

Maria Danilova

A novel tool for directionality assessment in medical images

Computing and Electrical Engineering
Master's thesis
April 2019

Abstract

Maria Danilova: A novel tool for directionality assessment in medical images
Master's thesis
Tampere University
Master's Degree Programme in Electrical Engineering
April 2019

Modern medicine relies on visual data. Until recently, manual inspection was the only method available, although it introduced a large amount of subjectiveness into the final diagnosis. Computer-aided medical data processing aims to increase diagnosis accuracy. This thesis covers one method to use computers for determining image directionality, which can be applied in nerve and vessel disease detection. In particular, a number of directionality evaluation methods for 2-dimensional (2D) images are examined. For one of the methods, a 3-dimensional (3D) extension is proposed. This method is based on the Fourier Transform. The relationship between the spacial domain lines and their frequency domain counterparts and the way it can be used to define the directionality is covered in the 'Materials and methods' part of this thesis.

The proposed extension was implemented using MATLAB and supplied with a graphical user interface (GUI). It was tested against both test images, such as lines and volumetric spheres, and real optic nerve MicroCT images. Results of these tests were analyzed and suggestions were given on the potential usage of the method.

Keywords: directionality, imaging, MATLAB.

The originality of this thesis has been checked using the Turnitin Originality Check service.

Preface

This thesis was made at Tampere University of Technology (TUT) as a part of Biomedical Engineering major studies. The tool developed in scope of this thesis can be applied to tissue and cellular images assessment for directional pattern detection purpose.

I would like to thank my supervisors, Professor Jari Hyttinen, Michelangelo Paci, and Florentino Caetano Dos Santos, for their support, patience, and expertise.

Tampere, 01.12.2018

Maria Danilova, danilova.maria@rambler.ru

Contents

1	Introduction	2
1.1	Optic nerve imaging	2
1.2	Cerebrovascular system tomography	3
1.3	Directionality in medical imaging	4
2	Background	6
3	Materials and methods	12
3.1	Fourier transform of a line in 2D and 3D	12
3.2	Reference method	16
3.3	Image enhancement techniques	17
4	Implementation	21
4.1	Interface	21
4.2	Ray tracing algorithm	23
4.3	Plane tracing	25
4.4	Performance considerations	25
4.5	Bug hunting	27
5	Results and discussion	28
5.1	2D images	28
5.1.1	Sample images	28
5.1.2	Real images	30
5.2	3D images	33
5.2.1	Test 3D images	33
5.2.2	Medical images	40
5.3	Discussion	41
6	Conclusions	44
	References	48

List of Figures

1.1	Eye structure	3
1.2	MRI anatomy of the cruciate ligaments	5
2.1	Directional histogram	7
2.2	Tamura method slope calculation	8
2.3	One level of a wavelet decomposition in three steps	10
3.1	Fourier domain rotation	14
3.2	3D Fourier transform explanation	15
3.3	Edge blur in spatial and frequency domain	18
3.4	Windowing effect	20
4.1	Application main window, 2D mode	21
4.2	Processing result, 2D mode	22
4.3	Application main window, 3D mode	23
4.4	Line fitting using Bresenham algorithm	24
4.5	Line fitting using Xiaolin Wu algorithm	24
4.6	Plane tracing performance improvement	26
5.1	2D sample image: originals	28
5.2	2D sample image: processing result	29
5.3	2D sample image: DC removal effect	30
5.4	2D real image: settings	30
5.5	2D real image: no enhancements	31
5.6	2D real image: windowing, DC removal, and normalization applied	32
5.7	2D real image: edge detector applied	33
5.8	3D sample image: sphere	34
5.9	3D sample image: skew lines	35
5.10	3D sample image base: sine wave intersection	36
5.11	3D sample image: sine wave intersection	37
5.12	3D sample images base	37
5.13	3D sample image based on the image 5.12(a)	38
5.14	3D sample image based on the image 5.12(b)	38
5.15	Cross-section of the right pane of the image 5.14	39
5.16	3D sample image based on the image 5.12(c)	39
5.17	3D real image: settings	40
5.18	3D real image: processing result	40
5.19	3D real image: edge detector applied	41

List of abbreviations and symbols

DC	‘Direct Current’, or component with 0 Hz frequency
FFT	Fast Fourier Transform
GUI	Graphical user interface
CT	Computed Tomography
MicroCT	Micro-Computed Tomography
MRI	Magnetic Resonance Imaging
ROI	Region of interest
TUT	Tampere University of Technology

1 Introduction

Modern medicine relies on visual data. Regarding data acquisition, numerous imaging techniques have been developed over the last century. However, until recently manual human check was considered as the only method of data analysis. Since most of the modern imaging techniques provide output in digital format, computer-aided analysis can be applied in addition to the traditional visual check. Moreover, the imaging system itself may contain additional processing tools used for noise reduction, artifact removal, reconstruction, etc.

Computer vision techniques for quantitative or objective analysis are the major strength of computers [1]. Visual analysis of medical images, while it is still the most common approach, limits the correctness of interpretation due to the human factor. It includes intrapersonal perception variations, environmental distractive factors, fatigue originated errors, errors caused by unclear representation of the abnormality in a screening application, etc. Thus, visual analysis of medical images is always performed with considerable subjectivity. Common practice is to collect opinions from multiple experts. The use of computer-aided analysis has the potential to add objectivity, reproducibility and accuracy to the assessment process. Though it may not fully substitute the human-performed analysis, it can still become the main tool of assessment process facilitation.

Medical analysis algorithms development are based on the type of imaging system in question. Based on the digital data obtained from the imaging equipment either 2D image (X-Ray, MRI, Ultrasound), or a number of 2D projections (Ultrasound, fMRI) or slices, or 3D image (CT) can be formed. In addition, the acquired data depend on the applied imaging technique, such as sequential segmentation technique to detect thin vessels in medical images.

Common features of disorders detected by medical imaging technique are structural changes of the examined tissue. To illustrate how medical imaging can be used in practice, let us look at the examples of optic nerve imaging and cerebrovascular system tomography.

1.1 Optic nerve imaging

Optic nerve (figure 1.1) imaging techniques, such as optical coherence tomography (OCT), are frequently used to diagnose eye-related disorders. Such images can provide quantitative information for disease progress screening for all types of glaucoma, as its main cause is optic nerve degeneration [2].

A healthy optic nerve is composed of approximately one million individual nerve

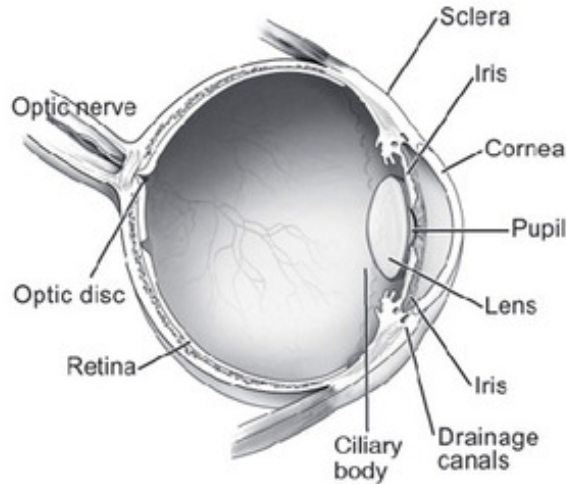


Figure 1.1 Eye structure [3]

fibers coming from the retina. The fibers bend about 90 degrees on their way from retina to the optic nerve head [3]. Normally, the diameter of notch at the front of the optic nerve head is smaller than the diameter of the optic nerve [4]. In the case of glaucoma, the diameter of the cup increases due to individual fibers degenerating along the nerve, and this is visible in the OCT image.

In general, there is a wide range of optic nerve pathologies, such as nutritional, toxic, and inherited optic neuropathy[4], for which computing analysis of the retrieved images might be a useful mean for diagnostics.

1.2 Cerebrovascular system tomography

CT imaging of human cerebrovascular system is another example of a promising technique to provide a quantitative morphological assessment of vascular structure at high spatial resolution.

The cerebral vascular network disorder can be a cause of variety of brain pathologies, such as ischemic and hemorrhagic strokes, neurodegenerative diseases, and brain tumors. Histological imaging of arteries is a standard approach for high resolution examination of vascular system. Unfortunately, the technique itself can be applied as ex vivo analysis only and is unsuitable for macroscopic analysis of vascular morphometry of the whole brain circulation, or for any kind of longitudinal studies.

The CT and other X-ray based techniques are intended for imaging of live animals. X-ray imaging techniques are do not depend on flow which means that potential issues of non-uniform signal intensity or vessel visibility, caused by variable blood flow, are eliminated. The high level vessel visibility is a key requirement for quantitative vascular morphometry. [5].

1.3 Directionality in medical imaging

Both eye and cerebrovascular structure described above transfer either signal or substance between different parts of the body. Forming the streamlines they show distinct directionality on medical images. In case when the integrity of a structure is damaged the defected area is seen as a pattern with orientation different from healthy areas of the tissue. Therefore, in terms of diagnostic imaging the directional features can be used for abnormalities detection and for defining the reference in image alignment.

However, most of image analysis techniques are noise-sensitive. It makes the preprocessing stage crucial for the analysis validity. General methods of image processing, such as noise filtering, contrast enhancement, bias correction, etc. are applied first, followed by more specific techniques.

As it was described previously, directional features can be used for medical images assessment. However, in order to characterize and assess the underlying pattern there should be at least one concrete feature defined. For this purpose the term of directionality was introduced. The directionality of a pattern can be expressed as an axis along which physical or mechanical properties of the underlying structure remain unchanged.

It is worth mentioning that there is a significant difference between macro- and micro-scale image analysis. For macro-scale, the directionality of a blood vessel image with clearly detectable contours can be easily defined and interpreted. For micro-scale images, the pattern within the region of interest (ROI) is represented by mostly homogeneous structures. In this case, directionality vectors can spread uniformly within 360° range. In any case, the presence of a control image is necessary for directionality assessment of micro-scale images. Basically, the conclusion on abnormality can be done based on variation between the current directionality value and the control image [6].

The anterior cross ligament injury case is an illustrative example. figure 1.2 shows anterior cross ligament MRI in healthy (figure 1.2(a)) and teared (figure 1.2(b)) states. On figure 1.2(a) the healthy ligament can be seen as an easily distinguishable line that connects femur to tibia (see arrows). On figure 1.2(b), the ligament is not connected to femur, and the directional line observed in figure 1.2(a) is no longer visible. Directionality patterns of healthy and injured images will differ, and this difference can be used as abnormality indicator that will suggest further diagnostics of the injury.

The aim of this thesis is to provide a tool for image directionality evaluation. As the biological structure can be represented in both ways as a plane image and as a volume, the feasibility for 2D and 3D analysis is the key requirement to the

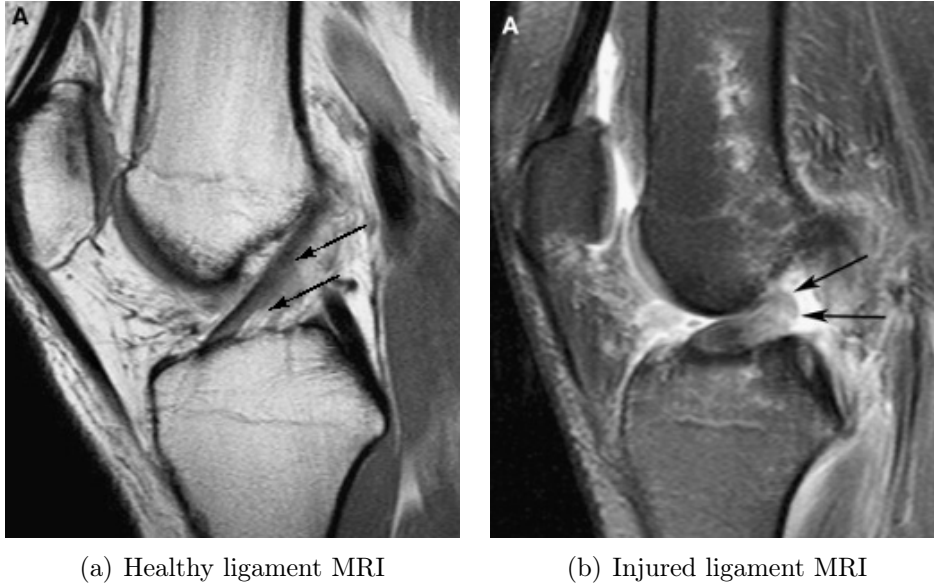


Figure 1.2 *MRI anatomy of the cruciate ligaments.
Healthy ligament has more distinct directional
features than the injured one
Source: [6]*

developed tool. In addition, it should be supplied with GUI to make it available for end-user. The further application of this tool for assessment of medical images can be considered as a subject of a separate thesis.

2 Background

The research on image directionality is usually related to structure anisotropy studies, especially to construction materials properties research. According to [7], structure anisotropy, or dependence of structure properties on the particular direction defined by angle α , can be characterized using function $f(\alpha)$ that describes the total length of collinear components.

One way to define such function, described in [8], is based on 2D Fourier transform. In frequency domain, an image is represented by values of its components correlating with gradient of image function in spatial domain (see section 3.1). All components of the Fourier frequency spectrum are compiled in the directional vector of angle α . The result can be visualized in a polar diagram, which represents an estimate of the target function $f(\alpha)$ and is called rose of directions. The rose of direction diagram will be widely used in the following of this thesis (see Sections 4 and 5).

The advantages of this method are its speed and ease of visualization. The result is immediately available after the algorithm application to an 2D image [8]. The main obstacle in the implementation of this approach is the impossibility to quantify the result. The only way to perform an assessment is visual comparison between current image and reference. Sufficient resolution of an image is another strict condition which assures the correctness of resulting graphical representation of structural directions distribution.

To evaluate directionality of the image, the conventional Tamura method [9] or one of its derivatives can be applied. The directional histogram (see example figure 2.1(a)) is calculated in a way similar to the previous approach. However, some additional steps described in [10] can help to quantify and evaluate obtained results.

On figure 2.1, angles in the range of -90° to 90° are divided into 12 bins, and vertical axis shows the percentage of pixels with different gradient angles. Assuming that the edges oriented at -90° are similar to those oriented at 90° , the angles can be arranged in a circle, as it is described on the figure 2.1(b).

