

Semester Thesis

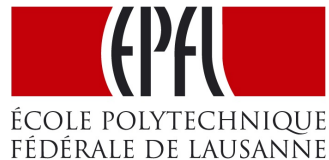
Spring 2010

Collagen Filament Detection

Student
François Curdy

Advisers
Daniel Sage
Prof. Michael Unser

Laboratoire d'Imagerie Biomédicale (LIB)
École Polytechnique Fédérale de Lausanne (EPFL)



Contents

1	Introduction	1
2	Method to detect the orientations	3
2.1	Implementation of the Gradient Square Tensor (GST)	5
2.1.1	Computation of the gradients using the cubic splines	6
2.1.2	Gradient for anisotropic volumes	7
2.1.3	Smoothing	8
2.2	Computation of the eigenvalues and eigenvectors	9
2.2.1	Computation of the eigenvalues and eigenvectors: Algebraical solution	9
2.2.2	Computation of the eigenvalues and eigenvectors: Analytical solution	10
2.3	Computation of the two angles φ and θ	11
2.4	Coherency feature	13
3	Construction of the bidimensional histogram	14
3.1	Clustering on the bidimensional histogram	15
4	Colour representation	17
5	Validation of the algorithm	19
5.1	Histograms and Color representation on validation volumes	21
5.2	Validation of the clustering algorithm	23
5.3	Effects of the input parameter: σ	24
5.4	Effect of the noise	25
6	Orientations analysis of 3-D collagen data	27
7	Conclusion	35
	References	36

1 Introduction

Collagen is a very abundant protein in the human body, which provides most of the tensional strength of the tissues. The fundamental unit of the collagen consists of three polypeptide strands twisted together into a helix: the *tropocollagen*. This subunit is approximately 280 nm long for 1.5 nm of diameter. Each strands is composed of a regular arrangement of three amino acids(aa), depending on the type of the collagen (29 different types). Glycine is the common amino acid for each type, accounting for one third of the sequence, it appears periodically each third position. The second aa is either proline or hydroxyproline. Therefore the sequence consists of $(Gly - Pro - X)_n$ or $(Gly - X - Hyp)_n$. *Tropocollagen* cohesion is assured by hydrogen bonds between these chains.[2]

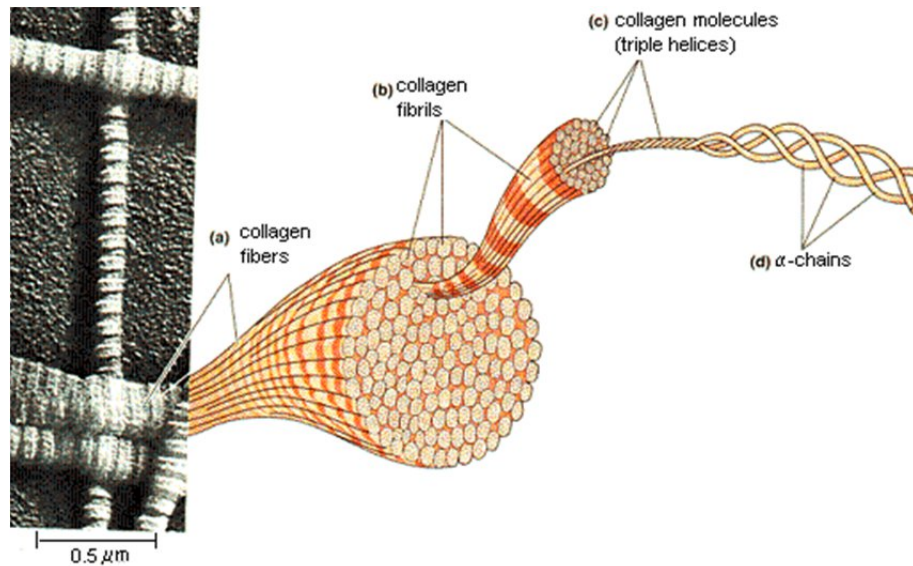


Figure 1: Collagen structure

Collagen is the main component of cartilages, ligaments, tendons, etc. It is present in blood vessels, such as arteries, to provide strength to the walls to resist against high pressures.

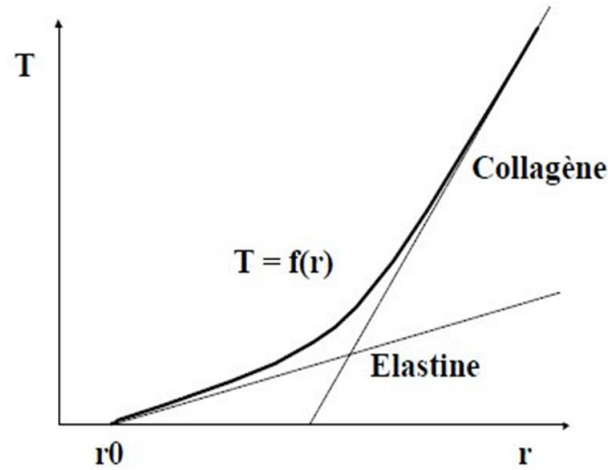


Figure 2: Diameter of the artery in function of the stress due to the blood pressure

The orientation of the filaments determines tissue mechanical properties.

The BioImaging and Optic Platform (BIOP) of the EPFL, has acquired 3D images of arterial wall sections with a confocal microscope and add a fluorescent green marker to collagen. The aim of this project is to design an image analysis tool for automatic 3D orientation analysis of these collagen fibres, to allow the creation of 3D models to simulate different arterial tissues with finite element analysis. To perform these simulations, informations about the distribution of the different orientations $\phi(\theta, \varphi)$ and the waves made by the filaments $\phi(\lambda)$ will be needed. This automatic 3D orientation will be built as an *ImageJ* plug-in.

2 Method to detect the orientations

A widely used algorithm to detect orientations in 3-D images consist of estimate the local dimensionality of the 3x3 Hessian matrix for each pixels. This means that we look at the local neighbourhood of each pixel and find the curvature. In three dimensional space, curvature is defined by a vector in a tangent direction of the curve at any point 3. Three principal curvatures and directions could be found in a three dimensional object, because the order of the matrix is 3. The maximum and minimum curvature are called the principal curvatures and their orientation are the principal directions.

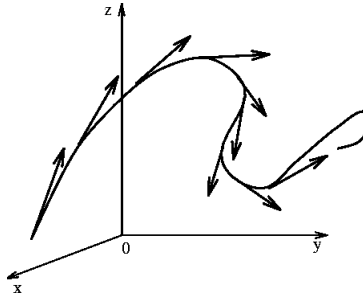


Figure 3: Tangent vectors at specific points represent the curvature of the line in a three dimensional space.

The eigenvalues of the matrix correspond to the principal curvatures and the eigenvectors to the principal directions [2][6][9][10]. Thus, the main orientation is given by the perpendicular of the eigenvector (principal direction) obtained with the lowest eigenvalue (principal curvature).

Figure 4 shows the different operations to obtain orientations and the histogram. We will first defined each steps separately and then present the results obtained with this algorithm.

