1, pixFactor}, {j, 1, pixFactor}, {h, 1, pixFactor}
 ];
timeEnd = TimeUsed[];
Print["Moment of Inertia one pass with iteration factor = ",
 pixFactor,
 " and image size {slices, rows, columns}=", {nSlis, nRows,
  nCols}, " took ", timeEnd - timeStart, " (s)"];
Return[vectorField];
)];
```

10.1.3 Weighted Vector Summation 2D – Algorithm

```
vectorSummation2DPart[imgdata_, window_] :=
  Module[{img,  nRows, nCols, vectors, tv, center, lengths, l,
    weighted1Vectors, images, vec, img1, img2, vari, meanIntensity,
    w2, pixelValues, pixValues, runningTotalX, runningTotalY,
    weightedVector, axialWV, vectorField, angleField, tempv2, tempv}, (
    img = Image@imgdata;
    {nRows, nCols} = ImageDimensions[img];
    vectors = {};
    weighted1Vectors = {};
    lengths = {};
    center = Ceiling[window/2];
    Do[(
      If[! (i >= center && j === center), (
         tv = {(i - center), (center - j)};
         AppendTo[vectors, tv];
         l = Sqrt[tv[[1]]^2 + tv[[2]]^2];
         AppendTo[lengths, l];
         AppendTo[weighted1Vectors, tv/(l^2)];
         )];
      ), {i, 1, window}, {j, 1, center}];

    runningTotalX = ConstantArray[0, {nRows, nCols}];
    runningTotalY = ConstantArray[0, {nRows, nCols}];

    Do[(
      vec = vectors[[i]];
      l = lengths[[i]];
      img1 =
       ImageTransformation[
        img, # + {vec[[1]]/nCols, vec[[2]]/nRows} &];
      img2 =
       ImageTransformation[
        img, # + {-vec[[1]]/nCols, -vec[[2]]/nRows} &];
      images = {img1, img, img2};

      Do[(
        If[imgdata[[r, c]] > 0.5, (
           pixValues = {};
           pixValues = (ImageData[#])[[r, c]] & /@ images;
           If[True || Variance[pixValues] != 0, (
             meanIntensity = Mean[pixValues];
             vari = Variance[pixValues];
             w2 = Sqrt[1/3] - Sqrt[vari];
             weightedVector = (vec/(l))*w2;
             axialWV = vectorAngleMultiply[weightedVector, 2];
             runningTotalX[[r, c]] += axialWV[[1]];
             runningTotalY[[r, c]] += axialWV[[2]];
             )];
           )];
        ), {r, center, nRows - center}, {c, center, nCols - center}];
      ), {i, 1, Length[vectors]}];

    vectorField = ConstantArray[{0, 0}, {nRows, nCols}];
```

```
Do[(
  If[imgdata[[r, c]] > 0.5,
    tempv = {runningTotalX[[r, c]], runningTotalY[[r, c]]};
    tempv2 = vectorAngleMultiply[tempv, 0.5];
    vectorField[[r, c]] = tempv2;
    ];
  ), {r, center, nRows - center}, {c, center, nCols - center}];
Return[{Null, vectorField}];
)];
```

10.1.4 Weighted Vector Summation 3D – Algorithm

```
vectorSummation3D[image_, window_: 7] :=
  Module[{img, imgdata, center, nSlis, nRows, nCols, xyVectors,
    yzVectors, zxVectors, averagedSlice, xyAveragedSlices,
    yzAveragedSlices, zxAveragedSlices, slices, xyVectorField,
    yzVectorField, zxVectorField, thetaField, betaField, gammaField,
    phiField, fieldLayer, angleFieldLayer, totalVectorPlot3D,
    totalVectorField, t, tvx, tvy, tvz, theta, phi, testX1 := 5,
    testX2 := 6, timeStart, timeEnd}, (
    timeStart = TimeUsed[];
    img = ColorConvert[image, "GrayScale"];
    imgdata = ImageData@img;
    {nSlis, nRows, nCols} = Dimensions[imgdata];
    center = Ceiling[window/2];
    xyVectors = ConstantArray[{0, 0}, {nSlis, nRows, nCols}];
    yzVectors = ConstantArray[{0, 0}, {nSlis, nRows, nCols}];
    zxVectors = ConstantArray[{0, 0}, {nSlis, nRows, nCols}];
    xyAveragedSlices = {};
    yzAveragedSlices = {};
    zxAveragedSlices = {};
    Do[(
      slices = imgdata[[1 ;; l]];
      averagedSlice = averageSlices[slices, 1];
      AppendTo[xyAveragedSlices, averagedSlice];
      ), {l, 1, center - 1}];
    Do[(
      slices = imgdata[[All, 1 ;; r, All]];
      averagedSlice = averageSlices[slices, 2];
      AppendTo[zxAveragedSlices, averagedSlice];
      ), {r, 1, center - 1}];
    Do[(
      slices = imgdata[[All, All, 1 ;; c]];
      averagedSlice = averageSlices[slices, 3];
      AppendTo[yzAveragedSlices, averagedSlice];
      ), {c, 1, center - 1}];

    Do[(
      slices = imgdata[[l - center + 1 ;; l + center - 1]];
      averagedSlice = averageSlices[slices, 1];
      AppendTo[xyAveragedSlices, averagedSlice];
      ), {l, center, nSlis - center}];
    Do[(
      slices = imgdata[[All, r - center + 1 ;; r + center - 1, All]];
      averagedSlice = averageSlices[slices, 2];
      AppendTo[zxAveragedSlices, averagedSlice];
      ), {r, center, nRows - center}];
    Do[(
      slices = imgdata[[All, All, c - center + 1 ;; c + center - 1]];
      averagedSlice = averageSlices[slices, 3];
      AppendTo[yzAveragedSlices, averagedSlice];
      ), {c, center, nCols - center}];

    Do[(
      slices = imgdata[[l ;; nSlis]];
```

```
  averagedSlice = averageSlices[slices, 1];
  AppendTo[xyAveragedSlices, averagedSlice];
  ), {l, nSlis - center + 1, nSlis}];
Do[(
  slices = imgdata[[All, r ;; nRows, All]];
  averagedSlice = averageSlices[slices, 2];
  AppendTo[zxAveragedSlices, averagedSlice];
  ), {r, nRows - center + 1, nRows}];
Do[(
  slices = imgdata[[All, All, c ;; nCols]];
  averagedSlice = averageSlices[slices, 3];
  AppendTo[yzAveragedSlices, averagedSlice];
  ), {c, nCols - center + 1, nCols}];

xyVectorField = ConstantArray[{0, 0}, {nSlis, nRows, nCols}];
yzVectorField = ConstantArray[{0, 0}, {nSlis, nRows, nCols}];
zxVectorField = ConstantArray[{0, 0}, {nSlis, nRows, nCols}];

Do[(
  {angleFieldLayer, fieldLayer} =
   vectorSummation2DPart[xyAveragedSlices[[i]], window];
  xyVectorField[[i, All, All]] = fieldLayer;
  ), {i, 1, Length@xyAveragedSlices}];