Once the diagram is re-arranged, the method calculates the sharpness value for each angle using the geometric slope between histogram bins (see figure 2.2). The sharpness can be derived as a weighted sum of slopes of all linear segments joining bin tops (see equation 2.1 [10]). The next step is to calculate directionality as

weighted sum a of sharpness values from all the hills in the histogram.

$$sharpness = \sum_i weight_i * slope_i + \sum_i weight_{r_i} * slope_{r_i} \quad (2.1)$$

where

$$i = \frac{circularDifference(peak_{pos}, bin_k)}{min(circularDifferenceBetweenTwoBins)} \quad (2.2)$$

Here *circularDifference* is distance between bins computed on the histogram once it has been rearranged as a circle (see figure 2.1(b)). It differs from the linear distance, as left and right bins on figure 2.2 are in fact neighboring after histogram rearrangement.

$$weight_{i_k} = 2^{-i+1} \quad (2.3)$$

Then we can define directionality as

$$Directionality = \sum_{for\ each\ hill, h} weight_h * sharpness_h \quad (2.4)$$

Directionality feature is also used in scope of image recognition methods. Local Directional Texture Pattern (LDTP) is a feature descriptor introduced in [11] which combines directionality and intensity analysis. This becomes possible to define the main directions from the local neighborhood, and to extract the information about a prominent relative intensity. Therefore, while using LDTP, the neighborhood is characterized by a combination of these two features in one code. Specifically, the structural information of a local neighborhood is encoded with LDTP via analyses of its directional information and the intensity values difference for the first and second maximum edge's responses.

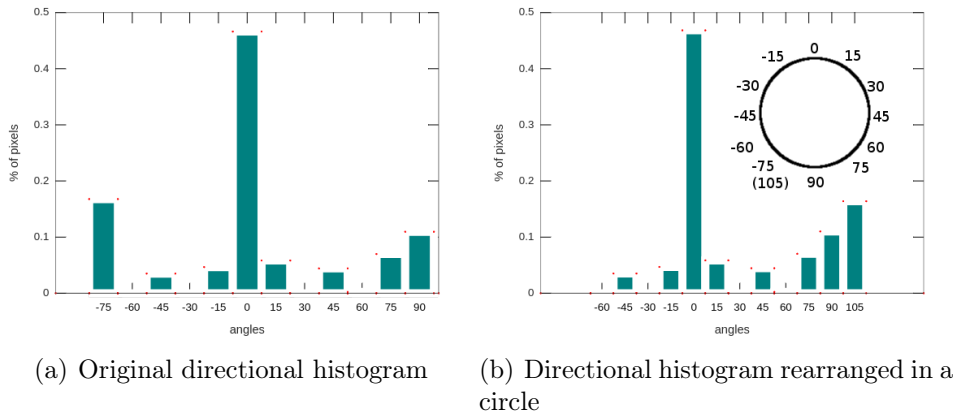


Figure 2.1 Directional histogram

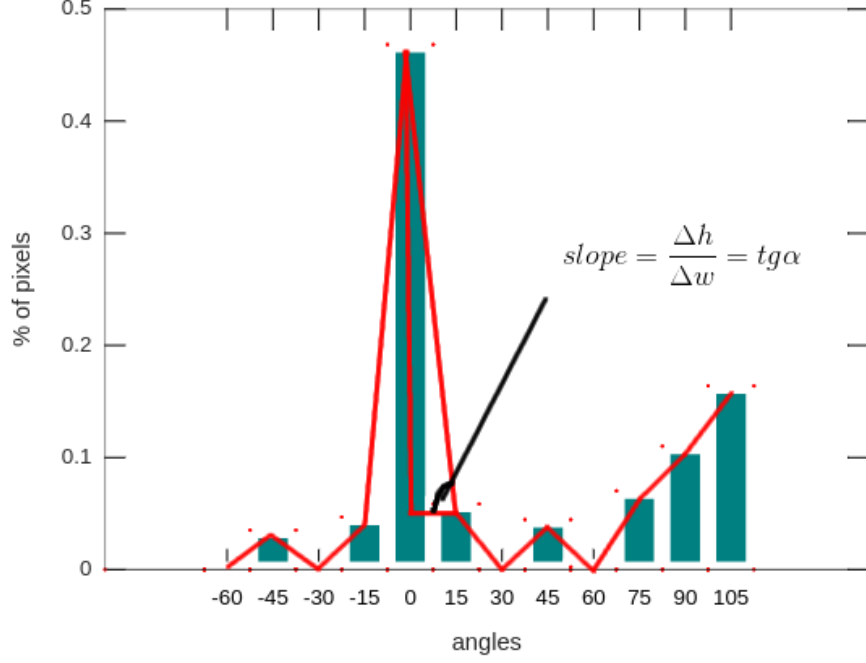


Figure 2.2 Tamura method slope calculation

The LDTP can be derived by calculating the principal directional numbers of the neighborhood [12] in eight different directions using the Kirsch compass masks. The directional number is defined as

$$P_{Dir}^i = \operatorname{argmax}_i (II_i | 0 \leq i \leq 7) \quad (2.5)$$

P_{Dir}^i is the principal directional number, II_i is the absolute value of the response of convolution between the image intensity matrix I and i -th Kirsch mask M_i :

$$II_i = |I * M_i| \quad (2.6)$$

The greatest two directional numbers are taken out of the eight produced by Kirsch mask application. These directions define the principal axis of the local texture [11]. The intensity difference of the opposite pixels within the neighborhood area is calculated in each of principal directions.

$$d_n^{(x,y)} = I(x_{P_{dir}^n+}, y_{P_{dir}^n+}) - I(x_{P_{dir}^n-}, y_{P_{dir}^n-}) \quad (2.7)$$

d_n is the n -th difference for the pixel (x, y) in the n -th principal direction; $I(x_{P_{dir}^n+}, y_{P_{dir}^n+})$ corresponds to the intensity of the next pixel in the given principal direction $(x_{P_{dir}^n+}, y_{P_{dir}^n+})$, and $I(x_{P_{dir}^n-}, y_{P_{dir}^n-})$ is the intensity value of the the previous pixel in the given principal direction $(x_{P_{dir}^n-}, y_{P_{dir}^n-})$.

Then each difference is encoded as

$$D_f(d) = f(x) = \begin{cases} 0 & -\epsilon \leq d \leq \epsilon \\ 1 & -d < -\epsilon \\ 2 & d > \epsilon \end{cases} \quad (2.8)$$

Where D_f is the encoded intensity difference, d is the actual intensity difference, and ϵ is a configurable threshold value.

The binary form of the principal direction is concatenated with the two differences forming the encoded image. It can be represented by the following equation.

$$LDTP(x, y) = 16P_{dir}^{1(x,y)} + 4D_f(d_1^{(x,y)}) + D_f(d_2^{(x,y)}) \quad (2.9)$$

The main advantage of this approach is that LDTP includes only principal information on the micro-pattern [11]. It avoids risks of errors due to noise appearance while processing all disposable information.

An image texture can also be analyzed with the wavelet multi-resolution approach. While the Fourier domain represents only frequencies, the wavelet transform keeps both frequency and spatial information. The method is based on decomposing 1D signal into a basis of wavelet functions [13].

$$(W_a f)(b) = \int f(x) \psi_{a,b}^* dx \quad (2.10)$$

This kind of a basis is obtained by translation and dilation of a single wavelet ψ .

$$\psi_{a,b}(x) = \frac{1}{\sqrt{a}} \psi \frac{x-b}{a} \quad (2.11)$$

To do this, wavelet ψ should be defined in both spatial and frequency domains, and it should have zero mean. The discrete wavelet transform (DWT) is obtained once

$$a = n^2, b \in Z \quad (2.12)$$

The efficient implementation of DWT in real space is accompanied with the usage of quadrature mirror filters [13]. The wavelet decomposition of a 2D image can be obtained by consecutive filtering along horizontal and vertical directions.

This filtering results in four images per one level of decomposition. Each subimage can be subsampled by a factor of two, that is, keeping a possibility of complete reconstruction. Therefore, the resolution is the same as of original image. The DWT provides an easily understandable representation. The information of a specific scale and orientation is conveniently separated and stored in the respective

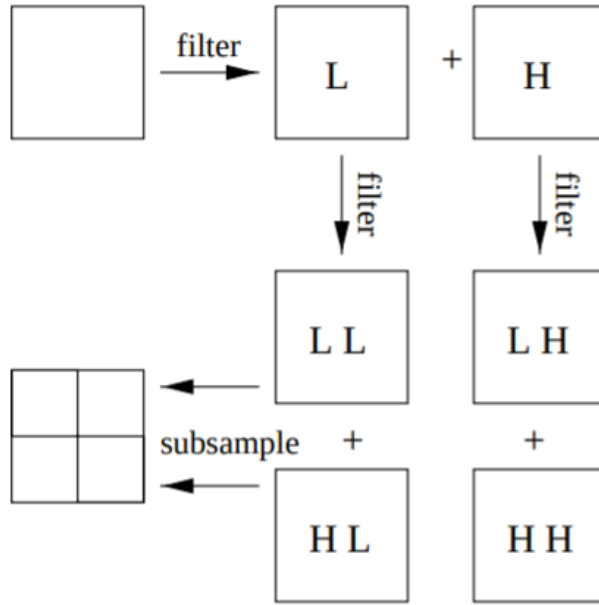


Figure 2.3 One level of a wavelet decomposition in three steps:
 1. Low and High pass filtering in horizontal direction,
 2. The same in vertical direction,
 3. Subsampling
 Source: [13]

subimage. Spatial data is also retained within the subimages. In order to obtain features that reflect scale-dependant properties there is a possibility to extract a feature from each subimage separately. The drawback of the method is that a large number of features, makes classification and segmentation much more difficult [13]. This approach is also not optimal for defining a directionality of an image due to its complexity. Classic DWT has some other drawbacks as well. For image processing, most notable ones are the shift variance and the artifacts produces for non-axial features [14]. To address these issues, additional steps, such as combining two DWTs with different filters [15], or rotating the original image multiple times while taking snapshots of the now-axial features for each rotation step, and then combining these features with their respective angles [16].

Another approach to 3D texture characterization is based on generalized co-occurrence matrices. The methods based on this approach estimate local anisotropy directly from the image and not from its local derivatives [17]. One of the oldest methods was proposed by Longuet-Higgins in [18]. It was made for estimating the elevation distribution on gaussian moving surfaces, originally for sea waves. Another method was used by Cooper and Graham in [19] for estimating motion in ultrasound medical images. This method relies on the optic flow estimation proposed by Heeger in [20, 21]. Heeger's method is based on spatiotemporal filtering, involving more

than one frame into a single algorithm step. Another method, proposed by Sato and Cipolla in [22], relies on local texture moments to estimate the orientation of the surface. Chetverikov [23] and Haralick [24] proposed the usage of features which were extracted from gray level difference histograms in order to characterize the anisotropy. However, the gradient-based method tends to be more popular for micro-texture characterization.

3 Materials and methods

All methods described in the section 2 work well with 2D images. One of the aims of this thesis is to provide a tool capable of processing 3D images. For this reason, one of the methods described in section 2, the FFT-based directionality assessment method described in [8], was taken as a reference. This method relies on the Fourier transform feature that each sharp line in spatial domain is transformed into an orthogonal line in frequency domain.

3.1 Fourier transform of a line in 2D and 3D

Many sources, such as [25] and [26], employ a particular property of Fourier transform that can help to define image directionality. If Fourier transform is applied to a 2D image, each line of this image in spatial domain will be transformed into the line orthogonal to it in frequency domain. In most cases, this property is presented without proper mathematical explanation, which makes it hard to scale it into higher dimensions.

In this thesis both an intuitive and mathematical explanation will be provided.

Intuitive explanation of this property is quite straightforward. When moving in direction orthogonal to the line, we will see a sharp edge, and it will take infinite number of sine waves to describe it. This kind of explanation gives us some insight on 2D case, but using it for higher dimensions is still questionable, at least because, even in 3D case, there is an infinite number of lines orthogonal to the given one.

Mathematical proof relies on Fourier transform's linearity and separability [27]. The proof itself will be done for a 2D case, with a brief explanation of 3D extension in the end.

Assume some 2D line for which we can easily apply transform. For example, let us think of a horizontal line at the very top of the image. Such a line can be described as

$$f(x, y) = \begin{cases} 1 & y = 0 \\ 0 & y \neq 0 \end{cases} \quad (3.1)$$

We can apply 2D Fourier transform as two 1D transforms, this property is called separability. First, we take transform of each row, and then take yet another transform of each column of the first transform's result. This is so-called 'row-column' method.

1. Rows:

After applying 1D Fourier transform to each row U ,

$$F'(U, V) = \begin{cases} F(1) = \delta(x) & U = 0 \\ F(0) = 0 & U \neq 0 \end{cases} \quad (3.2)$$

Expanding the delta function, we can say that

$$F'(U, V) = \begin{cases} \infty & U, V = 0 \\ 0 & U, V \neq 0 \end{cases} \quad (3.3)$$

Which means that we can say we have delta function not only in the first row $U = 0$, but also in the first column $V = 0$, just using the definition of delta function.

$$F'(U, V) = \begin{cases} \delta(V) & V = 0 \\ 0 & V \neq 0 \end{cases} \quad (3.4)$$

2. Columns:

Following the ‘row-column’ method, let us take the same transform column-wise,

$$F(U, V) = \begin{cases} F(\delta) = 1 & V = 0 \\ F(0) = 0 & V \neq 0 \end{cases} \quad (3.5)$$

Which is basically a vertical line.

In this way, we proved that the Fourier transform will project a horizontal line at the very top of the image into a vertical line. Next, we will need to show what will happen if we will rotate or/and shift this line.

For rotation, we can just say that Fourier transform is linear, thus the frequency domain image will rotate exactly the same way as the spatial domain one [28]. But mathematical proof would still be beneficial, especially because it is quite easy to formulate it not for 2D, but for N-dimensional space in general.

Let us write an N-dimensional Fourier transform for some linear N-dimensional signal $f(X)$.

$$F(\xi) = \int_{R^N} f(X) e^{-2\pi i X \xi} dX \quad (3.6)$$

Now, let us rotate the signal X using rotation matrix A , and denote the result

as signal Z .

$$Z = AX \quad (3.7)$$

As rotation matrix is orthogonal, we know that $|\det A| = 1$. Then, $dZ = |\det A|dX = dX$.