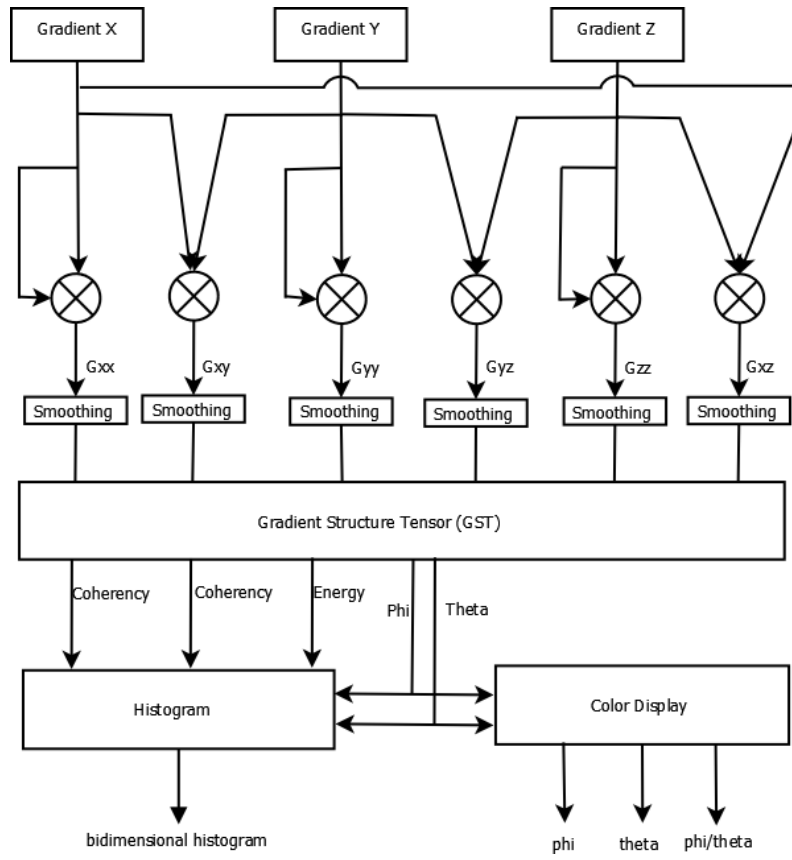


Figure 4: Computation of the GST, the angles, the color representation, and the bidiemensional histogram

2.1 Implementation of the Gradient Square Tensor (GST)

The analytical solutions for the eigenvalues and eigenvectors using the 3-D GST have been given by van Kempen et al. [9]. The structure tensor has first been introduced by Jähne [6]. Here are the principal steps that lead to this tensor:

- Squared scalar product between the gradient and the orientation $\bar{\mathbf{n}}$ is proportional to the cosine square of their angle:

$$(\nabla g^T \bar{\mathbf{n}})^2 = |\nabla g|^2 \cos^2(\angle(\nabla g, \bar{\mathbf{n}})) \quad (1)$$

- Following integral ($\mathbf{J}$) is maximized in a 3-dimensional neighbourhood:

$$\int \bar{\mathbf{n}}^T \mathbf{J} \bar{\mathbf{n}} \rightarrow \text{maximum} \quad (2)$$

$$\text{with } \mathbf{J}_{pq}(\mathbf{x}) = \int w(\mathbf{x} - \mathbf{x}') \left(\frac{\delta g(\mathbf{x}')}{\delta x'_p} \frac{\delta g(\mathbf{x}')}{\delta x'_q} \right) d^3 x' \quad (3)$$

- w is a window function of the neighbourhood of $\mathbf{x}$.
- the tensor is an adequate first-order representation of a local neighbourhood.

To create this tensor, we will first compute the three gradients in each directions (x, y, z) of the input image, using the cubic Splines method, and combine them. We will then smooth the gradients with a Gaussian smoothing.

$$GST = \begin{pmatrix} \overline{\frac{dg}{dx} \cdot \frac{dg}{dx}} & \overline{\frac{dg}{dx} \cdot \frac{dg}{dy}} & \overline{\frac{dg}{dx} \cdot \frac{dg}{dz}} \\ \overline{\frac{dg}{dy} \cdot \frac{dg}{dx}} & \overline{\frac{dg}{dy} \cdot \frac{dg}{dy}} & \overline{\frac{dg}{dy} \cdot \frac{dg}{dz}} \\ \overline{\frac{dg}{dz} \cdot \frac{dg}{dx}} & \overline{\frac{dg}{dz} \cdot \frac{dg}{dy}} & \overline{\frac{dg}{dz} \cdot \frac{dg}{dz}} \end{pmatrix} \quad (4)$$

Thus, the first step is to compute the three gradients in each direction x , y and z :

$$g_x = \frac{dg}{dx} \quad g_y = \frac{dg}{dy} \quad g_z = \frac{dg}{dz}$$

Then, multiply with each other to obtain: g_{xx} , g_{xy} , g_{yy} , g_{yz} , g_{zz} and g_{xz} respectively, (with $g_{ij} = g_i \cdot g_j$).

2.1.1 Computation of the gradients using the cubic splines

This method allows to compute the exact gradient of the continuous function $f(x, y, z)$ instead of the discrete one. An interpolation using cubic splines is used to find the functions $f(x)$, $f(y)$ and $f(z)$ [8]. Then the gradients are computed for each function separately. This is possible because the splines interpolation and the derivative operator are separable:

$$f(x) = \sum_{l \in \mathbb{Z}} c[l] \beta^3(x - l) \quad (5)$$

$$f_x(x) = \sum_{l \in \mathbb{Z}} c[l] \Delta \beta^2(x - l) \quad (6)$$

With:

$$\Delta \beta^2(x) = \beta^2(x + 0.5) - \beta^2(x - 0.5) \quad (7)$$

The $c[k]$ coefficients are computed recursively with the following filter:

$$H(z) = \frac{6}{z + 4 + z^{-1}} = \frac{6}{z^{-1}(1 + 4z^{-1} + z^{-2})} \quad (8)$$

This filter can be separate in two components, one causal, and one anti-causal:

$$H(z) = 6 \cdot \frac{1}{1 - a \cdot z^{-1}} \cdot \frac{-a}{1 - a \cdot z} \quad (9)$$

$$\begin{aligned} \Rightarrow c'[k] &= f[k] + a \cdot c[k - 1] \\ \Rightarrow c''[k] &= a \cdot (c''[k + 1] + c'[k]) \\ \Rightarrow c[k] &= 6 \cdot c''[k] \end{aligned}$$

With $a = \sqrt{3} - 2$.

Once these coefficients computed, we can insert them into the equation of $f_x(x)$:

$$f_x(x) = \sum_{k \in Z} c[k] \beta^2(x + 0.5 - k) - \sum_{k \in Z} c[k] \beta^2(x - 0.5 - k) \quad (10)$$

$$= \sum_{k \in Z} c[k] \beta^2(x + 0.5 - k) - \sum_{k \in Z} c[k + 1] \beta^2(x - 0.5 - k + 1) \quad (11)$$

$$= \sum_{k \in Z} c[k] \beta^2(x + 0.5 - k) - \sum_{k \in Z} c[k + 1] \beta^2(x + 0.5 - k) \quad (12)$$

$$= \sum_{k \in Z} (c[k] - c[k + 1]) \beta^2(x + 0.5 - k) \quad (13)$$

2.1.2 Gradient for anisotropic volumes

Collagen 3-D images are taken with a confocal microscope with an equivalent resolution in the $x - y$ plane, i.e. the increment from one pixel to another in the x and y -axis corresponds to the same quantity. This is not the case in the z -axis, where the distance between two slices of the volume is greater than in the x and y -axis. This anisotropy factor for the collagen images is approximatively 0.9:

Thus, the gradient in the z direction could not be computed the same way as the others. The function is dilated in the z directions:

$$f(x, y, z) \Rightarrow f(x, y, \alpha z)$$

Fortunately the gradients are computed separately and so only the *GradientZ()* function has to be changed.

$$f(\alpha z) = \sum_{l \in Z} c[l] \beta^3(\alpha z - l)$$

$$f_z(\alpha z) = \sum_{l \in Z} c[l] (\beta^2(\alpha(z + 0.5) - l) - \beta^2(\alpha(z - 0.5) - l))$$

The parameter t given to the *getQuadraticSpline()* function change from 0.5 for the isotropic case to: $(\text{Math.floor}(\alpha z) - z + 1)/2$.