Do[(
  {angleFieldLayer, fieldLayer} =
   vectorSummation2DPart[yzAveragedSlices[[i]], window];
  yzVectorField[[All, All, i]] = fieldLayer;
  ), {i, 1, Length@yzAveragedSlices}];

Do[(
  {angleFieldLayer, fieldLayer} =
   vectorSummation2DPart[zxAveragedSlices[[i]], window];
  zxVectorField[[All, i, All]] = fieldLayer;
  ), {i, 1, Length@zxAveragedSlices}];


totalVectorField =
 addThirdComponent[xyVectorField, 3] +
  addThirdComponent[yzVectorField, 1] +
  addThirdComponent[zxVectorField, 2];
t = N@FindThreshold[img];
Do[
 If[imgdata[[s, r, c]] < t, (
    totalVectorField[[s, r, c]] = {0, 0, 0};
    ), (
    If[totalVectorField[[s, r, c]][[3]] < 0,
      totalVectorField[[s, r, c]] = -totalVectorField[[s, r, c]];];
    (*totalVectorField[[s,r,c]]=Normalize[totalVectorField[[s, r,
    c]]];
    phiField[[s,r,c]]=calculatePhi[betaField[[s,r,c]],gammaField[[
    s,r,c]]];
    theta =thetaField[[s,r,c]];
    phi =phiField[[s,r,c]];
    tvx=Cos [theta] Sin[phi];
    tvy =Sin[theta] Sin[phi];
```

```
    tvz = Cos[phi];
    totalVectorField[[s,r,c]]={tvx, tvy, tvz};*)
    )];
 , {r, 1, nRows}, {c, 1, nCols}, {s, 1, nSlis}
 ];
timeEnd = TimeUsed[];
Print["Vector Summation one pass with window=", window,
 " and image size {slices, rows, columns}=", {nSlis, nRows,
  nCols}, " took ", timeEnd - timeStart, " (s)"];
Return [totalVectorField];
)];
```

10.1.5 Fourier Analysis 2D – Algorithm

```
angularAmplitude[ Po_, theta1_, window_: 0.05] := Module[{pts, sum}, (
    pts = Select[Po, (#[[2]] > theta1 && #[[2]] < theta1 + window) &];
    sum = 0;
    Do[(
      sum += pts[[i, 3]];
      ), {i, 1, Length@pts}];
    If[Length@pts === 0, Return[0], Return[sum/Length@pts]];
    )];

fourierDo[io_, t_: 2] :=
  Module[{data, nR, nC, nCh, d, fw, fudgeFactor, abs, arg, laIm1,
    laIm2, ur, theta, xh, tempXh, sortedXh, tempPo, sortedPo, Po,
    Atheta, u, v, plot, fc, f1, f2, filteredPo, filteredPlot,
    angularAmplitudes, tempAngularAmplitudes,
    filteredAngularAmplitudes, angularAmplitudePlot, fw2, ifw, idata,
    iImage, tu, tv, oi, thetaPlot, thetaVsFreqPlot, filterPlot,
    fitCurve, fitPlot, dataPlot}, (
    data = ImageData[io];
    {nR, nC} = Dimensions[data];
      d = data[[All, All]];
      (*d=d*(-1)^Table[i+j,{i,nR},{j,nC}];*)
      fw = Fourier[d, FourierParameters -> {1, 1}];
      fudgeFactor = 100;
      abs = fudgeFactor*Log[1 + Abs@fw];
      laIm1 =
     Labeled[Image[abs/Max[abs], ImageSize -> 300],
      Style["Magnitude spectrum", 18]];
    arg = Arg@fw;
      laIm2 =
     Labeled[Image[arg/Max[arg], ImageSize -> 300],
      Style["Phase spectrum", 18]];
    xh = {};
    Po = {};
    filteredPo = {};
    Atheta = {};
    fc = Max[nR, nC]/(2*t);
    f1 = 0.9*fc;
    f2 = 1.1*fc;
    filterPlot =
     Labeled[Plot[
       Piecewise[{{1, f1 < x && f2 > x}}, 0], {x, 0,
        Sqrt[nR^2 + nC^2]}, Axes -> True, ImageSize -> 450,
       FrameLabel -> {"Frequency", "Amplitude"}, Frame -> True],
      Style["Bandpass Filter", 18]];
    Do[(
      ur = N@Sqrt[u^2 + v^2];
      theta = N@ArcTan[u, v];
      AppendTo[xh, {ur, theta, fw[[u, v]]}];
      AppendTo[Po, {ur, theta, Abs[fw[[u, v]]]^2 }];
      AppendTo[Po, {ur, -N@Pi/2 + theta, Abs[fw[[u, v]]]^2 }];
      If[ur > f1 && ur < f2, (
        AppendTo[filteredPo, {ur, theta, Abs[fw[[u, v]]]^2 }];
        AppendTo[
```

```
     filteredPo, {ur, -N@Pi/2 + theta, Abs[fw[[u, v]]]^2 }];
    )];
  ), {u, 1, nR}, {v, 1, nC}];
plot =
 Labeled[ListPlot[({#[[1]], #[[3]]}) & /@ Po, Axes -> True,
   ImageSize -> 450, FrameLabel -> {"Frequency", "Power"},
   Frame -> True], Style["Radial Frequency vs. Power", 18]];
thetaPlot =
 Labeled[ListPlot[({#[[2]] / Degree, #[[3]]}) & /@ Po,
   Axes -> True, ImageSize -> 450,
   FrameLabel -> {"Phase Angle (deg)", "Power"}, Frame -> True],
  Style["Phase Angle vs. Power", 18]];
thetaVsFreqPlot =
 Labeled[ListPlot[({#[[2]]/Degree, #[[1]]}) & /@ Po,
   Axes -> True, ImageSize -> 450,
   FrameLabel -> {"Phase angle (deg)", "Frequency"},
   Frame -> True], Style["Phase Angle vs. Radial Frequency", 18]];
filteredPlot =
 Labeled[ListPlot[({#[[1]], #[[3]]}) & /@ filteredPo,
   Axes -> True, ImageSize -> 450,
   FrameLabel -> {"Frequency", "Power"}, Frame -> True],
  Style["Bandpass Filtered - Radial Frequency vs Power", 18]];
angularAmplitudes = {};
Do[
 AppendTo[
   angularAmplitudes, {90 - tt*(1/0.25),
    angularAmplitude[filteredPo,
     tt *(Degree/0.25), (1/0.25) Degree]}];
 , {tt, Round[-90*0.25], Round[90*0.25]}];
tempAngularAmplitudes =
 Sort[angularAmplitudes, #1[[1]] < #2[[1]] &];
angularAmplitudes = tempAngularAmplitudes;
filteredAngularAmplitudes =
 MeanFilter[angularAmplitudes[[All, {2}]], 1];
fitCurve =
 Fit[angularAmplitudes, {1, x, x^2, x^3, x^4, x^5, x^6, x^7, x^8},
   x];