From here, we can prepare the transformation of the power of the Euler function in the transform equation as

$$X\xi = A^T Z\xi = ZA\xi \quad (3.8)$$

Now, let us calculate Fourier transform for function $g(X) = Af(X) = f(AX)$

$$\int_{R^N} f(AX)e^{-2\pi i X\xi} dX = \int_{R^N} f(Z)e^{-2\pi i Z A\xi} dZ = F(A\xi) = AF(\xi) \quad (3.9)$$

$$F(Af(x)) = AF(\xi) \quad (3.10)$$

Which means that the Fourier spectrum will rotate the same way as the original spatial domain signal. Consequently, if the original frequency domain image was orthogonal to the spatial domain one, it will continue being orthogonal.

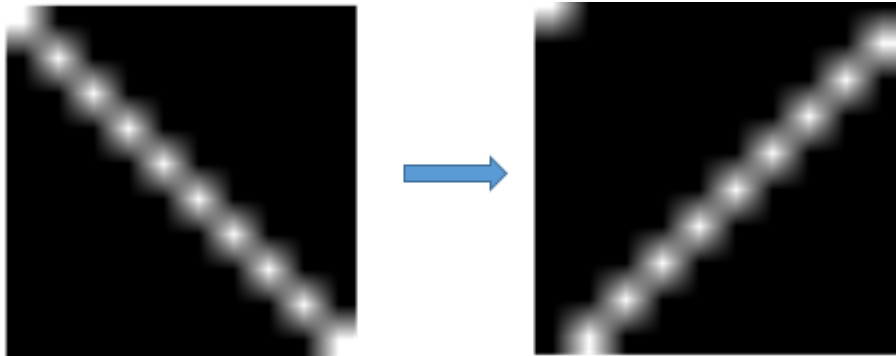


Figure 3.1 *Fourier domain rotation:*
left image - spatial domain, right image - Fourier domain

You can see the DC value that was left out in the upper left corner of the Fourier domain image. If we swap image quarter-parts to move DC to the center of the image, we will get a nice orthogonal line.

Next, let us talk about shifting the line. In any Fourier transform standard transforms table [28] we can find the following property:

$$f(X - X_0) \rightarrow e^{-2\pi i X_0 \xi} F(\xi) \quad (3.11)$$

When plotting signal or image in Fourier domain, we usually plot its absolute value. The absolute value of multiplication equals the multiplication of its parts'

absolute values. The absolute value of this added exponent part is 1, as it is yet another Euler function. Which means that shift in spatial domain only produces a phase shift in the frequency domain, but it does not affect frequencies and their absolute values. The line will always start from the DC value, or, if the image quarter-parts are shifted, it will go through the center of the image.

With this we can end our proof of having an orthogonal line in frequency domain for each line in spatial domain. The only addition to make is that due to Fourier transform linearity we can always represent a transform of set of lines as a set of transforms of those lines.

$$F(f(x) + g(x)) = F(f(x)) + F(g(x)) \quad (3.12)$$

Now let us expand this proof into the third dimension. The 2D Fourier principle of separability can be used for this purpose. It states that the discrete Fourier transform in 2D can be obtained in two steps by successive applications of the 1D Fourier transform. That is, using the separability principle, we can say that 3D Fourier transform is a 1D Fourier transform of a 2D Fourier transform along each line orthogonal to the 2D transform plane. This conclusion can be supported with the visual explanation below.

Visual explanation is given on figure 3.2. The leftmost cube is the original image in the spatial domain. Let us suppose that there are "ones" along the front left edge and "zeroes" in the rest volume of the cube. Performing a 2D Fourier transform for this non-zero edge first along the vertical axis and second, along the horizontal axis, we will end up with the central image, as it was proven before with the "row-column" method. We do not consider the rest of the cube volume since the rest of the volume will be zeroed out, as there is no signal anywhere else except this non-zero edge.

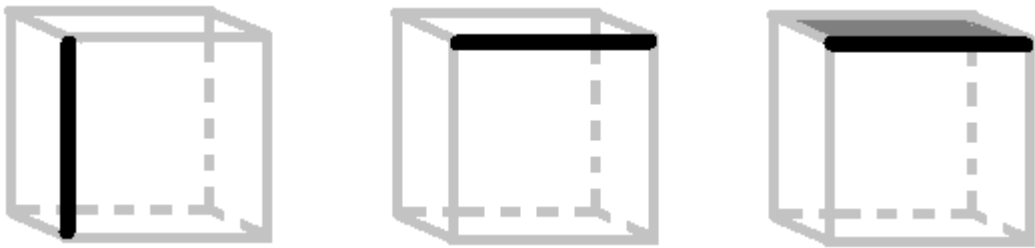


Figure 3.2 3D Fourier transform explanation.

A line in a spatial domain (the leftmost image) will be transformed into a N-1 dimensional plane in a Fourier domain with the original line direction serving as a normal for this plane (the rightmost image)

Furthermore, performing a Fourier transform in third dimension we consider the line of the central image as a number of single dots. These dots can be considered

as delta functions divided by some constant. The dots will turn into lines after the Fourier transform, and altogether will form a plane (grayed out on the rightmost cube).

With this we can expand the Fourier transform ‘orthogonal line’ property a bit further: a line in a spatial domain will be transformed into a N-1 dimensional plane in a Fourier domain with the original line direction serving as a normal for this plane.

In fact, it is easy to prove the duality of this claim: N-1 dimensional plane in spatial domain will be translated into a line orthogonal to it in Fourier domain. The formal proof is almost identical.

As it was mentioned previously, strict 3D expansion using 3D Fourier transform is not the only way of using the proposed method in 3D. Two more methods were tested in scope of this thesis.

The first way was quite straightforward. A 3D image is essentially a stack of 2D layers. It is possible to process the image layer-by-layer, and represent the result in a way similar to how the original image was constructed, as a stack of processed 2D layers. The main drawback of this method is an inability to process non-horizontal features. For example, let us imagine a cube with a diagonal line inside, sliced into multiple horizontal layers. Each slice will contain only a single point at which the diagonal crosses the slicing plane. A single pixel does not contain any directional information. Its frequency domain representation will be a constant magnitude in every point of the slice. Yet, this method might work well if an image has some distinct axis, such as spinal cord, as we are interested in features strictly orthogonal to this axis.

Second way was to use 2D slices with a rotating plane. This method is quite similar to the previous one, with the only difference that layer-by-layer processing is changed for rotation. It works well to determine radial outliers, for example when we want to inspect possible knee bone artifacts.

3.2 Reference method

The reference method described in [8] is based on evaluating the power spectrum density (PSD) of the image using its frequency domain representation. For a 2D case, the PSD can be described as:

$$P(u, v) = |F(u, v)|^2 \quad (3.13)$$

Where $F(u, v)$ is the result of applying Fourier transform to the image.

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})}, \quad (3.14)$$

Here, $f(x, y)$ is the original image represented.

$F(u, v)$ can be divided into real part $R(u, v)$ and imaginary part $I(u, v)$. Therefore, the $|F(u, v)|$ is defined as:

$$|F(u, v)| = \sqrt{R(u, v)^2 + I(u, v)^2} \quad (3.15)$$

For simplicity the spectrum can be scaled to 8-bit gray levels so the $P(u, v)$ can take the following form:

$$P(u, v) = \log(1 + |F(u, v)|^2) \quad (3.16)$$

With this we can define directionality as sum of PSD in some direction α .

$$\rho(\alpha) = \sum_{r=0..R} P(r * \cos(\alpha), r * \sin(\alpha)) \quad (3.17)$$

We can rephrase this problem as the ‘sum of values of the ray traced from the center of the image PSD’. In 3D case it will become not a ray but a plane orthogonal to the original line, which passes through the center of the image.

Ray tracing is a common mathematical problem which has a number of solutions. In the reference method only one ray tracing algorithm is used. In the applied method for ray tracing two algorithms are implemented. Inputs and outputs of both implementations are compatible with each other, and the option to switch between them is available on the final tool.

More details are mentioned in the ‘Implementation’ part (Section 4).

3.3 Image enhancement techniques

Some additional image enhancements were implemented to increase the algorithm efficiency. These enhancements include image normalization, edge detection with hole filling, DC removal (i.e. removal of the image component with 0 Hz frequency), and windowing [29, 27].

Image normalization

Image normalization performs a simple linear scaling of the image values from the original value range to [0, 255]. It helps to process low contrast images in some cases, making edges more visible to the algorithm.

Edge detection

In this thesis, edge detection relies on Prewitt edge detector algorithm, with a binary image as a result. After a number of tests it was found that the algorithm is more suitable for images with sharp edges, and smooth images with blurry edges yield less accurate results. Use of binary images achieves better results, as they have no blur.

The influence of edge sharpness is illustrated on figure 3.3. Black and white image with a sharp edge is blurred with moving average with different kernel sizes. Note the diagonal line orthogonal to the edge becoming less visible as the kernel size increases. The nature of cross-shaped artifacts, also visible on the figure, will be discussed later.

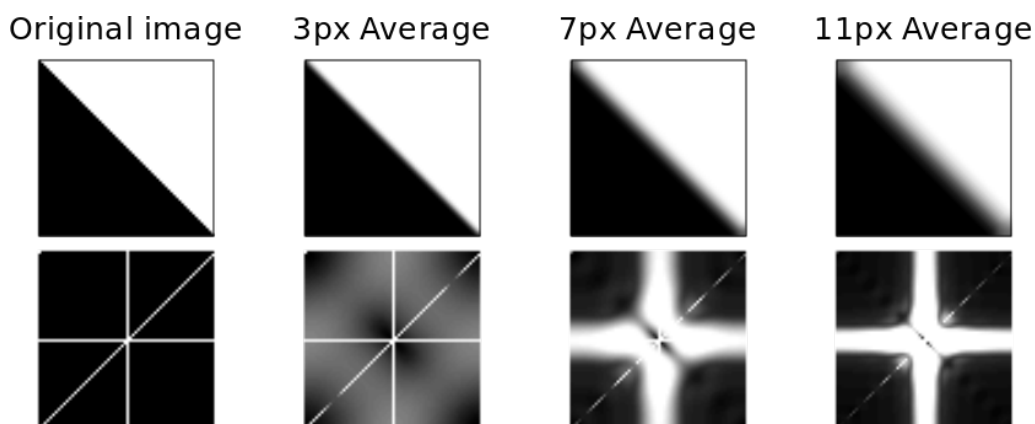


Figure 3.3 *Edge blur in spatial and frequency domain. The developed algorithm is more suited for images with sharp edges, and smooth images with blurry edges yield less accurate results*

By applying edge detector to the image we can transform it into a binary image with only sharp edges left. The definition of "sharpness" depends on the edge detection algorithm. For example, many edge detectors rely on the pixel intensity gradient to find edges [27].

Hole filling

Hole filling features can be used together with edge detection if the image is too noisy in terms of edges. This program uses MATLAB internal `imfill()` function which closes up all regions with distance between edges lower than the threshold. This function works with binary images, so one typical application for this option is setting the edge detector threshold a bit lower on noisy images to prevent noise from appearing on binary images, and then connecting the edges back together. Having edges continuous can help reducing the Fourier domain noise caused by the sharp ends of the broken edge.

DC removal

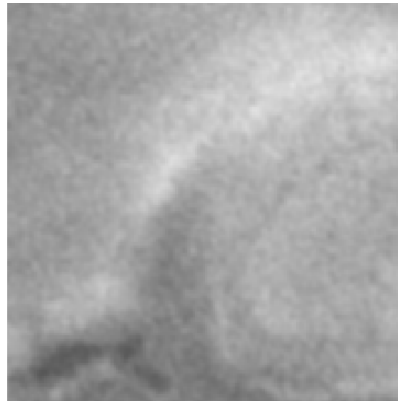
DC removal option sets the DC value, located in the cell with coordinates (0, 0) in frequency domain, to zero [27]. Mathematically, the DC value is equal to the sum of all pixel values of the image. Because of that, DC is usually the biggest image frequency domain value in most cases, and it can easily saturate all other directional components, making the rose of directions look like a circle. Removing it will greatly increase algorithm sensitivity, but it will also make small noise more visible, which can also saturate the result in case of a noisy image with blurry edges. Section 5.1.1 gives example of how exactly DC value can affect the result.

Windowing

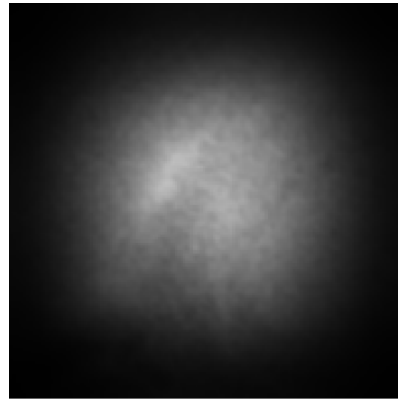
One of the limitations of the Fourier transform is that it is applicable only to infinite signals. Fast Fourier Transform (FFT) overcomes this issue by assuming that the dataset is periodical, or, strictly speaking, is duplicated along its borders in all directions [27].

Most of the real world images do not have that kind of periodicity (figure 3.4(a)), which results in the sharp edge on the border, which consequentially transforms into the cross-shaped artifact in the Fourier domain (figure 3.4(c)). The main idea of windowing is to downscale the image intensity with the distance from the image center using some sort of smoothly decreasing function, such as Gaussian. One need to assure that the informative part of the image is in the center of ROI while using it. In this thesis, Hamming window is used for 2D images, and Gaussian window is used for 3D ones.

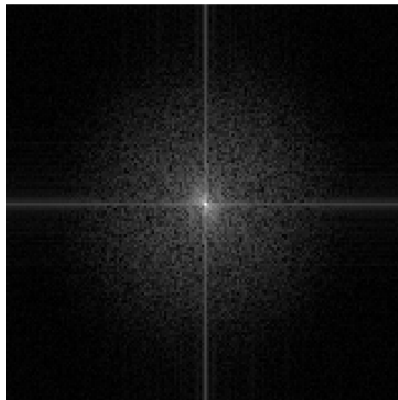
All image enhancement techniques mentioned can greatly affect the result quality, which will be shown later in part 5. As some of these techniques require parametrization, e.g. threshold definition, they add additional complexity to the initial problem.