2.1.3 Smoothing

We smooth the computed gradients with a 3-D separable isotropic Gaussian kernel (i.e. $\sigma_x = \sigma_y = \sigma_z$). The sigma is a parameter given by the user, it depends on the dimension of the elements present in the images. With big structures, one may use a quite large sigma and conversely for small structures.

$$G(\mathbf{x}) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} e^{-\frac{z^2}{2\sigma^2}} \quad (14)$$

The smoothing in the z-axis has no longer to be computed with a different kernel, because the gradientZ has already brought the function back to an isotropic case.

2.2 Computation of the eigenvalues and eigenvectors

The eigenvalues of the GST give us the necessary informations to find the angles, as explained in section 2. We will describe in this section two different ways to compute theses values and compare them in term of accuracy and computation time. The subsection 2.1 described how to implement this GST:

$$GST = \begin{pmatrix} \overline{g_{xx}} & \overline{g_{xy}} & \overline{g_{xz}} \\ \overline{g_{xy}} & \overline{g_{yy}} & \overline{g_{yz}} \\ \overline{g_{xz}} & \overline{g_{yz}} & \overline{g_{zz}} \end{pmatrix}$$

The eigenvalues of this matrix are now computed by solving the following equation:

$$\det(\mathbf{G} - \lambda \cdot \mathbf{I}) = 0$$

$$\begin{vmatrix} \overline{g_{xx}} - \lambda & \overline{g_{xy}} & \overline{g_{xz}} \\ \overline{g_{xy}} & \overline{g_{yy}} - \lambda & \overline{g_{yz}} \\ \overline{g_{xz}} & \overline{g_{yz}} & \overline{g_{zz}} - \lambda \end{vmatrix} = 0$$

2.2.1 Computation of the eigenvalues and eigenvectors: Algebraical solution

The class JAMA gives to the user the opportunity to work with matrices. The method *getRealEigenvalues* gives the real part of the eigenvalues of the input matrix. Here, all eigenvalues are strictly real and positive because the matrix is positive-defined (Hermitian). This method uses the eigendecomposition of the input matrix (diagonalization) to find these eigenvalues and eigenvectors and returns the following elements:

$$\mathbf{M} = \mathbf{Q} \cdot \Lambda \cdot \mathbf{Q}^{-1} \quad (15)$$

With $\mathbf{Q}$ a square matrix, whose columns are the normalized eigenvectors of $\mathbf{M}$, and Λ a diagonal matrix, whose elements are the eigenvalues ($\Lambda_{ii} = \lambda_i$).

$$\mathbf{Q} = \begin{pmatrix} u_0 & u_1 & u_2 \\ v_0 & v_1 & v_2 \\ w_0 & w_1 & w_2 \end{pmatrix} \quad \Lambda = \begin{pmatrix} \lambda_0 & 0 & 0 \\ 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_2 \end{pmatrix}$$

We take the lowest eigenvalue (which is always λ_2) to find the corresponding eigenvectors, that gives us the main direction.

2.2.2 Computation of the eigenvalues and eigenvectors: Analytical solution

The cubic equation obtained with the computation of the determinant of the GST is:

$$\begin{aligned} (g_{xx} - \lambda)\{(g_{yy} - \lambda)(g_{zz} - \lambda) - g_{yz}^2\} - g_{xy}\{g_{xy}(g_{zz} - \lambda) - g_{yz} \cdot g_{xz}\} \\ + g_{xz}\{g_{xy} \cdot g_{yz} - (g_{yy} - \lambda)g_{xz}\} = 0 \end{aligned} \quad (16)$$

The monic form of this equation allow us to find the roots. It is given by:

$$\lambda^3 + a \cdot \lambda^2 + b \cdot \lambda + c = 0 \quad (17)$$

With the following coefficients:

$$\begin{cases} a &= -g_{xx} - g_{yy} - g_{zz} \\ b &= g_{xx} \cdot g_{yy} + g_{yy} \cdot g_{zz} + g_{xx} \cdot g_{zz} - g_{xy}^2 - g_{yz}^2 - g_{xz}^2 \\ c &= g_{zz} \cdot g_{xy}^2 + g_{yy} \cdot g_{xz}^2 + g_{xx} \cdot g_{yz}^2 - g_{xx} \cdot g_{yy} \cdot g_{zz} - 2 \cdot g_{xy} \cdot g_{yz} \cdot g_{xz} \end{cases}$$

The three roots are real values ($\Delta < 0$), and can be computed with [9]:

$$\lambda_0 = -2.0 \cdot \sqrt{Q} \cdot \cos\left(\frac{\theta + 2\pi}{3}\right) - \frac{a}{3} \quad (18)$$

$$\lambda_1 = -2.0 \cdot \sqrt{Q} \cdot \cos\left(\frac{\theta - 2\pi}{3}\right) - \frac{a}{3} \quad (19)$$

$$\lambda_2 = -2.0 \cdot \sqrt{Q} \cdot \cos\left(\frac{\theta}{3}\right) - \frac{a}{3} \quad (20)$$

Using the following parameters:

$$\begin{aligned} \theta &= \arccos\left(\frac{R}{\sqrt{Q^3}}\right) \\ Q &= \frac{a^2 - 3 \cdot b}{9} \\ R &= \frac{2 \cdot a^3 - 9 \cdot a \cdot b + 27 \cdot c}{54} \end{aligned}$$

With three eigenvalues, we would be able to find three eigenvectors, but we are only interested in one direction, so we compute the eigenvector with the lowest λ , which is always λ_2 . It is composed of three projections along the x-, y- and z-axis, respectively u , v , w .

$$\begin{pmatrix} g_{xx} - \lambda_0 & g_{xy} & g_{xz} \\ g_{xy} & g_{yy} - \lambda_0 & g_{yz} \\ g_{xz} & g_{yz} & g_{zz} - \lambda_0 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = 0 \quad (21)$$

This give us the following system:

$$u = \frac{v \cdot g_{xy} + w \cdot g_{xz}}{\lambda_0 - g_{xx}} \quad (22)$$

$$v = \frac{u \cdot g_{xy} + w \cdot g_{yz}}{\lambda_0 - g_{yy}} \quad (23)$$

$$w = \frac{u \cdot g_{xz} + v \cdot g_{yz}}{\lambda_0 - g_{zz}} \quad (24)$$

All components are linearly dependant, and thus, we cannot find them without setting one arbitrarily. This is not a problem, because the computation of the angles rely only on the ratios of these projections [9].