fitPlot = Plot[fitCurve, {x, 0, 180}];
dataPlot = ListPlot[angularAmplitudes];
angularAmplitudePlot =
 Labeled[Show[fitPlot, dataPlot,
   PlotRange -> {{0, 180}, {0, Automatic}} ,
   Axes -> {False, True}, ImageSize -> 450,
   FrameLabel -> {"Phase angle (deg)", "Amplitude"},
   Frame -> True], Style["Angular Amplitude Plot", 18]];
fw2 = ConstantArray[0, Dimensions@fw];
Do[
 tu = Round[filteredPo[[i, 1]]*Cos[filteredPo[[i, 2]]]];
 tv = Round[filteredPo[[i, 1]]*Sin[filteredPo[[i, 2]]]];
 fw2[[tu, tv]] = fw[[tu, tv]];
 , {i, 1, Length@filteredPo}];

ifw = InverseFourier[fw2, FourierParameters -> {1, 1}];
(*If[Max@Abs@Im@ifw/Max@Abs@Re@ifw>10^-10,Print@
"Warning, Inverse FFT returned complex results,\n proceeding with \
```

```
real part only"];*)
    ifw = Re@ifw;
    idata = Map[{#, #, #} &, ifw, {2}];
    iImage =
     Labeled[Image[idata, ImageSize -> 300] // ImageAdjust,
      Style["Reconstructed image", 18]];
    oi = Labeled[Image[io, ImageSize -> 300],
      Style["Original image", 18]];
    Return[{oi, laIm1, laIm2, thetaPlot, plot, filterPlot,
      filteredPlot, angularAmplitudePlot, iImage, angularAmplitudes}];
    )];
```

10.1.6 Gradient orientation filter 2D – Algorithm

```
findOrientationGradFilter[img_] :=
  Module[{grad, vectorField, nRows, nCols, vx, vy, theta, t, spa}, (
    grad =
     GradientOrientationFilter[img, 1,
      Method -> "DerivativeKernel" -> "Sobel"];
    {nRows, nCols} = grad // ImageDimensions;
    vectorField = ConstantArray[{0, 0}, {nRows, nCols}];
    t = FindThreshold[img];
    Do[(
      theta = (ImageData@grad)[[r, c]];
      If[theta <= N@(-Pi/2) || (ImageData@img)[[r, c]] > t, (
        vx = 0;
        vy = 0;
        ), (
        vx = Cos[theta];
        vy = Sin[theta];
        )];
      If[vy < 0, vx = -vx; vy = -vy;];
      vectorField[[r, c]] = {vx, vy};
      ), {r, 1, nRows}, {c, 1, nCols}];
    spa = SparseArray[vectorField];
    Return[vectorField];
    )];
```

10.1.7 Filtering and binarization – Core

```
getBinarizedImage3D[im_, method_: 1, radius_: 3, threshold_: 0.08] :=
  Module[{meanFiltered, gradientFiltered, isrgb, img, binarized,
    sobelX1, sobelX2, sobelX3, sobelY1, sobelY2, sobelY3, sobelZ1,
    sobelZ2, sobelZ3, sobelX, sobelY, sobelZ, sobelIm}, (
    binarized = Null;
    img = im;
    If[img // ImageData // Dimensions //
      Length > 3, (isrgb = True), (isrgb = False)];
    If[isrgb, (
      img = firstPrincipalComponentFromRGB3D[im];
      ), (
      img = RemoveAlphaChannel@ColorConvert[im, "Grayscale"];
      )];
    meanFiltered = MeanFilter[img, radius];
    If[method == 1, (
      binarized = Binarize[img];
      )];
    If[method == 2, (
      binarized =
        MorphologicalBinarize[img,
         FindThreshold[img, Method -> "Cluster"]];
      )];
    If[method == 3, (
      gradientFiltered = GradientFilter[img, radius] // ImageAdjust;
      binarized = getBinarizedImage3D[gradientFiltered, 1];
      )];
    If[method == 4, (
      sobelX1 = {{-1, -3, -1}, {-3, -6, -3}, {-1, -3, -1}};
      sobelX2 = 0*sobelX1;
      sobelX3 = -1*sobelX1;
      sobelY1 = {{1, 3, 1}, {0, 0, 0}, {-1, -3, -1}};
      sobelY2 = {{3, 6, 3}, {0, 0, 0}, {-3, -6, -3}};
      sobelY3 = sobelY1;
      sobelZ1 = {{-1, 0, 1}, {-3, 0, 3}, {-1, 0, 1}};
      sobelZ2 = {{-3, 0, 3}, {-6, 0, 6}, {-3, 0, 3}};
      sobelZ3 = sobelZ1;
      sobelX = {sobelX1, sobelX2, sobelX3};
      sobelY = {sobelY1, sobelY2, sobelY3};
      sobelZ = {sobelZ1, sobelZ2, sobelZ3};
      sobelIm =
       Image[Sqrt[
          ImageData[ImageConvolve[meanFiltered, sobelX]]^2 +
           ImageData[ImageConvolve[meanFiltered, sobelY]]^2 +
           ImageData[ImageConvolve[meanFiltered, sobelZ]]^2]] //
        ImageAdjust;
      binarized = MorphologicalBinarize[sobelIm];
      )];
    If[method == 5, (
      binarized =
        EdgeDetect[GaussianFilter[img, radius], Method -> "Sobel"];
      )];
    If[method == 6, (
      binarized =
```