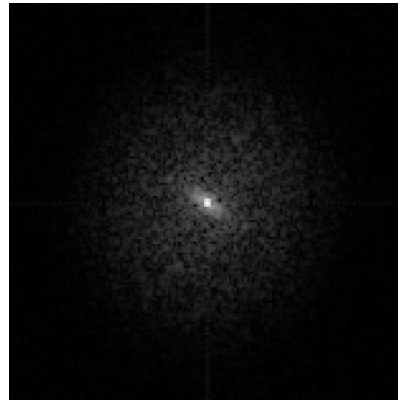
(a) Original image w/o windowing



(b) Original image with windowing



(c) FFT w/o windowing



(d) FFT with windowing

Figure 3.4 *Effect of windowing on FFT*

4 Implementation

4.1 Interface

Developing a GUI was one of the key outcomes of this project. The GUI was created using MATLAB's internal *guide* system (see figure 4.1). 2D and 3D image load buttons were kept separate, since they enable different tools and views. Controls unavailable in the chosen mode (2D or 3D) are either grayed or hidden.

ROI selection box appears upon checking the 'Region only' box. The box can be dragged by the edge, resized, and removed by un-checking the box.

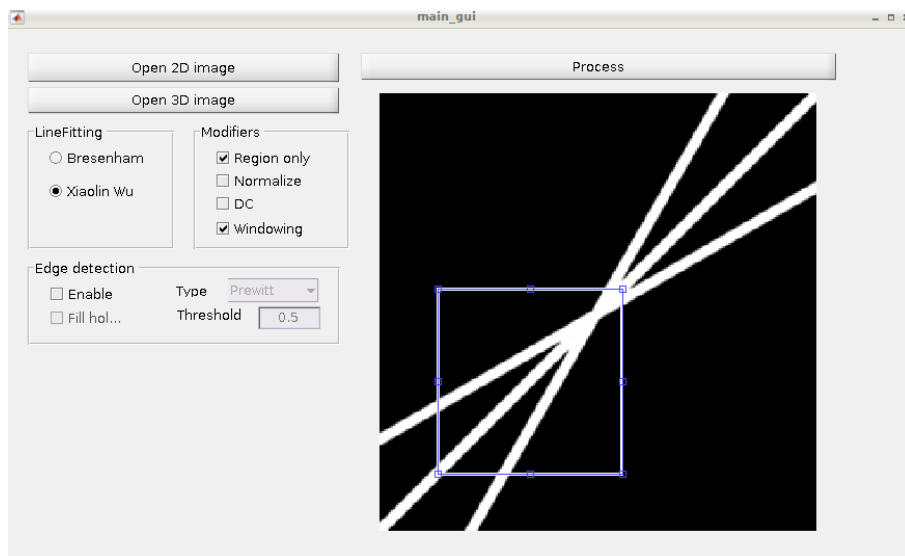


Figure 4.1 Application main window, 2D mode

After pressing the process button, one more window will appear showing the computation result. For 3D images, a progress bar will appear first, as processing a 3D image might take a considerable amount of time.

2D spatial domain images are shown as-is, and 3D images are constructed slice-by-slice. 3D image viewer function has transparency as an additional parameter. Making the image semi-transparent can greatly improve both visual experience and readability of the image, but it will consume additional GPU resources.

Result windows for both 2D and 3D images consists of four subplots arranged into 2-by-2 grid (see figure 4.2). The upper left subplot is the original image after all spatial domain enhancements were applied. Upper right and lower left images show shifted FFT and $\log(1 + \text{abs}(FFT))$ of the original image. Lower right image show the rose of directions, which is the result of the computation. FFT images might not be that relevant to the actual result, but they can be very useful for tracing down

artifacts and deciding on image enhancement techniques used to mitigate them. In addition, the result window was intentionally left as a standard MATLAB plot viewer, as it has Plot Tools as its toolbar's rightmost button. Plot Tools allows shaping and styling the image in the preferred way.

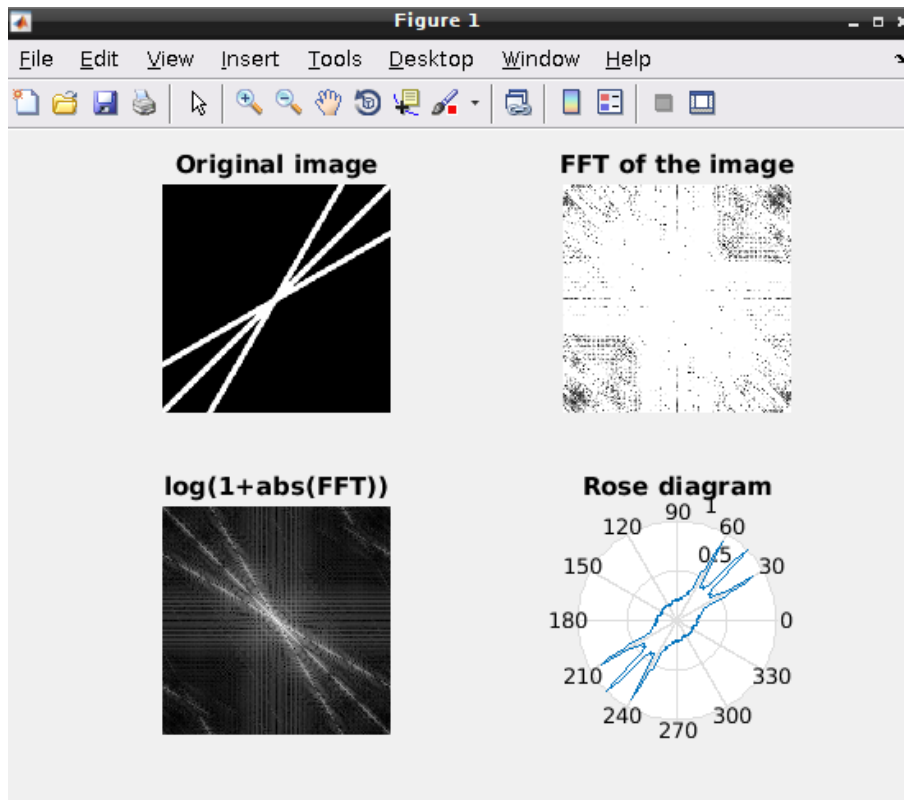


Figure 4.2 Processing result window in 2D mode. Original (top-left), magnitude of the FFT (top-right), log-scaled magnitude of the FFT (bottom-left), and the processing result (bottom-right)

For 3D images, the interface slightly differs. First, some features are not supported in 3D mode. Their checkboxes gray out. Second, it features one more control panel titled ‘3D visualization control’. This panel serves two different purposes. It helps to partially adjust the image by setting value thresholds, which might be crucial if the image contains salt and pepper noise or similar artifacts. It also helps to adjust the visual experience by adding or removing transparency, reducing the image quality, and using interpolation to make the image smoother. These options are needed because MATLAB viewer can be resource-demanding when dealing with 3D images.

The second difference is that only the upper hemisphere of the result is shown. It is done because of performance considerations, as fully rendering completely symmetrical data would be too resource-consuming.

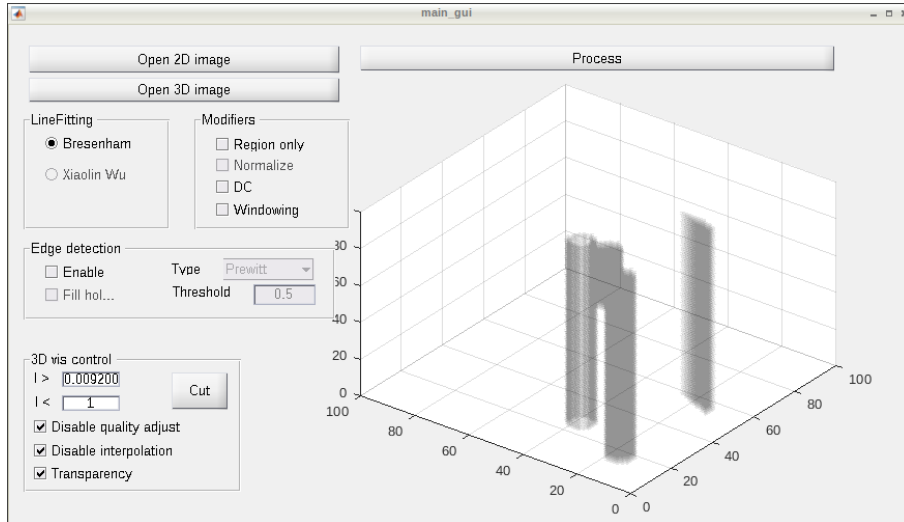


Figure 4.3 Application main window, 3D mode

4.2 Ray tracing algorithm

The main feature to implement, as well as the base of the whole algorithm, was ray tracing. In 3D we can actually call it ‘plane tracing’, as we fit a plane, not a line. Two different algorithms were implemented: Bresenham algorithm and Xiaolin Wu algorithm.

Bresenham algorithm [30] constructs a sharp-edged ladder of pixels following the line direction. For each x it calculates the y coordinate based on line equation:

$$y = k * x + b \quad (4.1)$$

Or, using two points:

$$y = \frac{x - x_0}{x_1 - x_0}(y_1 - y_0) + y_0 \quad (4.2)$$

Then the pixel with y coordinate being closest to the computed one is selected.

If the line is too steep, one x step will correspond to several y steps. In this case, some pixels can be missed. There are two ways to address this issue. The first way is to repeat the same algorithm with x and y coordinates swapped. The other way, used by many implementations, is to rotate the plane so that the target line will move to the first octant. Yet another issue of this algorithm is that it may be not very accurate, especially on low-resolution images (figure 4.4).

Xiaolin Wu algorithm, first introduced in [31], is a method frequently used to perform anti-aliasing in computer graphics. It creates not a ladder, but a cloud of weighted values for each pixel (figure 4.5). The actual line value can then be reconstructed as the sum of values multiplied by their weights.

The algorithm first rotates the line segment so that it will be inclined to the

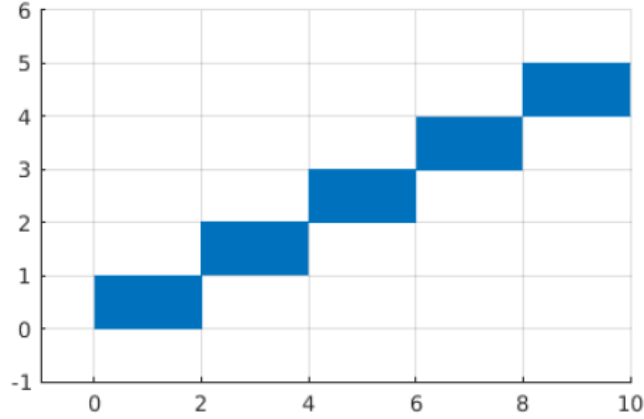


Figure 4.4 Line fitting using Bresenham algorithm

positive direction of the x axis at a positive angle of no more than 45° . This step is crucial, otherwise a number of conditions should be introduced to handle different line directions without skipping any pixels. Then, for each x , the y coordinate is calculated from the line equation. A fractional part of the y coordinate is then used to calculate weights for pixels located higher and lower than the actual y value.

$$\begin{aligned} c_{y_i} &= y - \text{floor}(y) \\ c_{y_{i+1}} &= 1 - c_{y_i} \end{aligned} \quad (4.3)$$

Result is then presented as a list of paired 3-tuples:

$$[(x_i, y_i, c_{y_i}), (x_i, y_i + 1, c_{y_{i+1}}), \dots] \quad (4.4)$$

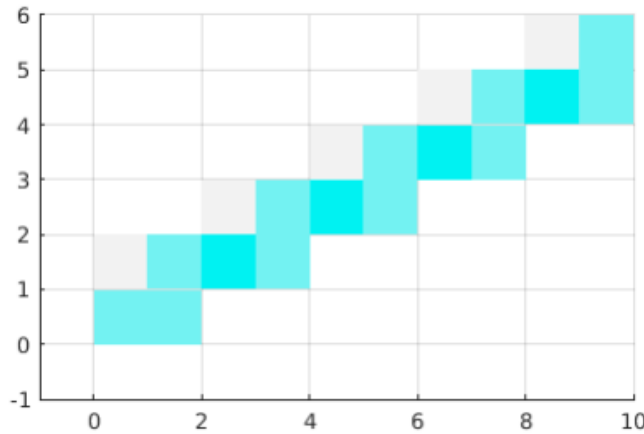


Figure 4.5 Line fitting using Xiaolin Wu algorithm.
Color intensity denotes weight of each pixel, which shows how close to the pixel position the original line lied

Both algorithms were originally implemented manually, but later both were

swapped for 3rd party libraries [32, 33] to increase performance. Libraries were adapted to return results with similar structure. In particular, Bresenham algorithm implementation library now returns 3-tuples formed in the same way as Xiaolin Wu method, with weight coefficient c equal to 1 for all cells.

Using different ray tracing algorithms yields different results for similar images. It will be shown further in part 5.1.1. In general, Wu algorithm is much more stable for higher dimensions than the Bresenham algorithm.

4.3 Plane tracing

In 3D case, FFT translates a line in the spatial domain into a plane in the frequency domain. Thus, 2D ray tracing will become plane tracing in 3D.

The first algorithm implemented was based on Bresenham method. Iterating over two out of three coordinate axes, it calculates values of the third coordinate using plane equation:

$$x = -\frac{b * (y - y_0) + c * (z - z_0)}{a} + x_0 \quad (4.5)$$

Then it chooses the pixel with the center nearest to the actual coordinate value. Similar equations should be solved for both y and z -axes. Otherwise, there will be cases when the angle is too sharp and a single x -axis step correspond to multiple steps on another axis.

The currently used algorithm is based on MATLAB *interp3* function. It constructs a horizontal plane and rotates it in polar coordinates, interpolating values or taking the nearest value depending on which mode is used. Bresenham algorithm is can be roughly approximated with the nearest neighbor method, and Xiaolin Wu method can be represented as linear interpolation. This method substitution is not exact, but in 3D case interpolation shows more stable results than the original cell-based method described in [8]. It also works faster than the manual cell selection as it relies on MATLAB's built-in functions.

4.4 Performance considerations

The initial implementation attempt, which closely followed mathematical reasoning, was made without considering performance as its major goal. Most of the operations were done as *for*-loops without any vectorization. On the quad-core Phenom II x4 925 (2.8 GHz) based machine with 6GB RAM, the algorithm took around 2 minutes to process a 128 x 128 2D images (see section 5.1), and it more than 6 hours were needed to take on the 351 x 351 x 351 3D image.

The first plane construction algorithm described in Section 4.3 has complexity that can be roughly estimated as $3 * N^2$. It takes 3 plane equation solutions to

project equation-defined plane onto discrete space without losing any intermediate point. Otherwise, there will be cases when the angle is too sharp and a single x -axis step correspond to multiple steps on another axis.