We have seen two possibilities to compute the eigenvector, one algebraic and one analytical. The advantage of the first one is that all components are known and normalized. Therefore, we can use them directly to compute the angles. The second one gives us only the ratios, but the accuracy is exactly the same, and, as said previously, these ratios are enough to compute the angles. So the primordial element here is the computation time, which is almost the half with this second method than with the first one. Thus, we chose to use it, because with large images (512x512x200 for most of the samples), this computation time is a serious issue.

2.3 Computation of the two angles φ and θ

The two angles are calculated with the ratios of the components u, v, w of the eigenvector, using the spherical coordinates, with $r = 1$ ([2][9]):

The angle φ is given with the ratio u/v , θ with u/w and v/w :

$$\varphi = \text{atan2}(u, v) \quad (25)$$

$$\theta = \text{acos} \left(\left(\frac{u}{w} \right)^2 + \left(\frac{v}{w} \right)^2 + 1 \right)^{-1/2} \quad (26)$$

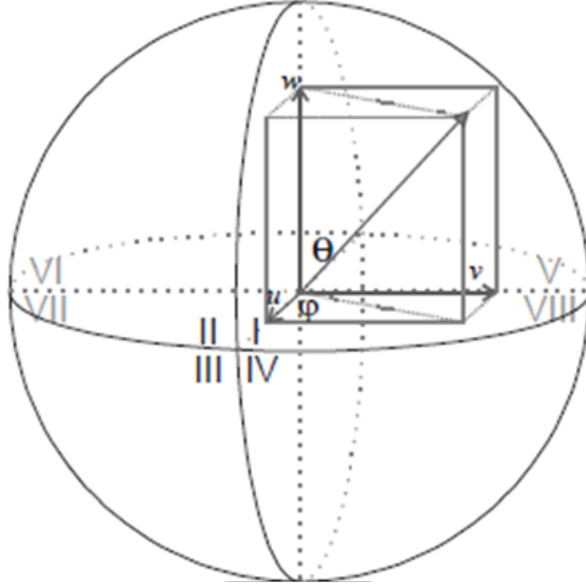


Figure 5: Projection of the eigenvector's components u, v and w , and corresponding angles [9]

With:

$$\frac{u}{v} = \frac{g_{xy}(\lambda - g_{zz}) + g_{xz} \cdot g_{yz}}{g_{yy} \cdot g_{zz} - g_{yz}^2 - (g_{yy} + g_{zz})\lambda + \lambda^2}$$

$$\frac{u}{w} = \frac{g_{xy} \cdot g_{xz} + g_{yz}(\lambda - g_{xx})}{g_{xy}^2 - (\lambda - g_{xx})(\lambda - g_{yy})}$$

$$\frac{v}{w} = \frac{g_{xy} \cdot g_{yz} + g_{xz}(\lambda - g_{yy})}{g_{xy}^2 - (\lambda - g_{xx})(\lambda - g_{yy})}$$

The *arccos* function gives us values in the range $[0; \pi/2]$ because of the squares of the ratios (only positive values). Thus, only half of this sphere could be represented. The second half is reached when $w < 0$, but with our method, this value is not computed. We will have to work with the ratios again (u/v , u/w , v/w have been calculated). The following table summarize the conditions that lead to a negative w :

For all these cases: $\theta = \pi - \theta$.

Table 1: Conditions that lead to a negative w

φ	sign of u and v	$w < 0$ if
$0 < \varphi < \pi/2$	$u > 0$ and $v > 0$	$v/w > 0$ and $u/w > 0$
$\pi/2 < \varphi < \pi$	$u < 0$ and $v > 0$	$v/w < 0$ and $u/w > 0$
$0 > \varphi > -\pi/2$	$u > 0$ and $v < 0$	$v/w > 0$ and $u/w < 0$
$-\pi/2 > \varphi > -\pi$	$u < 0$ and $v < 0$	$v/w < 0$ and $u/w < 0$

2.4 Coherency feature

Coherency or anisotropy measure is a value computed with the eigenvalues of the GST, comprised between 0 and 1. It could be computed from several different ways, but always represent the same idea: The ratio of these eigenvalues are representative of the shape of the analysed object within the smoothing window. For example, spherical objects will have eigenvectors of the same size in each direction. For tubular objects, one will be smaller than the two others. This is why we chose to implement a coherency value using the ratio between the mean eigenvalue and the total sum.

$$coherency = \frac{\lambda_2}{\lambda_0 + \lambda_1 + \lambda_2} \quad (27)$$

3 Construction of the bidimensional histogram

In order better to see the different orientations, we construct a bidimensional histogram with two angles φ and θ as inputs. A condition on the energy of the pixel ($g_{xx} + g_{yy} + g_{zz}$) is taken to get rid of outside-structure elements, that give false orientations, even if they have a high coherency. This threshold is given by the user as a percentage of the maximum energy of the pixels.

The histogram is a 180x180 pixels image built by summing the coherency of the current pixel ($x = \varphi$, and $y = \theta$) if the threshold is reached, for each angles from 0 to 180°.

We expect with this kind of construction to find an aggregation of points around the same area for elements with a single orientation. This will be the case for our validation volumes with unidirectional lines.

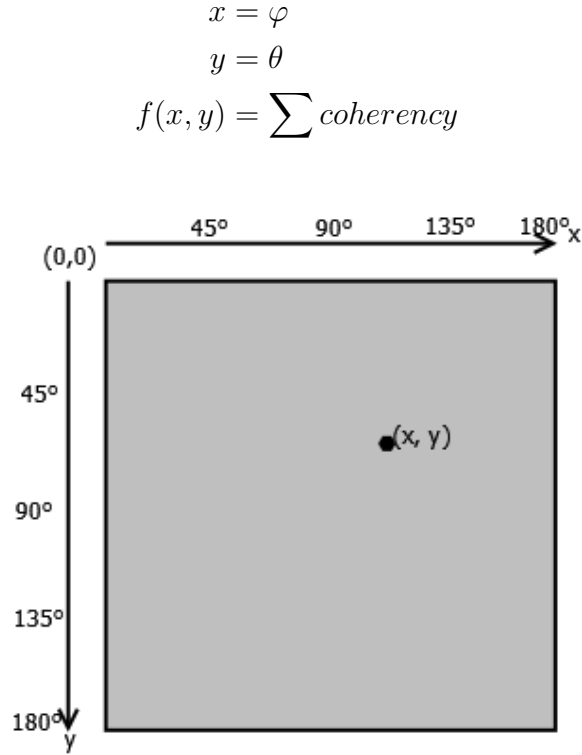


Figure 6: Histogram

3.1 Clustering on the bidimensional histogram

An additional plugin allow the clustering of the data. Because we do not know the number of classes a priori, we could not implement a classification like the K-means algorithm. This is why we chose to use the QT-clustering algorithm, that does not need a given number of classes but has a higher computational cost. It consists on combining points of the bidimensional histogram depending on their weights and distance with each others. The threshold on the distance is given as an input parameter, as the maximum number of iteration. The flowchart 7 shows the operations and ending conditions:

A special attention has to be paid to the computation of the distance. There exists several different ways to calculate this (Manhattan, Euclidian, Minkowski, etc.). We chose here to compute the Euclidian distance, which is given by:

$$distance = \sqrt{|x_2 - x_1|^2 + |y_2 - y_1|^2}$$

But the boundaries of the histogram are periodic because we are looking at angles. Thus, special conditions have to be implemented:

$$\begin{aligned} x &= |x_2 - x_1| \\ \text{if}(x < 90) &\rightarrow x = 180 - x \end{aligned}$$

And idem for y (both φ and θ are $\in [0; \pi]$). When the algorithm ends, the output histogram contains cluster composed of summed points at barycentre positions. A second iteration labels the higher points with a colour to give a better look at the most represented angles.