```
    MorphologicalBinarize[
     meanFiltered, {threshold, FindThreshold[meanFiltered]}];
  )];
If[method == 7, (
  binarized =
    MorphologicalBinarize[meanFiltered,
     FindThreshold[meanFiltered]];
  )];
Return[binarized];
)];
```

10.2 2D Algorithms further analysis graphs

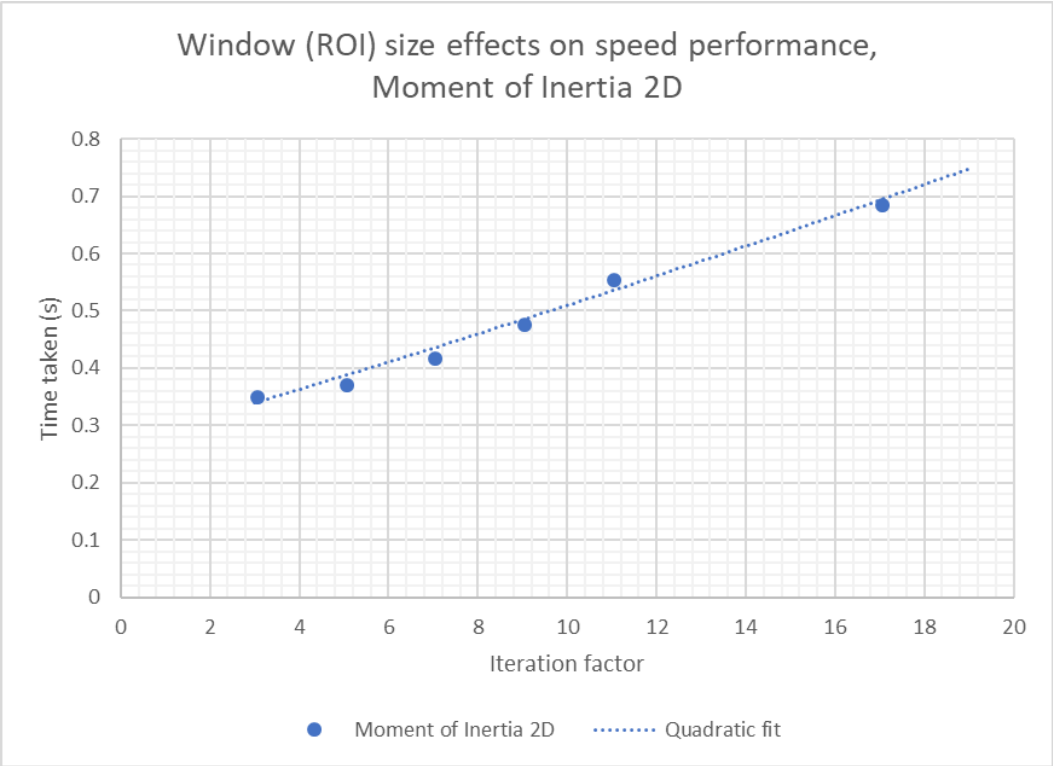


Fig 10.1: Speed performance vs. iteration factor (inverse of window size) for 2D Moment of Inertia algorithm, (Lower time value represents better speed performance)