Symmetry can be used to reduce the total number of iterations needed to process the whole image. Because we know that the grid has regular step, and the plane passes through the center of the grid, we can at least reduce the number of iterations twice if we flip one of the axes (see figure 4.6).

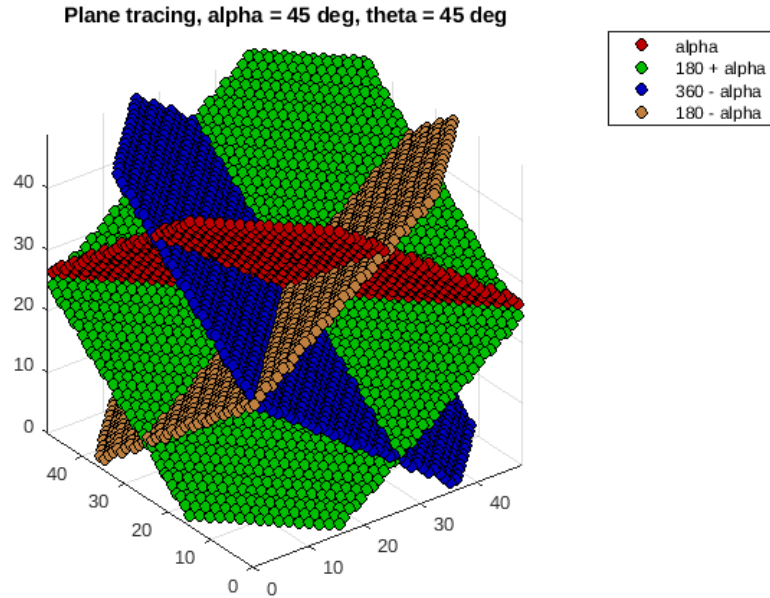


Figure 4.6 Plane tracing performance improvement.
The performance can be improved by constructing four planes at a time

For example, we ‘scan’ our 3D FFT image by fitting a plane rotating around vertical z -axis with constant angle for each major step. Minor steps only change plane angles for x and y -axes. By flipping x and y -axes will give us $360 - \alpha$ and $180 - \alpha$ values. Following the same logic, flipping z -axis should give us $-\theta$ angle. But using the FFT symmetry property, we can say that

$$\mathcal{F}(-\theta, \alpha) = \mathcal{F}(\theta, 180 + \alpha) \quad (4.6)$$

Thus, we have total 360° divided into four parts, which means that plane fitting is only required in 1 out of 4 cases. In some special cases, namely $X = Y = Z$, it can be further simplified by swapping coordinate axes, adding $\pm 90^\circ$ cases and reducing the required range of plane fitting to 45° in both α and θ directions.

The improved algorithm based on the *interp3* function still has the N^2 complexity, but it only requires a single plane to be constructed, and the rotation in

applied using rotation matrix. The multiplane construction logic described above also applies, helping to reduce the computation time even more. Removing plane construction at every step and using MATLAB built-in functions allowed to decrease the processing time from original 6 hours to a couple of minutes.

4.5 Bug hunting

A lot of time was spent on bug hunting. A number of bugs were related to the definition of the center of the image. For example, multiplane construction logic described in section 4.4 assumes that all four planes intersect at the image center. But the image center is not always located exactly at $size(I)/2$ coordinates.

In computer graphics, all images are discrete with the minimal step of one pixel. It means that image coordinates are all integer, and for the image with odd size in some direction, the center of the image coordinate in this direction will be rounded.

Yet another obstacle to determine the center coordinate is the *fftshift* function call, which shifts zero-frequency component to center of the image spectrum. What this function does internally is splitting the image in four quadrants and rearranging them, swapping first quadrant with fourth, and second quadrant with third.

MATLAB indexing brings even more confusion to this matter. For example, let us consider a 3D volumetric image with the side length of 180 pixels. The first pixel, which hold the zero-frequency component value, is located at the coordinate (1, 1, 1). After the shift, it will be moved to the coordinate (91, 91, 91), not (90, 90, 90), as we need to keep in mind MATLAB indexing. If the image will have the side length of 181 pixel, will be located at the coordinate (91, 91, 91) as well. The reason is that for odd-sized images MATLAB *fftshift* function rounds the first quadrant down.

The best way to determine where the new center of the image will be is to strictly follow the *fftshift* function logic. Divide the volumetric image into four starting from the first quadrant, making the first quadrant the smallest one. Rearrange the quadrants swapping the first quadrant with the fourth. Determine what will be the location of the top-left corner of the first quadrant after the shift.

Failing to determine the center of the image correctly results in asymmetric artifacts on symmetric images, such as spheres and vertical lines.

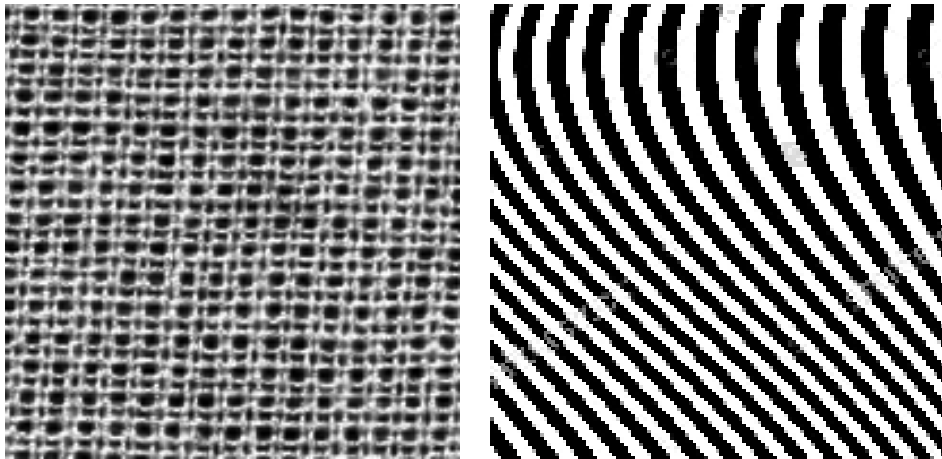
5 Results and discussion

The algorithm was tested on both 2D and 3D images, with both test and real medical images. Test images have 128 x 128 pixel resolution for 2D, and 50 x 50 x 50 for 3D, and contain differently oriented lines. For real image tests, 351 x 351 x 351 3D optic nerve MicroCT image is used. 2D tests use one of the slices of this image.

5.1 2D images

Sample 2D images were taken from [8] (image 5.1(a)), as well as from internet image stock libraries, such as www.shutterstock.com (image 5.1(b)). A real 2D image is one of the slices of a MicroCT image used as a real 3D image example.

5.1.1 Sample images



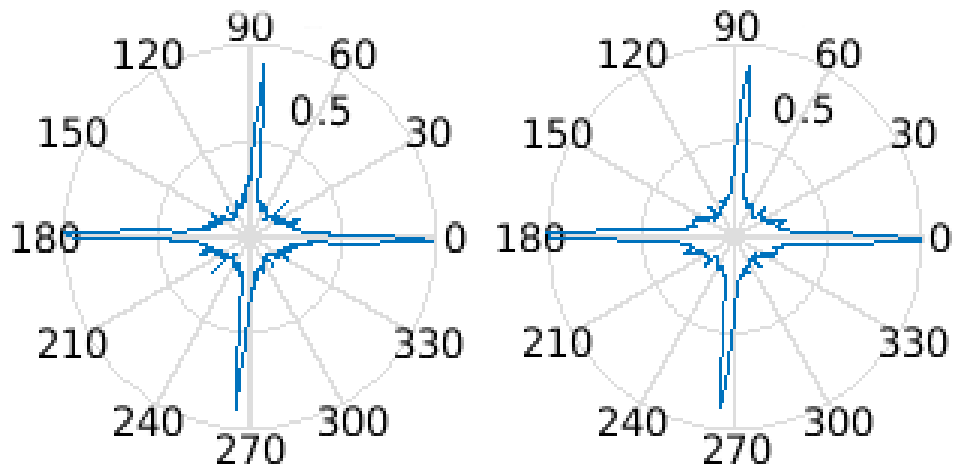
(a) Sample image 1. Source: [8]

(b) Sample image 2. Source: Shutterstock

Figure 5.1 2D sample image: originals

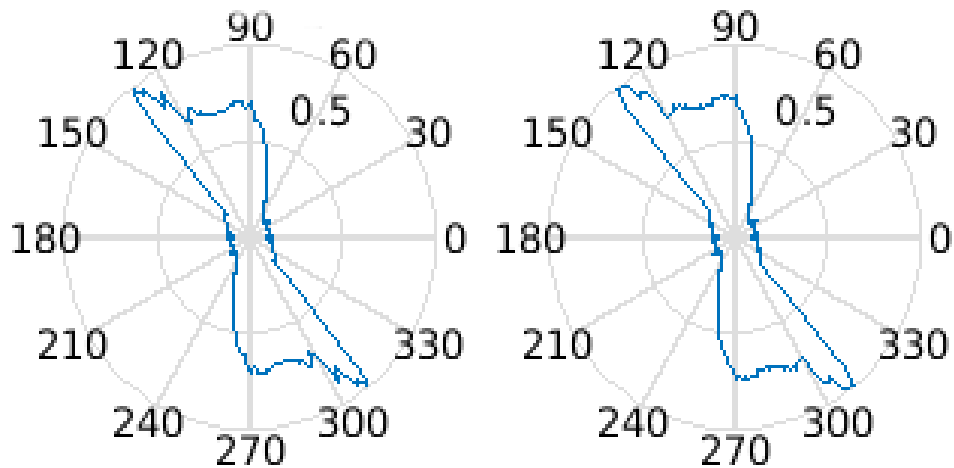
Images used as samples are shown on figure 5.1. For each image, two algorithms were tested, one using Bresenham method, and another using Xiaolin Wu method. Resulting roses of directions are shown on figure 5.2.

It can be seen from the figure 5.2 that the ray tracing method does not always provide relevant difference. On figure 5.2(b) the noise near the center has a bit lower noise magnitude and variance than on figure 5.2(a), 0.08 against 0.1 and $4e^{-4}$ against $5e^{-4}$ respectively. But the difference is still neglectable. For lower resolutions this noise dumping effect is usually more visible.



(a) Sample image 1 rose, Bresenham algorithm

(b) Sample image 1 rose, Wu algorithm



(c) Sample image 2 rose, Bresenham algorithm

(d) Sample image 2 rose, Wu algorithm

Figure 5.2 2D sample image: processing result

The highest difference is obtained by the DC removal modifier. figure 5.3 shows the result of 5.1(a) processing without DC removal enabled. In comparison with figure 5.2(a), it has a wide circle in the center of the rose. This circle is in fact a DC value, which heavily saturates the directions diagram. In case of less sharp images, DC can saturate the entire rose of directions, making any kind of directionality invisible.

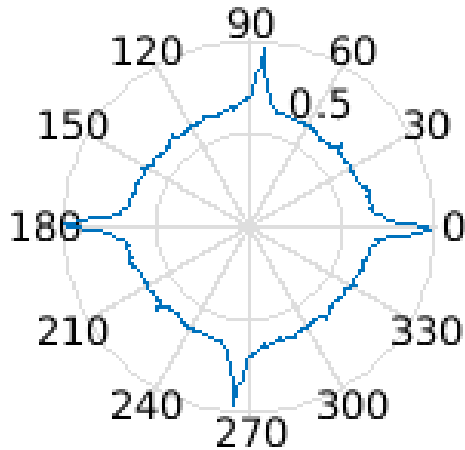


Figure 5.3 2D sample image 1: the processing result with disabled DC removal

5.1.2 Real images

For the real image case, one of the slices of optic nerve MicroCT image was used. This slice had 351 x 351 pixel resolution, is a bit noisy, and had distinguishable, yet blurry, edges. From the slice, a single directional feature was extracted for the evaluation using the tool's ROI feature.

figure 5.4 shows the program main window with image loaded and initial settings being set. For all iterations ROI was kept the same, an 150 x 130 pixel rectangle.

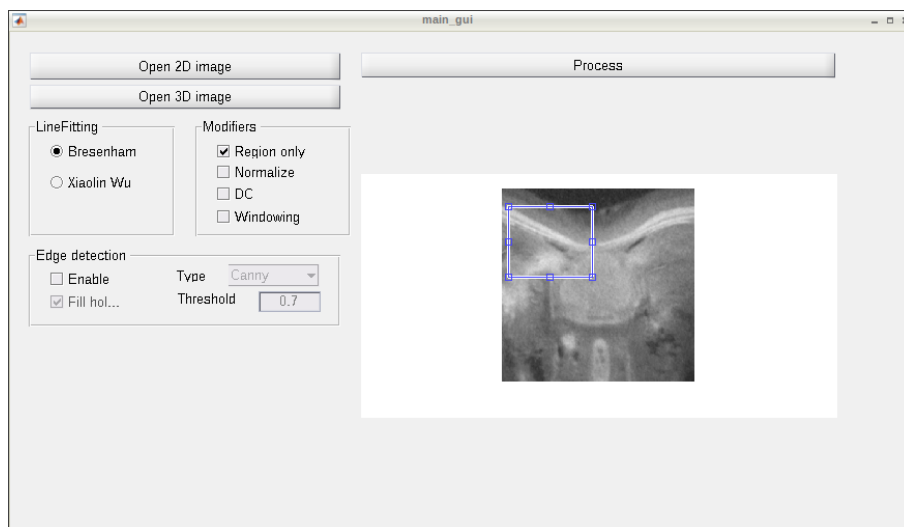


Figure 5.4 2D real image: settings

Processing result with these settings is shown on figure 5.5. Without any additional enhancements, we can still see a small 30° peak, which is almost completely saturated by the huge horizontal line. From the image FFT we can easily guess that this horizontal line is caused by the lack of periodicity.