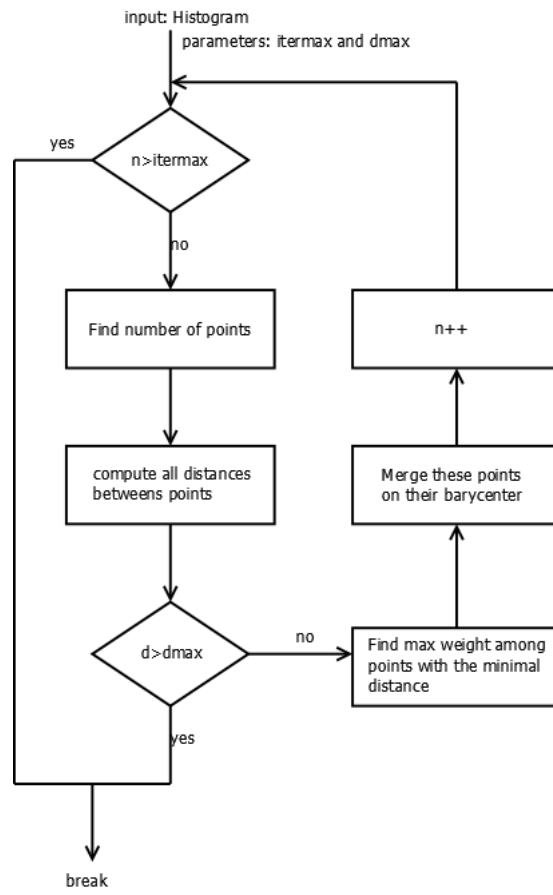


Figure 7: Flowchart of the QT-clustering algorithm. The *itermax* ending condition ensure that the program will not run indefinitely, and the *dmax* gives the final clusters dimension.

4 Colour representation

A nice way to look at the orientation of the fibres has been given by Pajevic et al. [7]. It consists of giving a color to the pixel depending on the computed angles. This could be done with several different methods. We chose to implement the color representation by giving the Hue channel φ and the Saturation channel θ . To avoid sudden change of color, the function given to the saturation channel is: $s = 1 - \frac{|\theta - 90|}{90}$.

Thus give us white color for θ near 0° , and maximal saturation for $\pm 90^\circ$. Finally, the Brightness channel contains a normalized version of the input image, that allow us only to see the interesting regions, where fibres are located.

$$\begin{aligned} h &= \varphi \\ s &= 1 - \frac{|\theta - 90|}{90} \\ b &= \text{input} / (\text{input.Maximum}()) \end{aligned}$$

With this method, all angles could be represented by a specific colour. To visualise this, the picture 8 shows a unit sphere and the associated colour depending on φ and θ ([7]):

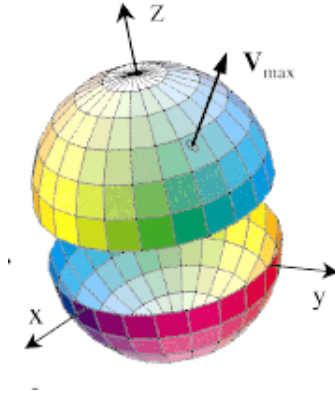


Figure 8: Unit sphere with associated colour for given angles

Another way to look at the angles with colours is to label only one of the two on coloured images. The visualization for the human eye is easier, because the information is only contained in the hue channel. Therefore, we will have two new coloured images containing the colour for φ and θ respectively. The saturation channel could be filled with the coherency or set to 1 (maximal coherency) to have fully saturated colours.

$$\begin{aligned} h &= \varphi \text{ or } \theta \\ s &= \text{coherency or } 1 \\ b &= \text{input}/(\text{input.Maximum}()) \end{aligned}$$

5 Validation of the algorithm

We made a second plugin that constructs stacks of images containing lines with a single orientation ($\varphi = cste$ and $\theta = cste$), in order to validate the first one. With these special cases, we expect the bidimensional histogram to have only one point at the coordinates of φ and θ .

To construct these lines, the user can give the two angles, and the plugin computes the rotation matrix that performs the following rotations for a unit vector $\mathbf{e} = [1, 0, 0]^T$:

1. Rotation around the Z-axis of φ

$$R_\varphi = \begin{pmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2. Rotation around the $[\cos(\varphi), -\sin(\varphi), 0]^T$ - axis of $\pi - \theta$

$$R_\theta = \begin{pmatrix} 1 + (1 - \cos\theta)(\sin^2\varphi - 1) & (1 - \cos\theta)(-\cos\varphi\sin\varphi) & \cos\varphi \sin\theta \\ (1 - \cos\theta)(-\cos\varphi\sin\varphi) & 1 + (1 - \cos\theta)(\cos^2\varphi - 1) & \sin\varphi \sin\theta \\ -\cos\varphi \sin\theta & -\sin\varphi \sin\theta & \cos\theta \end{pmatrix}$$

Pictures 9 and 10 show the created volumes for different angles.

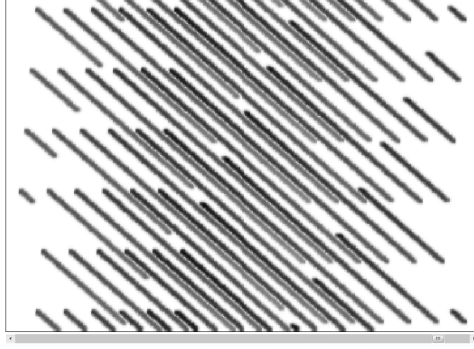


Figure 9: 3-D projection of created volume filled with lines: $\varphi = 30^\circ$ and $\theta = 70^\circ$