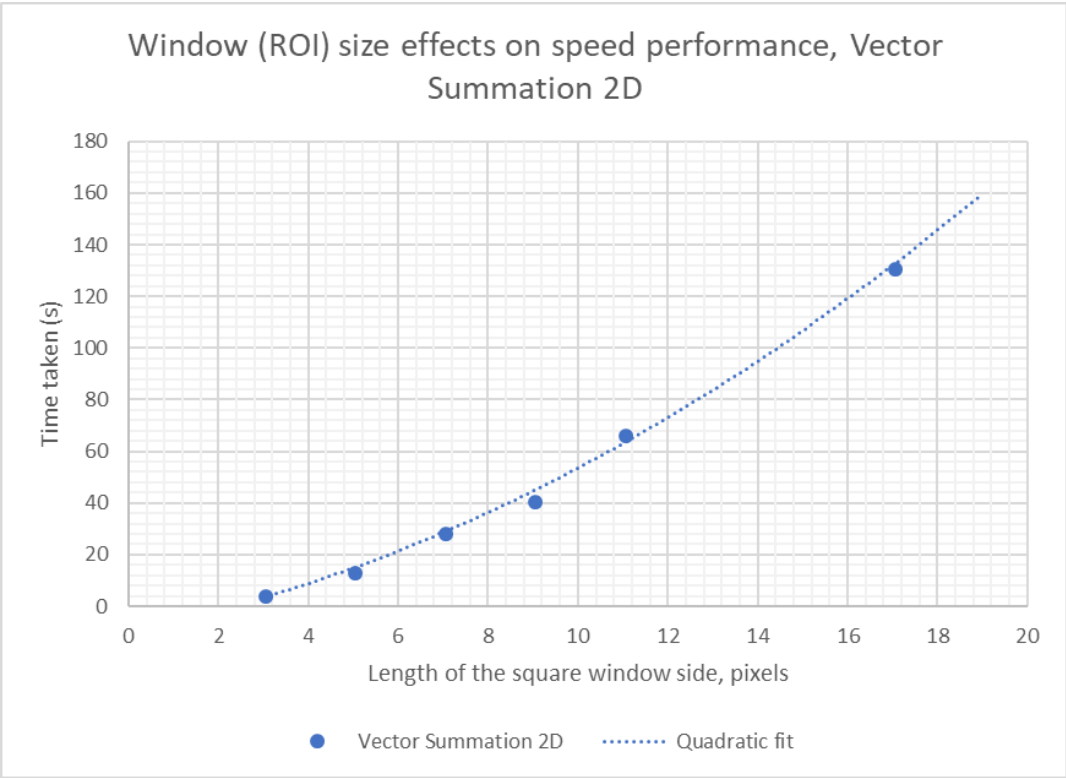


Fig 10.2: Speed performance vs. window size for 2D Weighted Vector Summation algorithm, (Lower time value represents better speed performance)

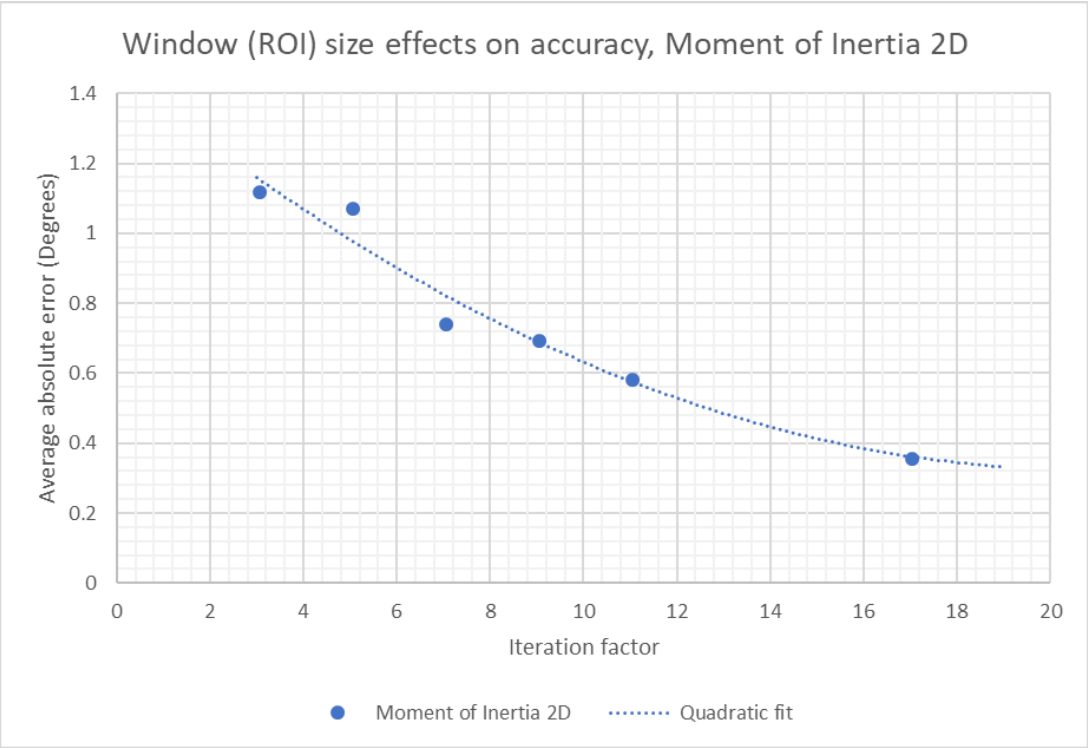


Fig 10.3: Accuracy vs. iteration factor (inverse of window size) for 2D Moment of Inertia algorithm, (Lower average error represents better accuracy)

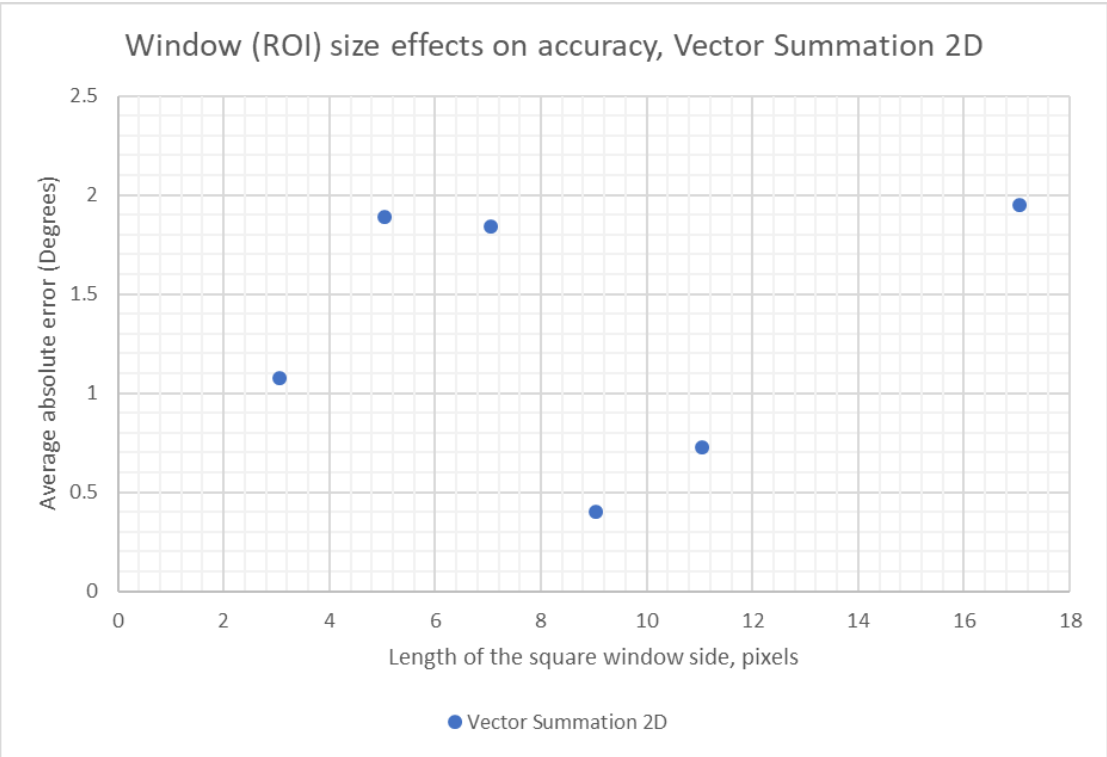


Fig 10.4: Accuracy vs. window size for 2D Weighted Vector Summation algorithm, (Lower average error represents better accuracy)