To remove the cross-shaped artifact caused by image non-periodicity (see Section

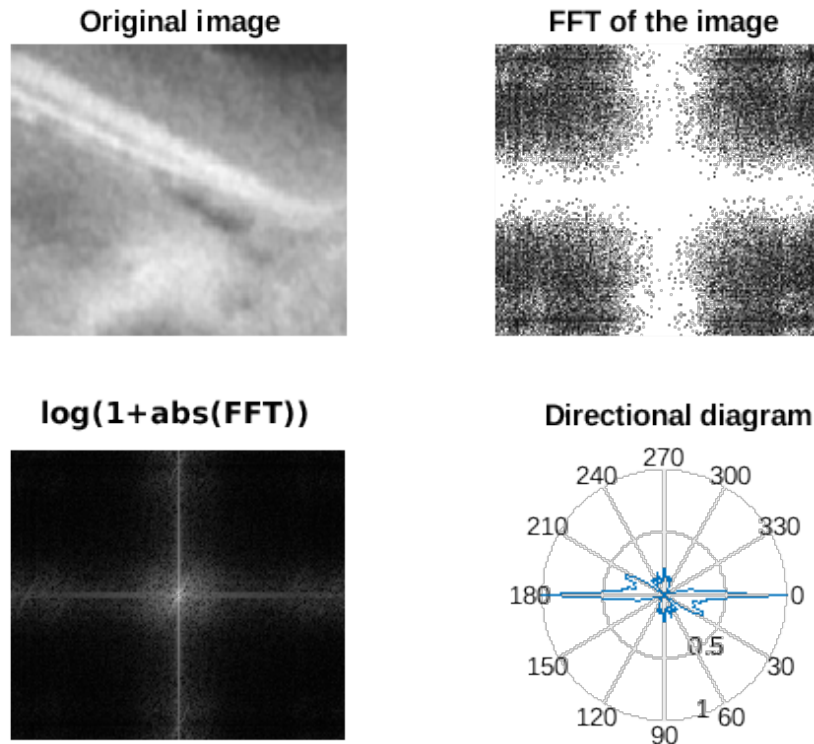


Figure 5.5 2D real image with no enhancements applied. Original (top-left), magnitude of the FFT (top-right), log-scaled magnitude of the FFT (bottom-left), and the processing result (bottom-right)

3.3) we can apply windowing. In addition, we can normalize image to make edges sharper, and remove the DC component. figure 5.6 shows the processing result with ‘Normalize’, ‘DC removal’, and ‘Windowing’ options enabled. The cross on the FFT image is dimmer than before, and the horizontal line no longer saturates the 30° spike, which is most likely caused by the hard white stripe on the image. But it is still present, and its power, though much lower, is still on par with the real value. In addition, in the rose of directions we can see a vertical spike, which is of the same nature as the horizontal one. Most likely, the cross can be dimmed even more by applying different window type and/or changing its windowing threshold. It also worth mentioning that having the same image in better resolution might significantly improve the result.

We tried one more technique which yields good results in most cases, at least when a good threshold value is found and set. figure 5.7 shows the same image processing result with edge detector and hole filling enabled. In this example, Canny edge detector is used with its threshold set to 0.7.

The result is one very sharp line tilted at approximately 30°, with almost zero additional artifacts. Binary images is the domain where FFT-based method has its

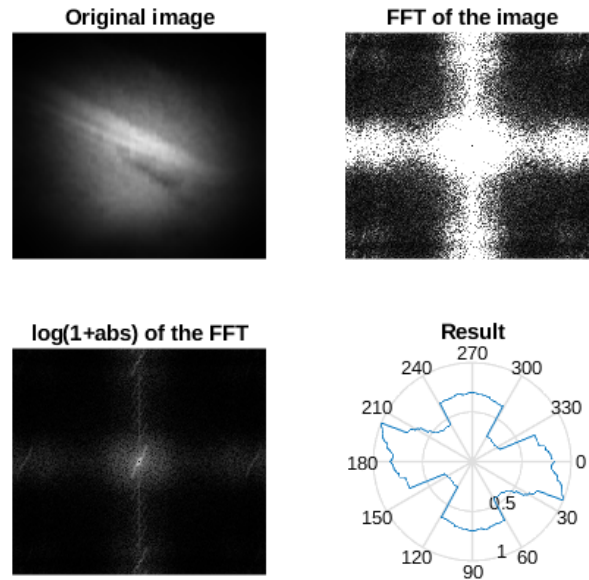


Figure 5.6 2D real image with windowing, DC removal, and normalization applied. Original (top-left), magnitude of the FFT (top-right), log-scaled magnitude of the FFT (bottom-left), and the processing result (bottom-right)

peak efficiency.

The drawback is that a lot of data is lost when edge detector is applied, especially some blurry edges which we would have naturally thought as being directional ones.

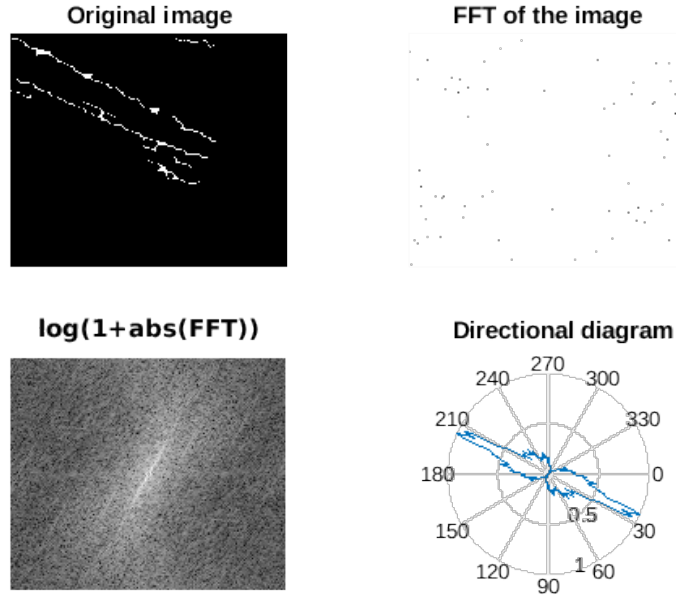


Figure 5.7 2D real image with edge detector applied. Original (top-left), magnitude of the FFT (almost uniform, top-right), log-scaled magnitude of the FFT (bottom-left), and the processing result (bottom-right)

5.2 3D images

Sample 3D images were made either from sample 2D images shown in section 5.1.1, or from geometrical primitives, such as lines and spheres. For real image tests, optic nerve MicroCT image was used.

5.2.1 Test 3D images

While switching to the 3D mode, the algorithm was first checked for correctness. For this purpose the sphere image was chosen, as it can be assumed that sphere should have no directionality at all (figure 5.8). One concern was whether or not to normalize the resulting rose of direction, as it makes the result visually sharper. The impact of the resulting diagram normalization is shown on the bottom row pane of figure 5.8. The bottom left pane shows the non-normalized result, which roughly resembles a sphere. The bottom right pane is the normalized result, which resembles a cross-shaped artifact mentioned in section 3.3. Based on this result the decision was made not to use the result normalization, as it might create significant artifacts.

Another test image was chosen to observe how the resolution affects the result. Two skew lines were used for this test, one aligned strictly with the grid and one without such strict alignment. The image itself and its processing result are shown

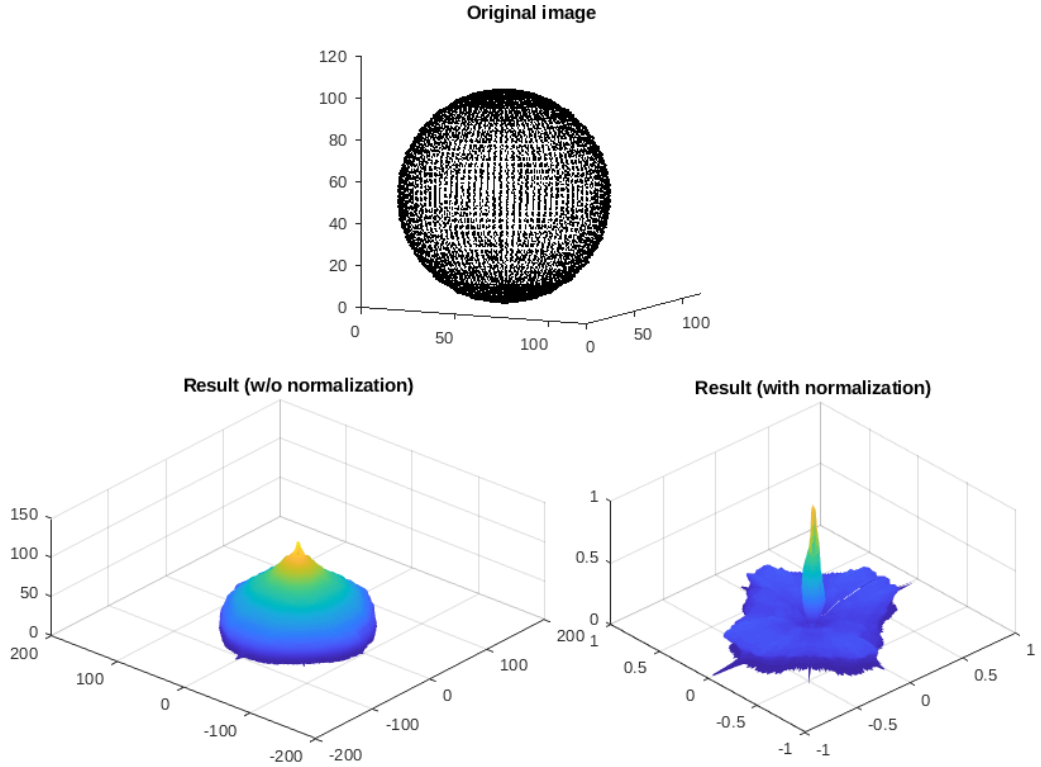


Figure 5.8 3D sample image with a sphere: original (top) and two results without normalization (bottom-left) and with normalization (bottom-right). This result shows that the result normalization can produce significant artifacts and therefore should be avoided.

on figure 5.9. The original image, shown on the top pane, is a 50x50x50 pixel 3D matrix. The resolution is quite low, and we can see a number of visible distortions on the black line. Its processing result is shown on the bottom pane. The vertical spike produced from the misaligned line is much wider than the horizontal one produced from the aligned line. This width difference is caused by the image resolution and is a direct consequence of the distortions seen on the original image. Similar to the 2D case, the higher the resolution, the sharper the result will be.

Next test image is based on the sine wave intersection. 2D base image and its processing result are shown on figure 5.10. Two sine waves, one vertical and one horizontal, intersect on the plane. Their interpolated values are encoded as pixel intensity values. The formula which made this image was

$$f(x, y) = (\sin(x) + 1)(\sin(y) + 1)/4 \quad (5.1)$$

The Fourier transform for this image returns sine wave frequency components for each direction.

If we will extrude this image into the third dimension, we will have a structure resembling a capillary system, with multiple vessels going in the same general di-

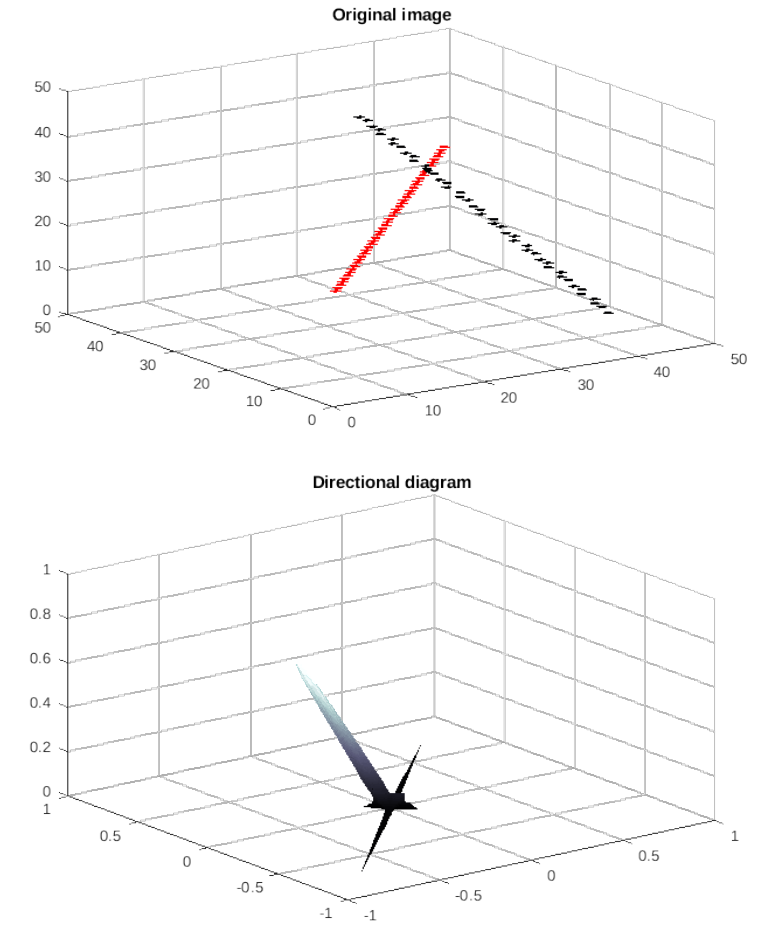


Figure 5.9 3D sample image with skew lines: original (top) and processed (bottom)
 The general direction of each line is estimated correctly,
 the width of the lines is different due to the resolution
 artifacts, which can be seen on the original image as well

rection. The 3D image and its processing result are shown on figure 5.11. While the general direction in which the base was extruded is clearly visible, the original planar frequency components from the base image are still visible as well.

The rest of the test 3D images were intended to repeat the test 2D images from sections 4.1 and 5.1.1. Therefore, the three test 2D images were taken as basis (see figure 5.12).

The 3D version of the image 5.12(a), shown on figure 5.13 together with its processing result. The figure itself is quite close to the previous one (figure 5.9). The original 2D image was transformed to 3D by simply rotating it in the third dimension. The central line has a slope of 45° against all three axes, making it a perfect diagonal of the cube surrounding the figure. The processing result, shown on the bottom pane of the image, shows correct directional pattern, similar to the one seen previously on figure 4.2. It also has a small cross-shape artifact in the basis.