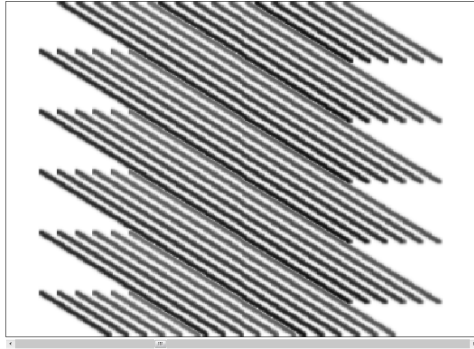
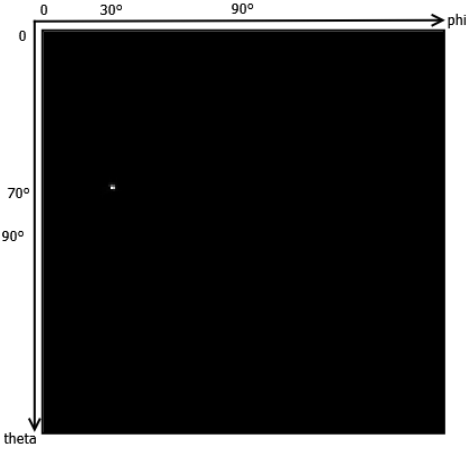
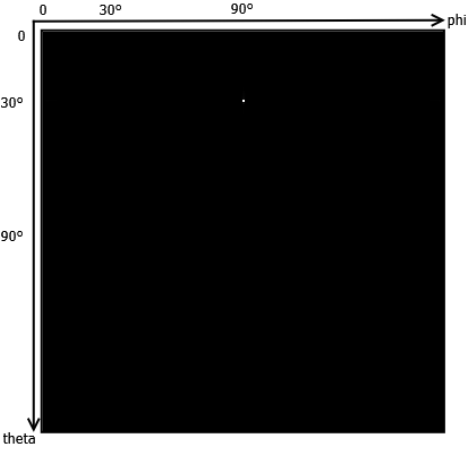
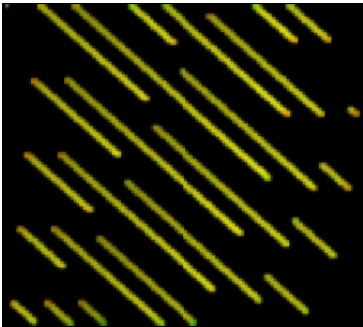
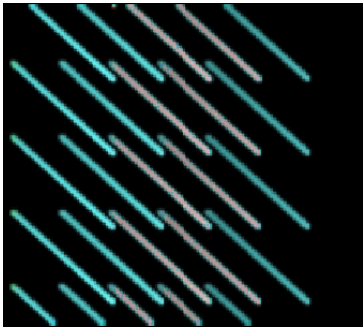


Figure 10: 3-D projection of the created volume filled with lines: $\varphi = 90^\circ$ and $\theta = 30^\circ$

5.1 Histograms and Color representation on validation volumes

The plugin has been run on the previous samples ($(\varphi = 30^\circ ; \theta = 70^\circ)$ and $(\varphi = 90^\circ ; \theta = 30^\circ)$). As expected, the resulting histograms contain only one single point at the corresponding coordinates. And the color representation gives a single colour, as explained in the previous section.

Table 2: Results on test volumes, filled with unidirectional lines.

(30;70)	(90;30)
Histogram 	Histogram 
Color representation 	Color representation 

We can see that the plugin works perfectly for these special cases, but what happens if there is more lines with different orientation? Picture 11 shows the 3-D projection of the sum of the two previous volumes, and table 3 the resulting histogram and colored image:

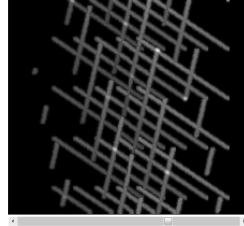
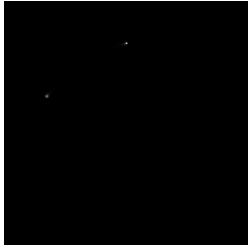
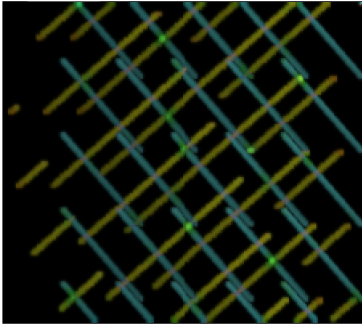


Figure 11: Created volume, filled with two sorts of lines with different orientations.

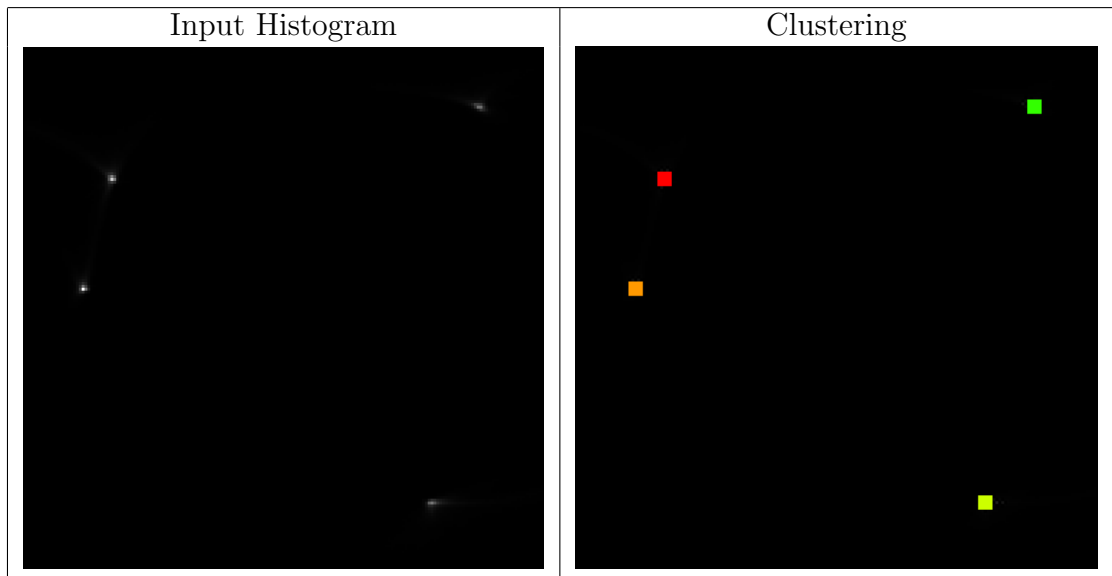
Table 3: Results

(30;70) + (90;30)	
Histogram	
	
Color representation	
	

5.2 Validation of the clustering algorithm

The clustering algorithm works very well for this kind of data (unidirectional lines give an histogram with grouped points, that look already like clusters). Table 4 show the coloured barycentres of the four clusters created with the algorithm. The input image was filled with lines in four different orientations.

Table 4: Validation of the clustering algorithm.



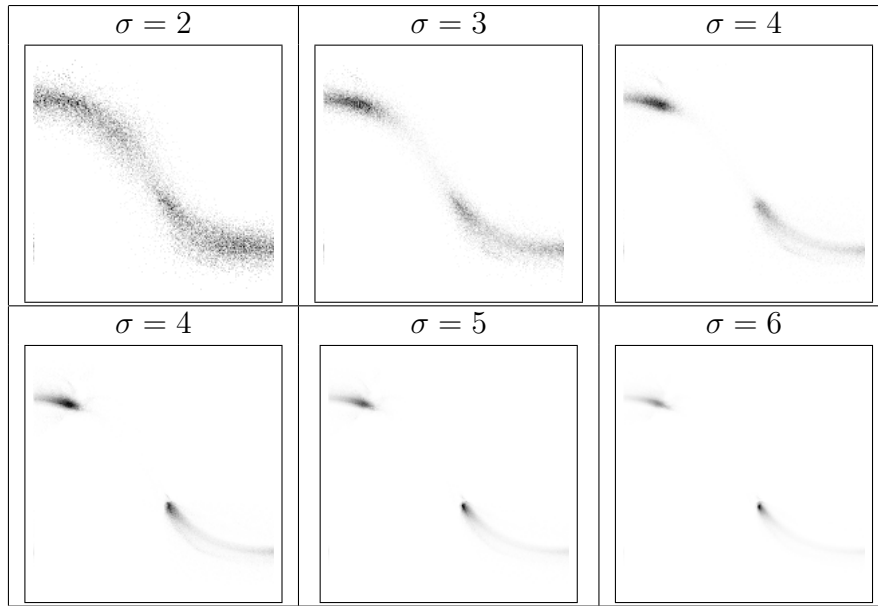
5.3 Effects of the input parameter: σ

The accuracy of the measure depends on this parameter, and thus, it must be chosen with care. In this section, we will see its effect on the histograms with test volumes filled with multi-directional lines.