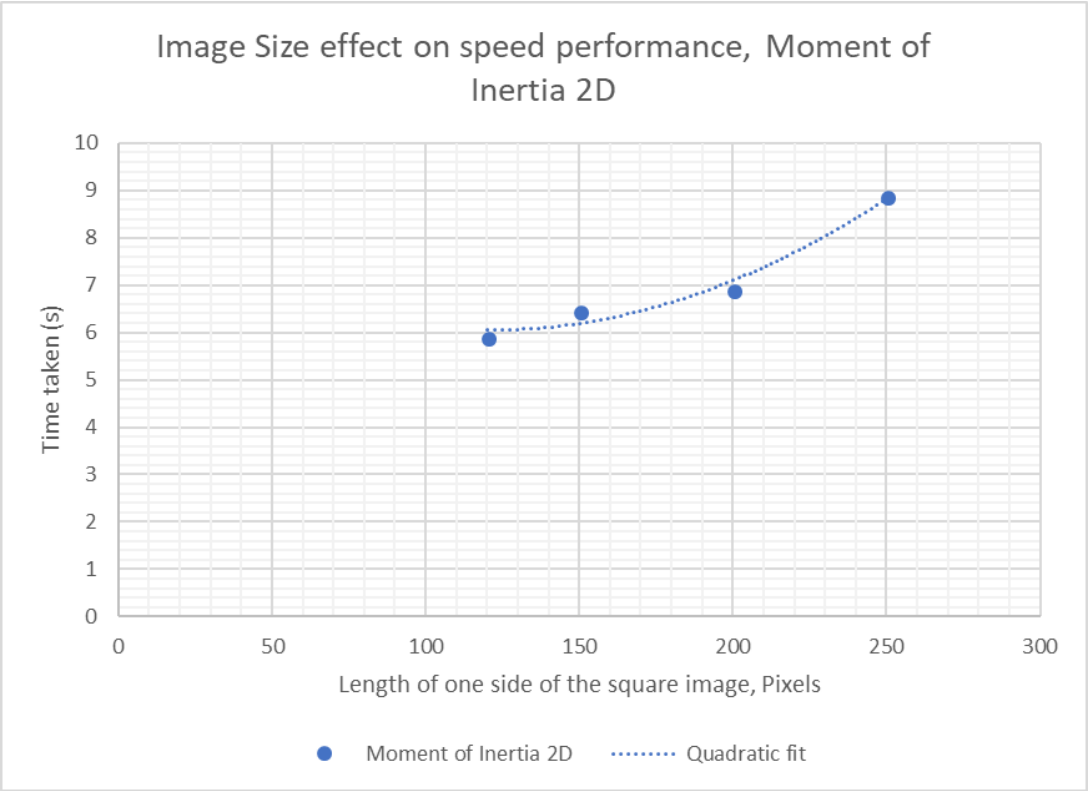


Fig 10.5: Speed performance vs. image size for 2D Moment of Inertia algorithm, (Lower time value represents better speed performance)

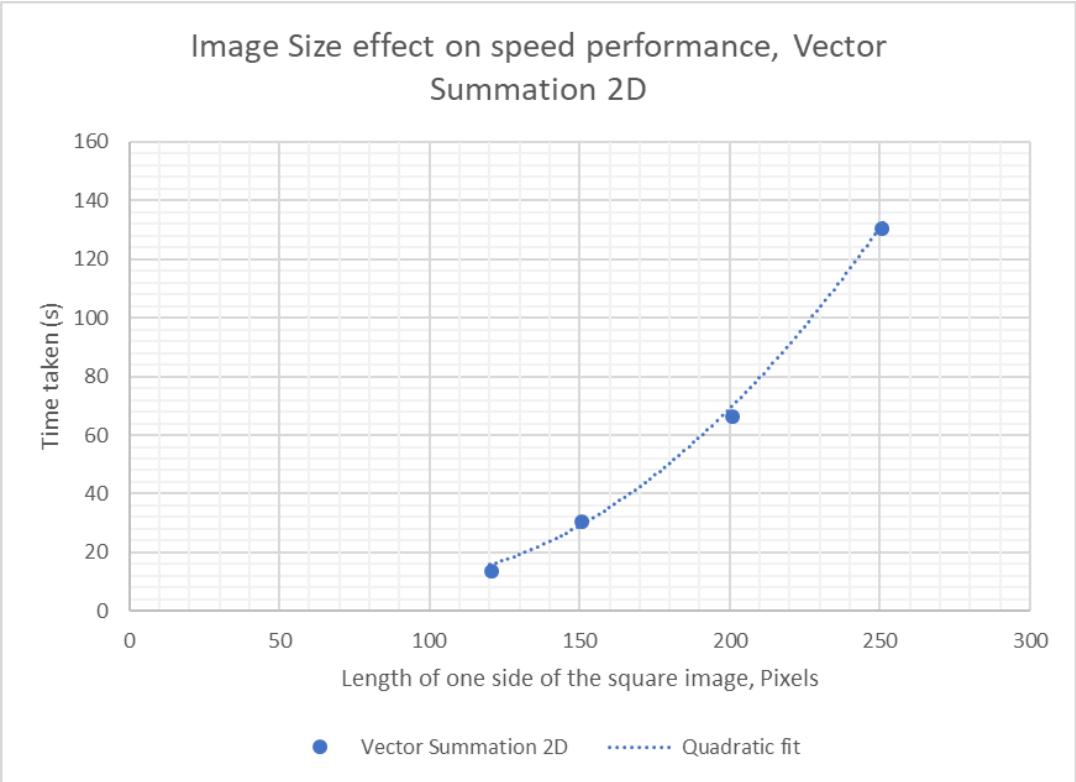


Fig 10.6: Speed performance vs. image size for 2D Weighted Vector Summation algorithm, (Lower time value represents better speed performance)

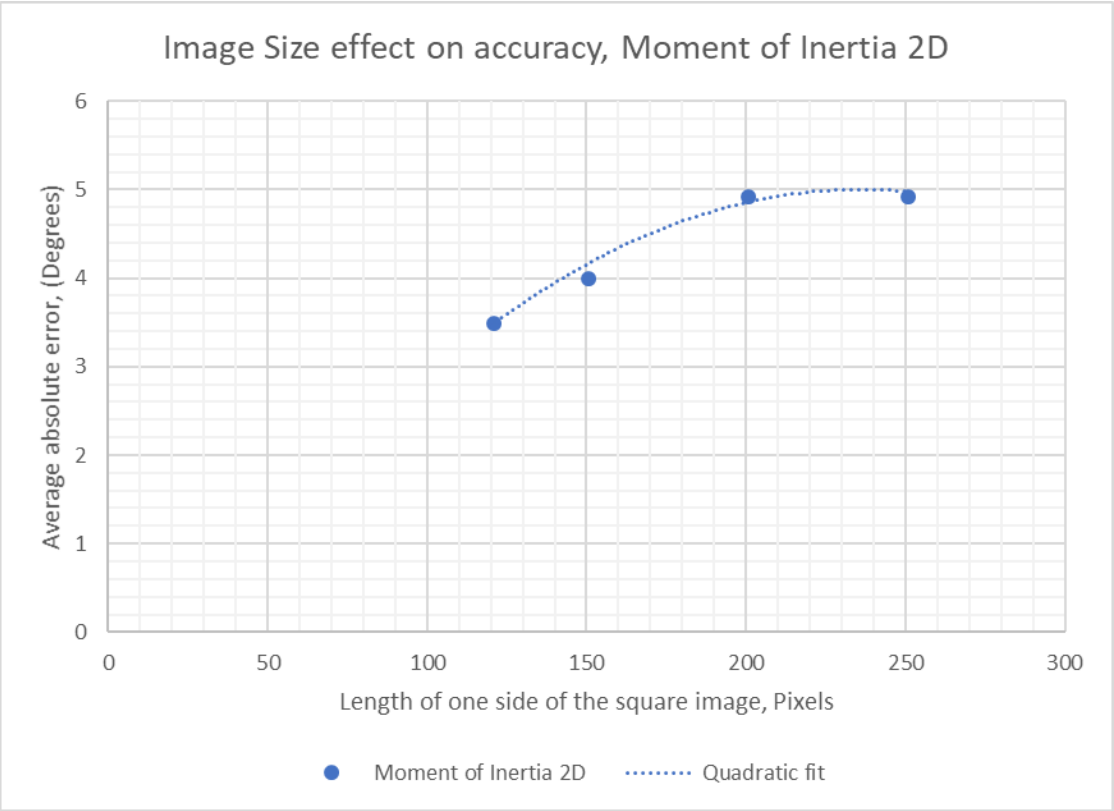