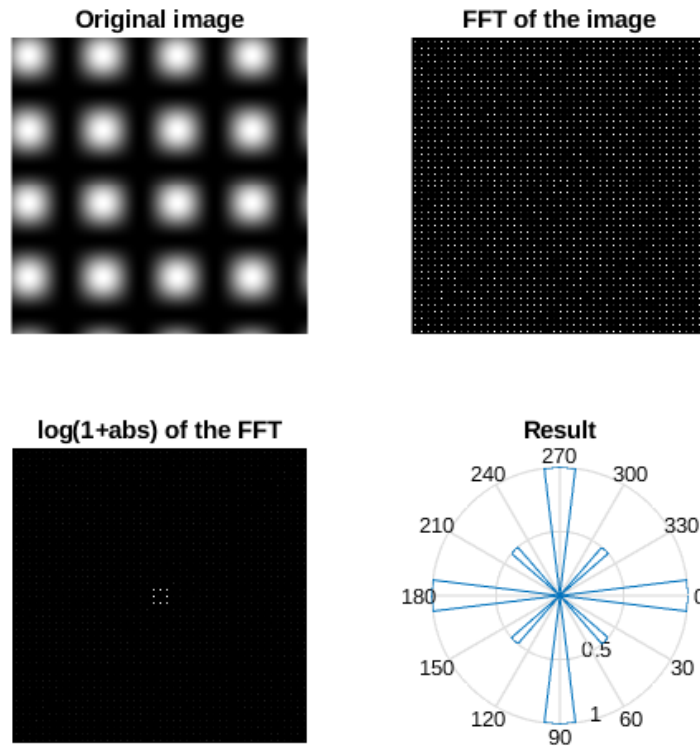
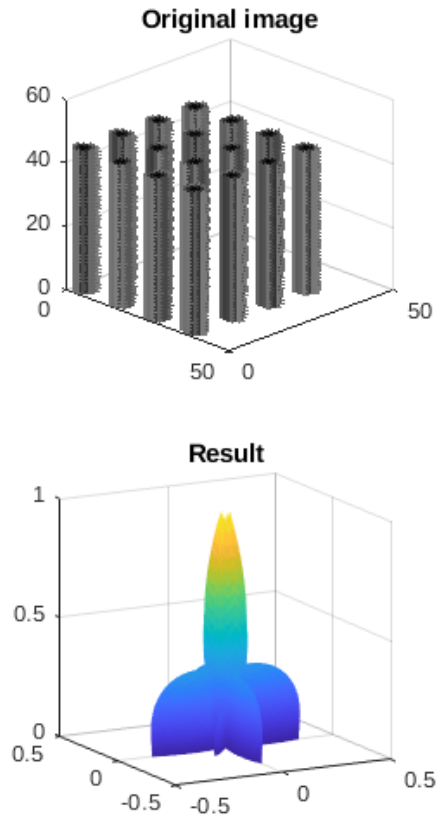


Figure 5.10 Base for the sine wave intersection image:
 original (top-left) and processed (bottom-right)
 The result returns sine wave frequency components
 in each direction.

The 3D image 5.14, based on the image 5.12(b) brings more complexity and represents the "wavy walls". Test image 5.1(b), used previously in section 5.1.1, is a part of this image. To make a 3D image, the original 2D image was extended along the third dimension. Black parts of the image were considered solid, while white parts of the image were considered to have no material.

The result, shown on the right pane of the figure 5.14, is dominated by the vertical direction. But if we will check the horizontal cross-section of this spike (figure 5.15), we will see that it resembles the 2D version of this image (figure 5.2(c)). The result being dominated by a vertical direction shows that the algorithm might not perform well on the images containing a number of small features in one direction and almost uniform another. It might be beneficial to use square root or some other weighted function instead of a simple sum when calculating the result.

The last 3D sample image is based on the figure 5.12(c). It represents a 3D rectangular grid. The image is shown together with its processing result on figure 5.16. The processing result clearly shows the main directions of this structure. This example encourages to apply the algorithm to 3D scaffold structures' assessment.



*Figure 5.11 3D sample image with sine wave intersection image: original (top) and processed (bottom)
The result returns the general direction plus sine wave frequency components from the base image.*

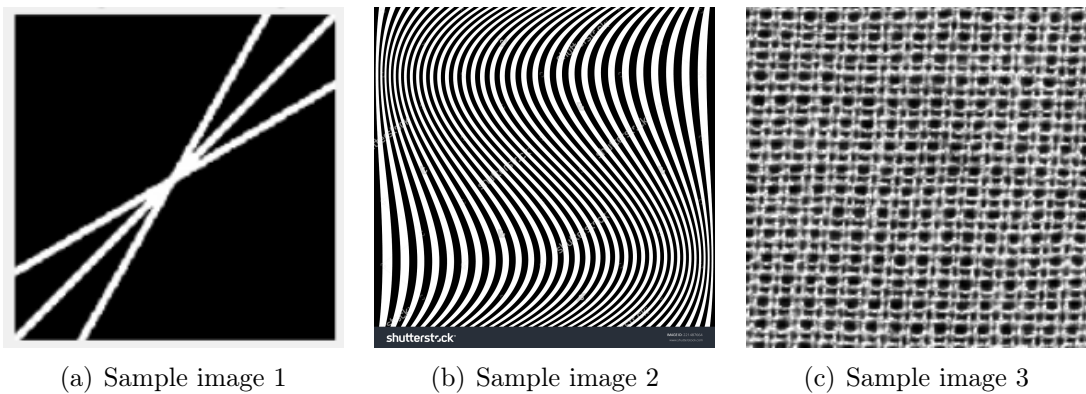


Figure 5.12 2D sample images which were used as a base for 3D sample images

Some artifacts around the center of the image are most likely produced by both DC value and imperfect symmetry of the image.

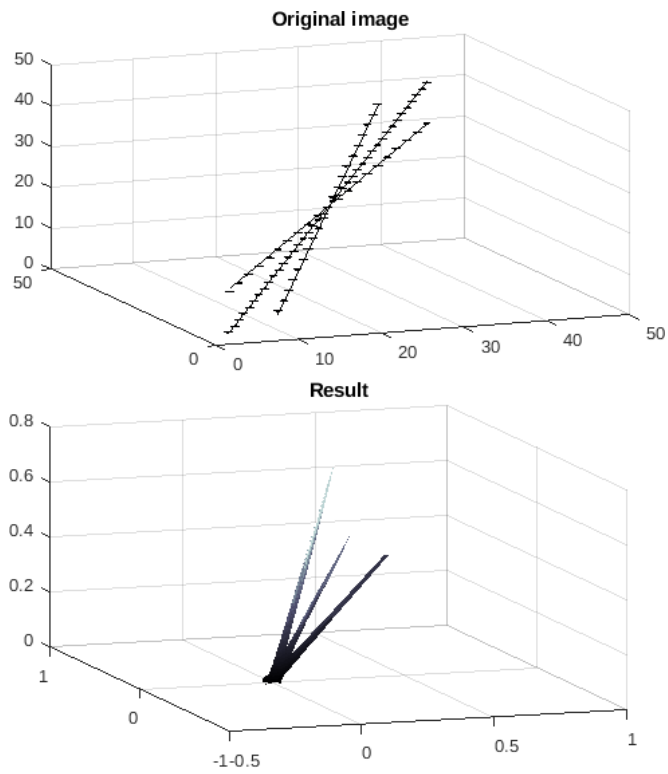


Figure 5.13 3D sample image made from the image 5.12(a) by applying rotation around the vertical axis: original (top) and processed (bottom)

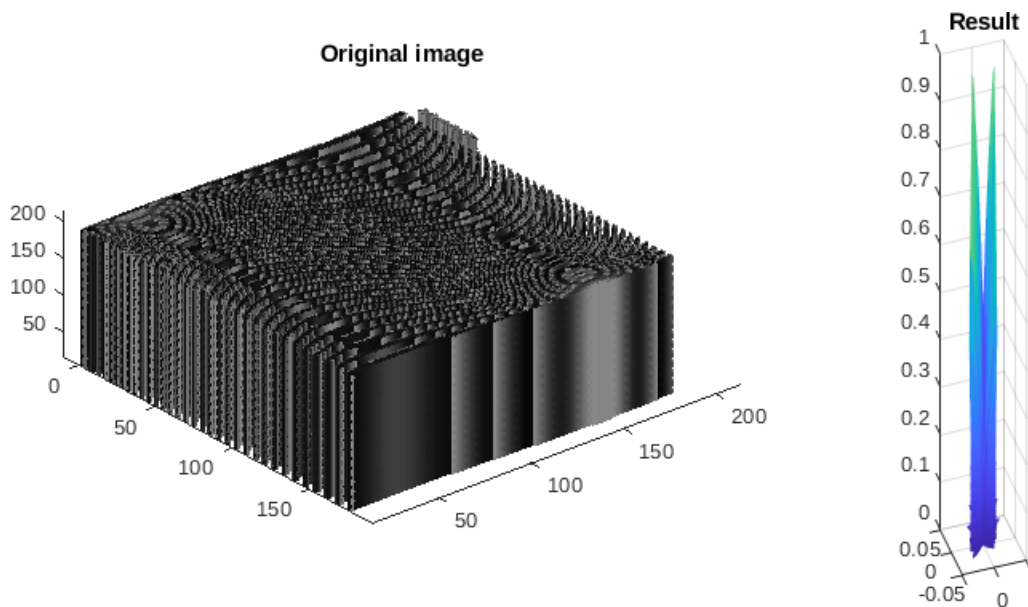


Figure 5.14 3D sample image made from the image 5.12(b) by extending it along the vertical axis: original (left) and processed (right)

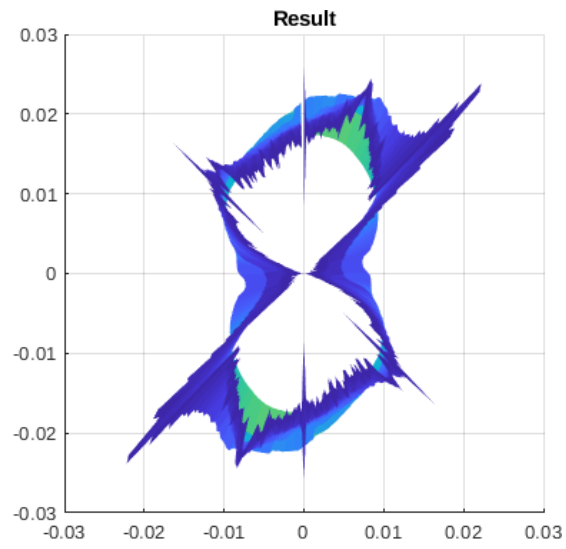


Figure 5.15 Cross-section of the right pane of the image 5.14

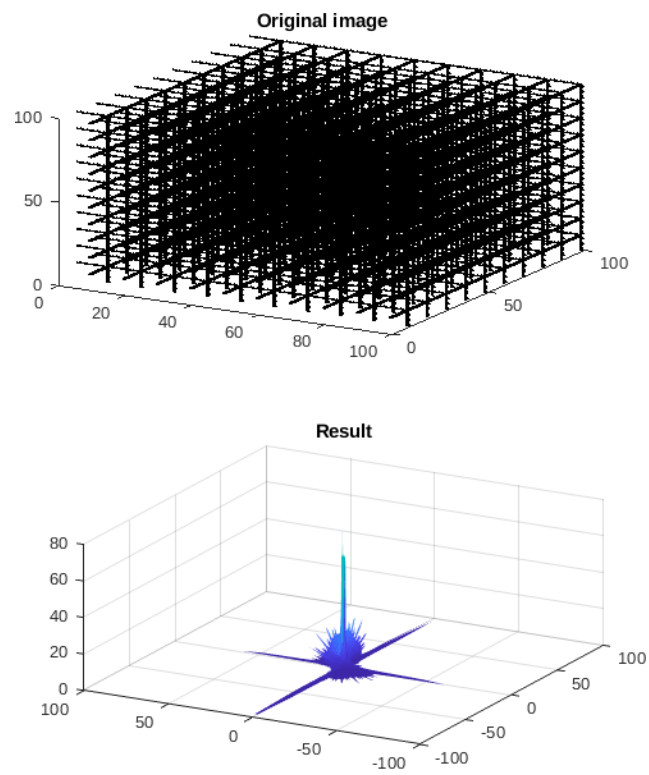


Figure 5.16 3D sample image based on the image 5.12(c)

5.2.2 Medical images

The optic nerve MicroCT image was used for the real 3D image testing. This image is a quite large 351 x 351 x 351 3D structure. A slice of it was used previously in 5.1.2 to evaluate the 2D processing mode.

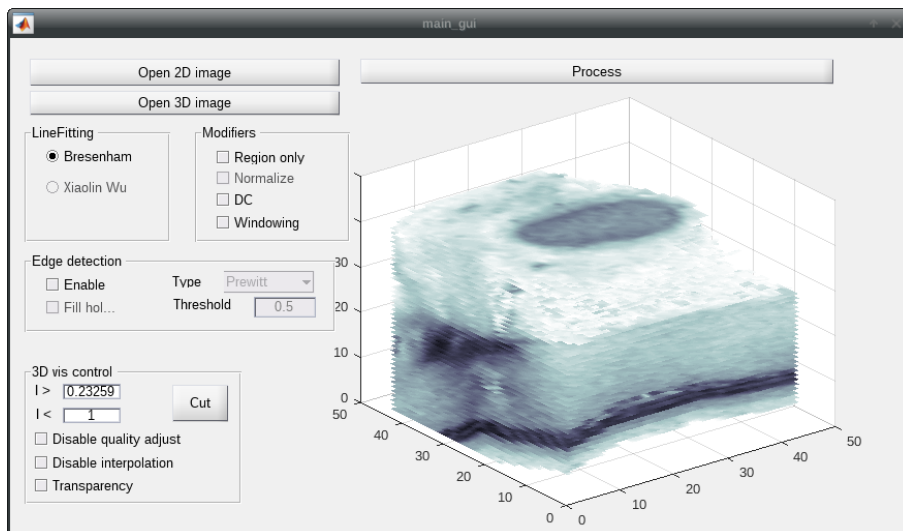


Figure 5.17 3D real image: settings

The preliminary interface options choice is shown in figure 5.17. The previous tests with 2D images suggest setting both DC and windowing on from the beginning. In addition, showing quite a large 3D structure adds additional strain in terms of visualization, so both visual quality reduction and image interpolation are used here to save resources.

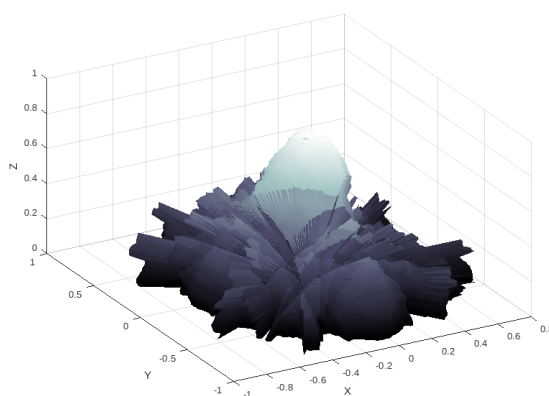


Figure 5.18 3D real image: processing result

The processing result is shown on figure 5.18. The result is not very clear. It shows some major direction, most likely belonging to the hard white stripes seen

before on the 2D slice (see figure 5.4), but the rest is almost indistinguishable. Even if there is some small internal spike, it might be saturated by much higher amplitude of the white stripes pair. Similar to the 2D case, the algorithm might yield better results if the edge detector is used.