This parameter will enlarge the Gaussian used for the smoothing. It will have an effect on the accuracy of the detection of structures, depending on their size. Indeed, the detection of structures with various sizes will be possible only if we look at the scale at which they give a maximal response. For very small structures, we will use a quite small σ , such as 1 or 2, and up to 5 or 6 for larger structures. We can see several different scales on collagen elements, and consequently, the response will vary with the σ chosen.

Table 5 contains the bidimensionnal histogram obtained with the same volume (with (30,40) and (100,70) orientated lines). An optimal σ is reached around 5, 5.5, 6 for these elements.

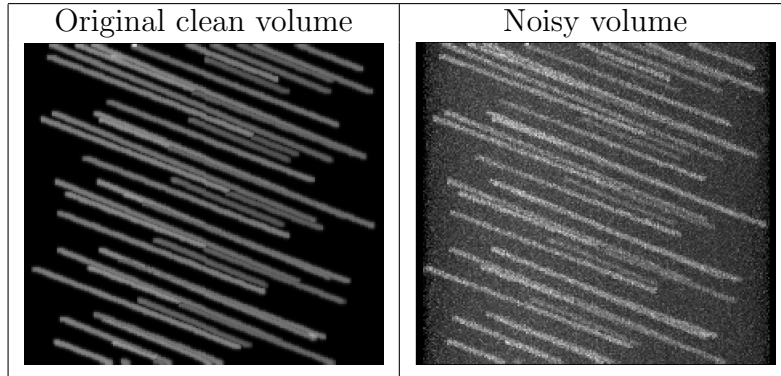
Table 5: Histograms



5.4 Effect of the noise

Even if collagen 3-D images taken with a confocal microscope are very clean, but we will see the effect of a Gaussian noise on the angle detection. Still using the test volumes with several lines of different orientations, we add noise until the algorithm cannot find the correct angles.

Table 6: Volumes



We can see that noise blur the histogram, giving more and more false orientations. The last image, with $\text{SNR} = -11$ dB, shows an interesting thing. The orientations given by this last histograms converge to the angles 45° and 135° for both φ and θ . This may be due to the Gaussian smoothing that could have made crosses with single points over darker background instead of circles, or by the computation of the gradient. This effect is exactly the same with salt and pepper noise. Table 8 gives a second analysis with other orientations, but the effect of the noise brings the histograms to the same configuration of the first example.

Table 7: Histograms

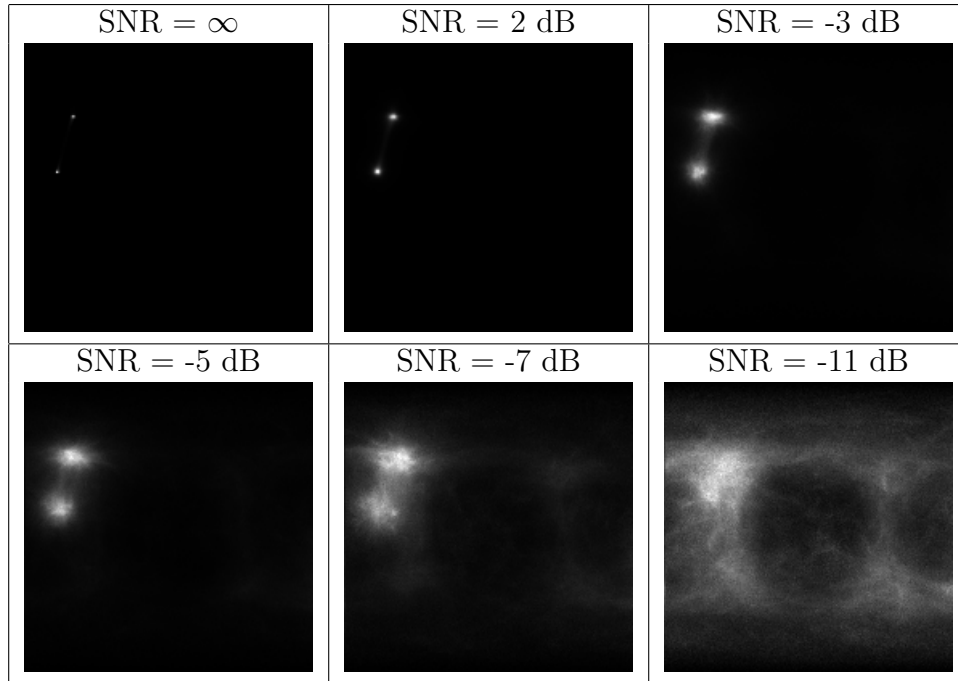
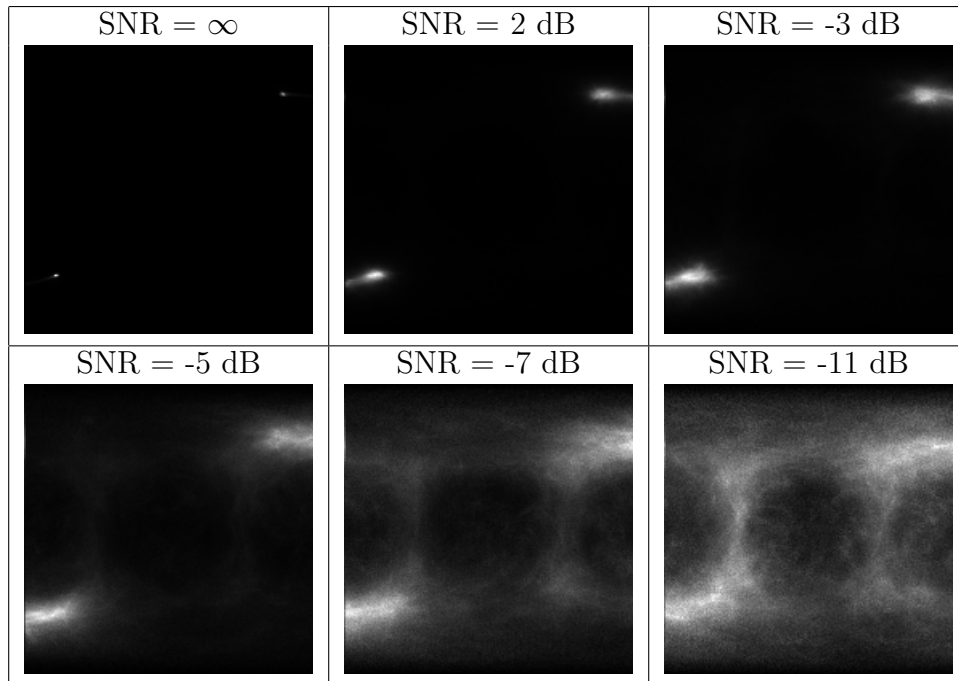


Table 8: Histograms



6 Orientations analysis of 3-D collagen data

Now that all the validations of the algorithm have been made, we can test it on real data. The slices are 512x512 images, and the stack contains approximately 100 to 150 slices. The computational cost is increased considerably compared with our previous test volumes (only 200x200x100).