Fig 10.7: Accuracy vs. image size for 2D Moment of Inertia algorithm, (Lower average error represents better accuracy)

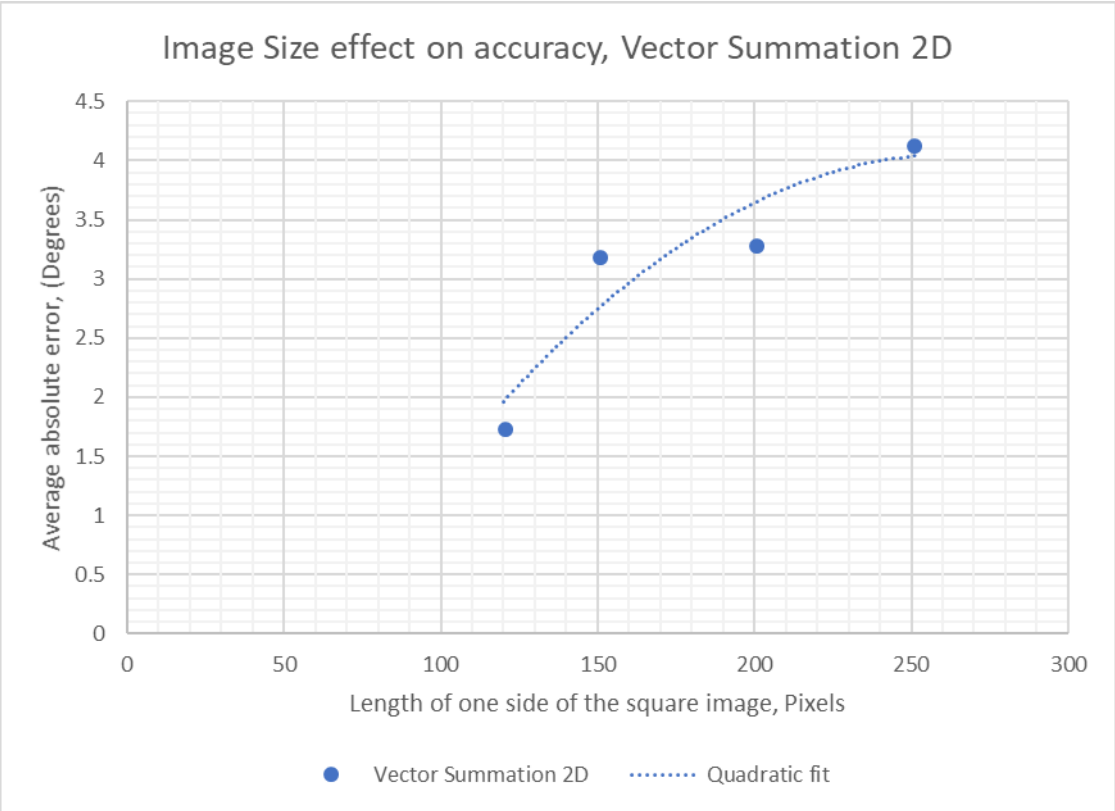


Fig 10.8: Accuracy vs. image size for 2D Weighted Vector Summation algorithm, (Lower average error represents better accuracy)

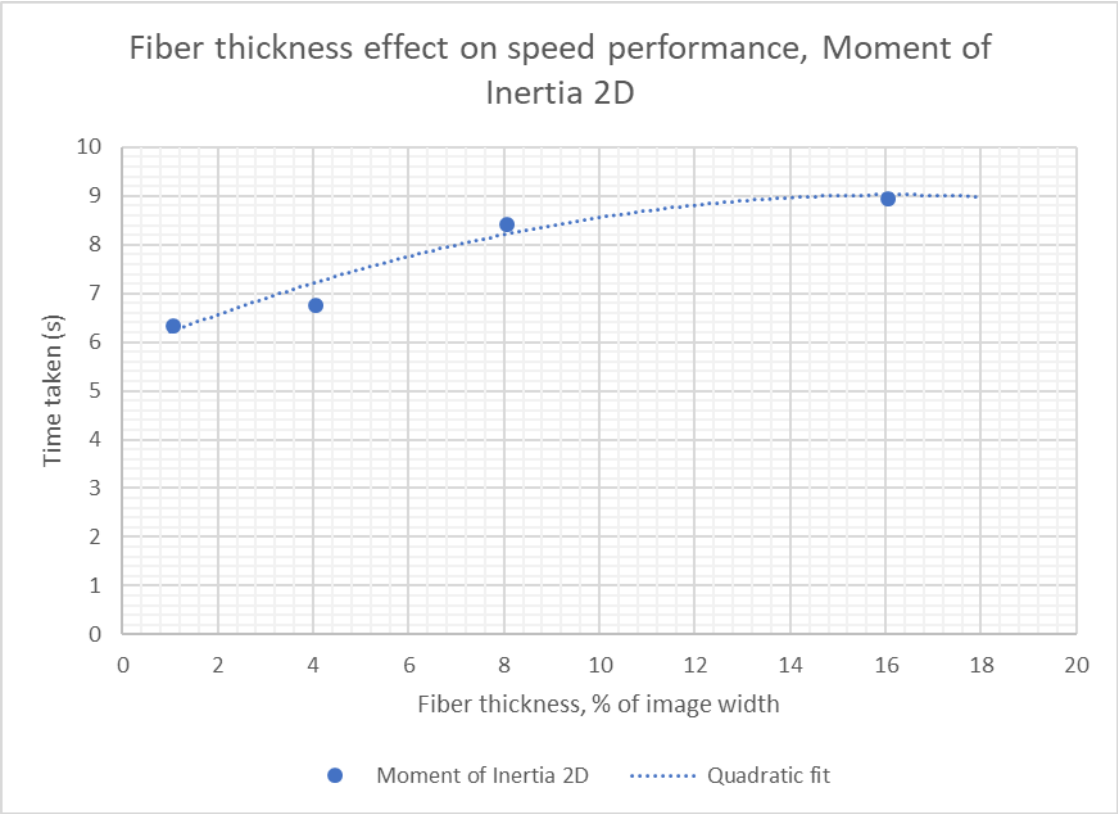


Fig 10.9: Speed performance vs. fibre thickness for 2D Moment of Inertia algorithm, (Lower time value represents better speed performance)

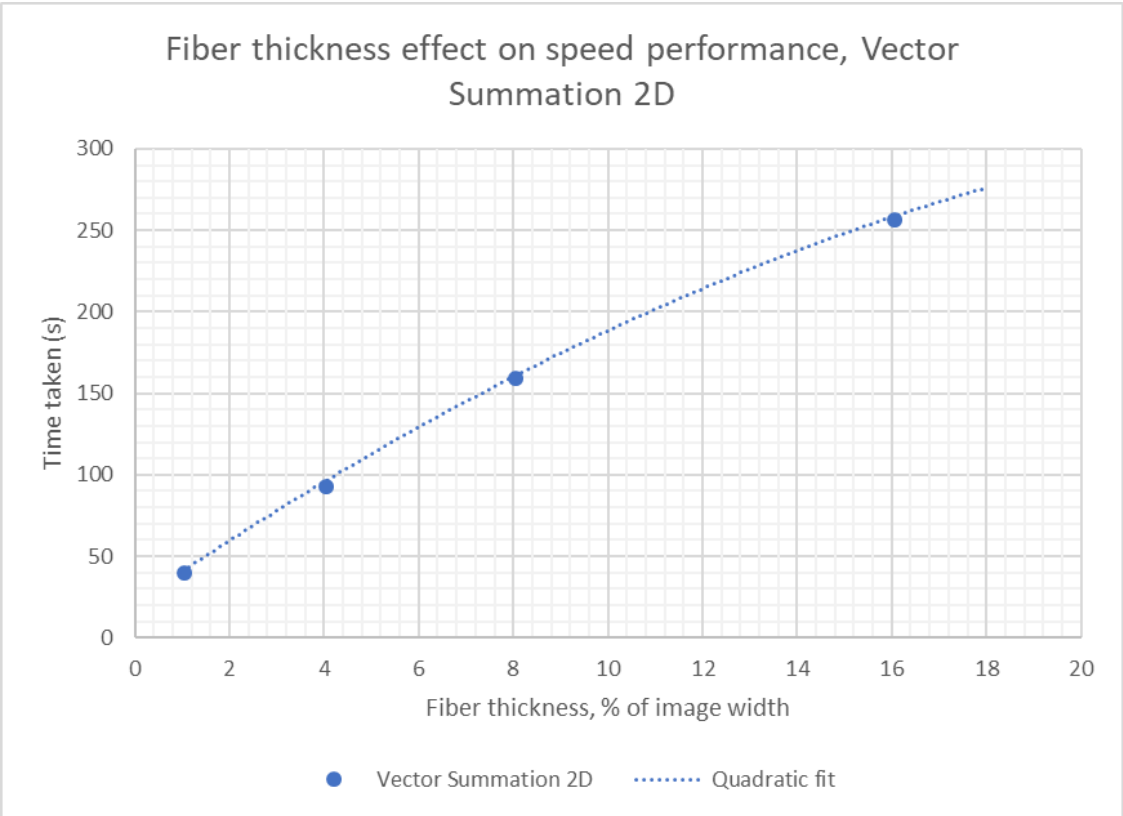