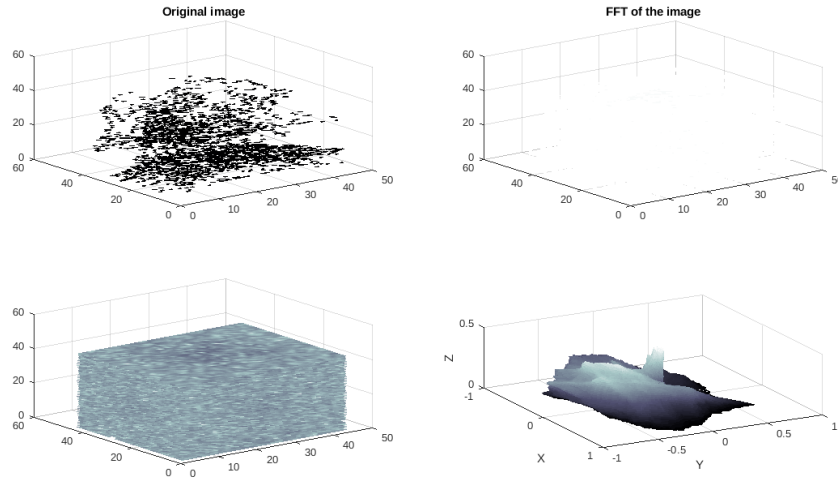


Figure 5.19 3D real image with with edge detector applied
*The general direction now can be guessed, but it is still not yet clear.
 Original (top-left), magnitude of the FFT (almost uniform, top-right),
 log-scaled magnitude of the FFT (bottom-left),
 and the processing result (bottom-right)*

The result of the running the algorithm with edge detection is shown on figure 5.19. We can see a stem in the middle, and a slight tilt to the side. The tilt is most likely produced by the hard edge at the bottom of the original image, and the stem corresponds to the number of edges in the center. Yet, it is still not clear which feature produced which line or inclination.

5.3 Discussion

The acquired results show that the algorithm perform differently on different tasks. In both 2D and 3D cases, the algorithm performs best when the image contains a structure made of distinct lines forming a scaffold (figure 5.2 for 2D, figures 5.9, 5.13, and 5.16 for 3D). Images with solid objects, continuous in a single direction, might produce a heavily saturated result, with a single direction dominating others (figure 5.14).

Soft tissue images without any sharp edges (figure 5.4 for 2D, figure 5.17 for 3D) might require a lot of additional processing including edge detector application to achieve best possible results (figure 5.7). Even if the edge detector is applied, the result might be very difficult to interpret, showing only the general direction

without any distinguishable features (figure 5.19).

To address this problem, we can try extracting some features, similar to the way it was done in [17]. For instance, we can take width-to-depth ratio for the result bounding box. Or we can try to take the plane in which the longest possible axis of the result lies, fit an ellipse around the result, and then calculate the maximum outlier distance for the rest of the result slice. There are many different ways to do feature extraction, and the trial-and-error approach to choose between those is the most commonly used one.

Using feature extraction might also lead to two major obstacles. One is the number of training and testing images required to have confidence in the extracted feature. In general this number can be quite high. Features are usually problem-specific, so for our case it is required to process multiple images for the same medical case in order to train the model. Alternatively, we can use our whole result as one big feature, and then try to combine it with other features extracted from patients' personal records. This way requires high-level expertise, but it will be the most proper way in terms of machine learning.

The second obstacle, also discussed in [17], is that the resulting feature is not always readable by humans. Once again, it is possible to identify some peaks and stems corresponding to pathological structures or their orientation, but such interpretation will not be obvious. This observation once again pushes us towards machine learning, where human-readability of the feature is not required.

Performance-wise, the initial implementation was lacking, as it can be seen from Section 4.4. After adding vectorization, switching to built-in function where it is possible, and reducing the amount of loops, the algorithm performs much faster. 2D image processing takes less than 5 seconds, and 3D MRI takes less than 20 minutes. Plane tracing in 3D is still the slowest part of the algorithm, and improving it will give a huge speed boost to the whole computation process. Multiplane construction method described in the same section allowed to improve performance even further.

Another unexpectedly compute-intensive part was the edge detection. 3D mode edge detector relies on MATLAB *edge3()* function, which appeared only in R2017 release, which is the latest one to the time this thesis is written. Hand-written 3D edge detectors, both in-house and 3rd party, were slow and memory consuming.

The only main performance-related limitation of the current implementation is the visualization of the 3D image. Layer-by-layer visualization with transparency enabled consumes quite a lot of GPU resources, and viewing the computation result in the MATLAB embedded viewer might negatively affect the end-user experience. One way proposed is to export MATLAB 3D figure as 3D XHTML [34]. Modern web browsers include highly tuned JavaScript processing and rendering engine, which outperforms MATLAB viewer even on very slow machines. Alternatively,

it is possible to create a 3D embedded PDF object [35], though embedding programmable entities inside PDF documents is considered a bad practice due to both performance and security issues.

6 Conclusions

This thesis introduces one way to solve the problem of defining medical image directionality. The reference method described in [8] was analyzed and extended into three dimensions. This extension allows additional directionality assessment to routine 3D medical CT images analysis. Another potential application could be 3D scaffold's structure assessment. The extended method was tested on both sample images and a MicroCT 3D image of the optic nerve.

It was shown that image preprocessing gives a significant boost to the overall result quality of this method. It works best on binary scaffold-like images, which can be obtained using an edge detector. Typical example of such image might be a preprocessed image of a vascular system of a human body.

Still, the result of this method might be hard to interpret if the original image contains complex structures without distinct sharp edges. One solution proposed is to use machine learning techniques in addition to the result produced by this method, which can be used as one of the features for classification.

References

- [1] R. Rangayyan. *Biomedical Image Analysis*. Biomedical Engineering. CRC Press, 2004. ISBN: 9780203492543. URL: https://books.google.ru/books?id=o%5C_8jEGNpEuwC.
- [2] C. Leung et al. “New Technologies for Glaucoma Imaging”. In: 2011 (Sept. 2011), p. 608975.
- [3] B. L. Schuster. *How Glaucoma Affects the Optic Nerve*. 2017. URL: <https://www.glaucoma.org/glaucoma/the-optic-nerve-questions-and-answers-from-dr-bradley-schuster.php> (visited on 03/20/2017).
- [4] E. M. Hoffmann et al. “Optic disk size and glaucoma”. In: *Surv Ophthalmol* 52.1 (2007), pp. 32–49.
- [5] Z. Starosolski et al. “Ultra High-Resolution In vivo Computed Tomography Imaging of Mouse Cerebrovasculature Using a Long Circulating Blood Pool Contrast Agent”. In: 5 (May 2015), p. 10178.
- [6] C. Kam, D. W. Chee, and W. C. Peh. “Magnetic Resonance Imaging of Cruciate Ligament Injuries of the Knee”. In: *Canadian Association of Radiologists Journal* 61.2 (2010), pp. 80–89. ISSN: 0846-5371. DOI: <https://doi.org/10.1016/j.carj.2009.11.003>. URL: <http://www.sciencedirect.com/science/article/pii/S0846537109002241>.
- [7] M. Tunák and A. Linka. “PLANAR ANISOTROPY OF FIBRE SYSTEMS BY USING MATLAB IMAGE PROCESSING TOOLBOX”. In: (Feb. 2018).
- [8] M. Tunák and A. Linka. “Analysis of planar anisotropy of fibre systems by using 2D Fourier transform”. In: *Fibres & Textiles In Eastern Europe* 15.5-6 (2007).
- [9] X.-J. He et al. “A New Feature of Uniformity of Image Texture Directions Coinciding with the Human Eyes Perception”. In: *Fuzzy Systems and Knowledge Discovery*. Ed. by L. Wang and Y. Jin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 727–730. ISBN: 978-3-540-31828-6.
- [10] M. M. Islam, D. Zhang, and G. Lu. “A geometric method to compute directionality features for texture images”. In: *2008 IEEE International Conference on Multimedia and Expo*. June 2008, pp. 1521–1524. DOI: [10.1109/ICME.2008.4607736](https://doi.org/10.1109/ICME.2008.4607736).
- [11] A. R. Rivera, J. R. Castillo, and O. Chae. “Local directional texture pattern image descriptor”. In: *Pattern Recognition Letters* 51 (2015), pp. 94–100.

- [12] Wikipedia. *Kirsch operator* — *Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/wiki/Kirsch_operator. [Online; accessed 15-February-2018]. 2018.
- [13] S. Livens et al. “Wavelets for texture analysis, an overview”. In: Aug. 1997, 581–585 vol.2. ISBN: 0-85296-692-X. DOI: 10.1049/cp:19970958.
- [14] A. Sovic and D. Sersic. “Signal decomposition methods for reducing drawbacks of the DWT”. In: *Engineering Review* 32 (Jan. 2012), pp. 70–77.
- [15] I. W. Selesnick, R. G. Baraniuk, and N. C. Kingsbury. “The dual-tree complex wavelet transform”. In: *IEEE Signal Processing Magazine* 22.6 (Nov. 2005), pp. 123–151. ISSN: 1053-5888. DOI: 10.1109/MSP.2005.1550194.
- [16] D. B. Aydogan et al. “2D texture based classification, segmentation and 3D orientation estimation of tissues using DT-CWT feature extraction methods”. In: *Data & Knowledge Engineering* 68 (Dec. 2009), pp. 1383–1397. DOI: 10.1016/j.datak.2009.07.009.
- [17] V. A. Kovalev, M. Petrou, and Y. S. Bondar. “Texture anisotropy in 3-D images”. In: *IEEE Transactions on Image Processing* 8.3 (Mar. 1999), pp. 346–360. ISSN: 1057-7149. DOI: 10.1109/83.748890.
- [18] M. S. Longuet-Higgins. “The statistical analysis of a random, moving surface”. In: *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 249.966 (1957), pp. 321–387. ISSN: 0080-4614. DOI: 10.1098/rsta.1957.0002. eprint: <http://rsta.royalsocietypublishing.org/content/249/966/321.full.pdf>. URL: <http://rsta.royalsocietypublishing.org/content/249/966/321>.
- [19] D. Cooper and J. Graham. “Estimating Motion in Noisy, Textured Images: Optical Flow in Medical Ultrasound”. In: *Proceedings of the British Machine Vision Conference*. doi:10.5244/C.10.9. BMVA Press, 1996, pp. 9.1–9.10. ISBN: 0-952189-83-6.
- [20] D. Heeger. “Model for the extraction of image flow.” English (US). In: *Journal of the Optical Society of America A: Optics and Image Science, and Vision* 4.8 (Aug. 1987), pp. 1455–1471. ISSN: 0740-3232.
- [21] D. J. Heeger. “Optical flow using spatiotemporal filters”. In: *International Journal of Computer Vision* 1.4 (Jan. 1988), pp. 279–302. ISSN: 1573-1405. DOI: 10.1007/BF00133568. URL: <https://doi.org/10.1007/BF00133568>.
- [22] J. Sato and R. Cipolla. “Image registration using multi-scale texture moments”. In: *Proceedings of the British Machine Vision Conference*. Vol. 1. 1994, pp. 94–104.

- [23] D. Chetverikov. “TEXTURAL ANISOTROPY FEATURES FOR TEXTURE ANALYSIS.” English. In: *Proceedings - IEEE Computer Society Conference on Pattern Recognition and Image Processing*. IEEE Comput Soc Press, 1981, pp. 583–588.
- [24] D. Chetverikov and R. M. Haralick. “Texture anisotropy, symmetry, regularity: Recovering structure from interaction maps”. In: *In Proc. British Machine Vision Conference*. 1995, pp. 57–66.
- [25] W. Lo et al. “Fast Fourier Transform-Based Analysis of Second-Harmonic Generation Image in Keratoconic Cornea”. In: *Investigative Ophthalmology & Visual Science* 53.7 (2012), p. 3501. DOI: 10.1167/iovs.10-6697. eprint: /data/journals/iovs/932981/i1552-5783-53-7-3501.pdf. URL: +%20http://dx.doi.org/10.1167/iovs.10-6697.
- [26] R. Holota and S. Nemecek. “Recognition of oriented structures by 2D Fourier transform”. In: *Applied Electronics* (2002), pp. 88–92.
- [27] S. Kiranyaz. *SGN-12006 Basic Course in Image and Video Processing, Lecture notes*. 2014.
- [28] E. Lam. *Some Properties of Fourier Transform*. 2008. URL: <https://www.eee.hku.hk/~work8501/WWW2008/ho4.pdf> (visited on 03/10/2019).
- [29] R. C. Gonzalez and R. E. Woods. *Digital image processing*. Upper Saddle River, N.J.: Prentice Hall, 2008. ISBN: 9780131687288 013168728X 9780135052679 013505267X. URL: <http://www.amazon.com/Digital-Image-Processing-3rd-Edition/dp/013168728X>.
- [30] Wikipedia. *Bresenham’s line algorithm — Wikipedia, The Free Encyclopedia*. <http://en.wikipedia.org/w/index.php?title=Bresenham's%20line%20algorithm&oldid=821189785>. [Online; accessed 14-February-2018]. 2018.
- [31] X. Wu. “An Efficient Antialiasing Technique”. In: *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '91. New York, NY, USA: ACM, 1991, pp. 143–152. ISBN: 0-89791-436-8. DOI: 10.1145/122718.122734. URL: <http://doi.acm.org/10.1145/122718.122734>.
- [32] A. Wetzler. *Bresenham optimized for Matlab*. 2010. URL: <https://www.mathworks.com/matlabcentral/fileexchange/28190-bresenham-optimized-for-matlab> (visited on 03/10/2019).
- [33] S. Abeywardana. *Xiaolin Wu line algorithm*. 2013. URL: <https://github.com/sachinruk/xiaolinwu> (visited on 03/10/2019).

- [34] D.-J. Kroon. *Matlab 3D figure to 3D (X)HTML*. 2011. URL: <https://www.mathworks.com/matlabcentral/fileexchange/32207-matlab-3d-figure-to-3d-x-html> (visited on 03/10/2019).
- [35] I. Filippidis. *Export figure to 3D interactive PDF*. 2017. URL: <https://www.mathworks.com/matlabcentral/fileexchange/37640-export-figure-to-3d-interactive-pdf> (visited on 03/10/2019).