Fig 10.10: Speed performance vs. fibre thickness for 2D Weighted Vector Summation algorithm, (Lower time value represents better speed performance)

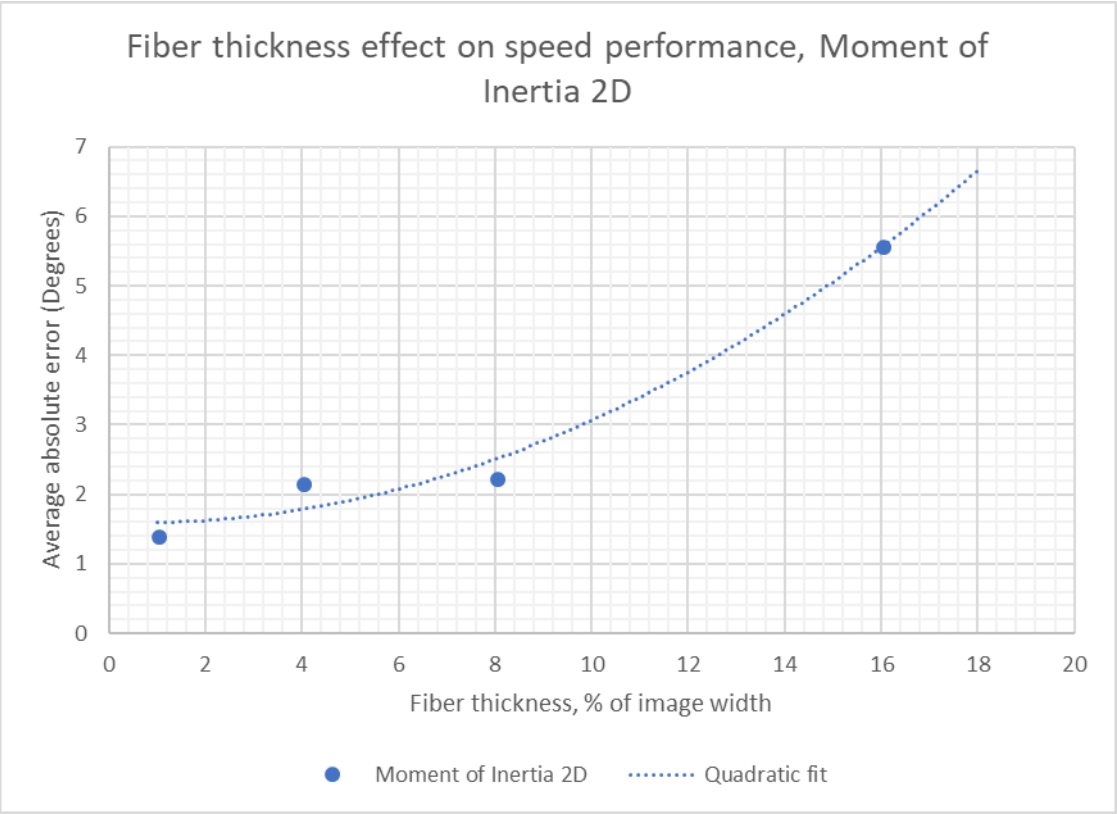


Fig 10.11: Accuracy vs. Fibre thickness for 2D Moment of Inertia algorithm, (Lower average error represents better accuracy)

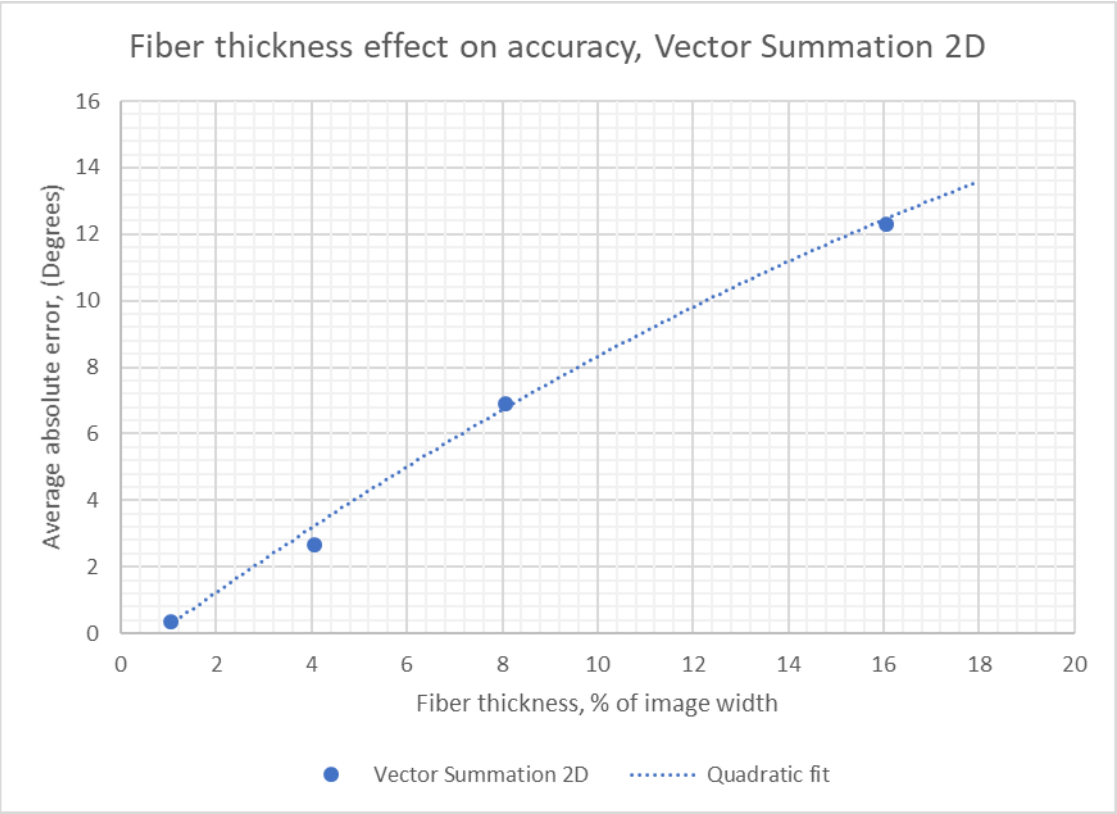


Fig 10.12: Accuracy vs. Fibre thickness for 2D Weighted Vector Summation algorithm, (Lower average error represents better accuracy)