In this section, we will compare the results obtained with our algorithm and the standard method used to detect the orientation, which consists of computing the maximal intensity projection (M.I.P.) on the z-axis and look at the φ angle. This method could bring some errors due to the projection: overlapping filaments for example. Then three sorts of color representation will be implemented and compared.

First of all, let us have a look at the volumes to have a global idea of the orientations. We will focus on 3 representative stacks. Here are the snapshots:

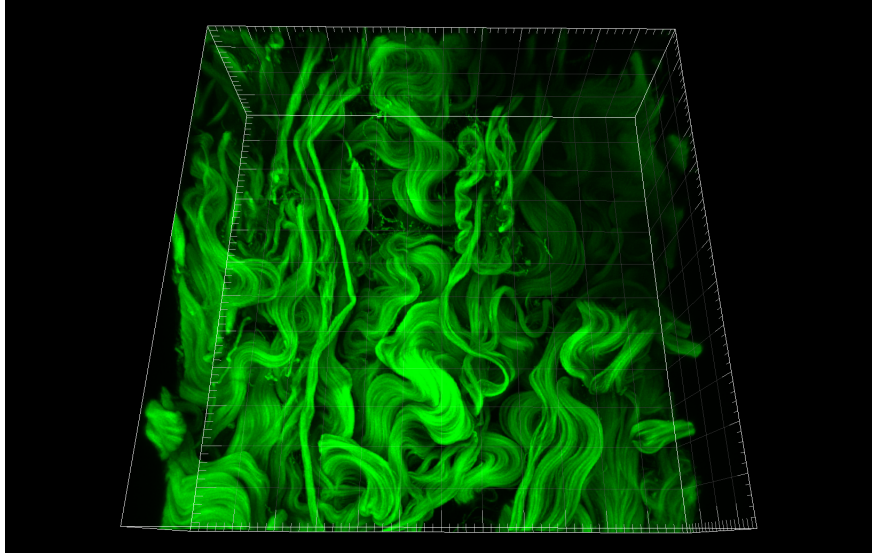


Figure 12: Volume 1: cut 19.8 serie03

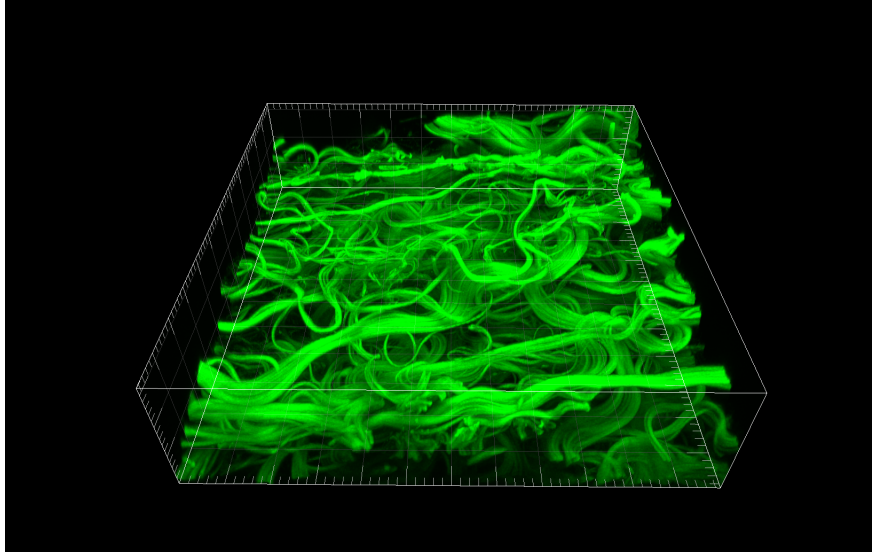


Figure 13: Volume 2: uncut 19.8 serie03

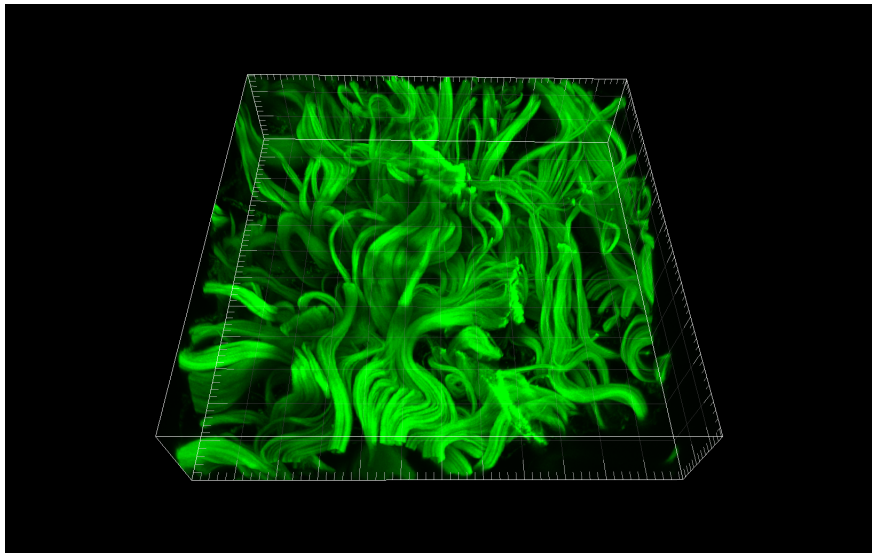
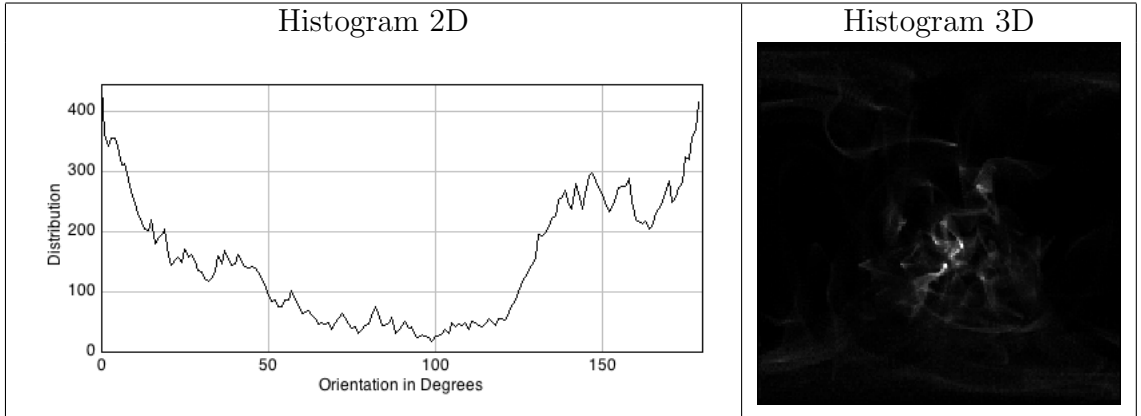


Figure 14: Volume 3: uncut 19.8 serie09

The plugin *OrientationJ* allows the detection of the orientations in the $x - y$ plane, i.e. the angle φ for each slice separately or for the max intensity projection. The following table gives the computed 2-D histogram and our 3-D bidimensional histogram for the first volume (cut 19.8 serie03 13 with $\sigma = 5$):

Table 9: Histograms for the first volume



The horizontal is represented by a null φ in our plugin, and by $+90^\circ$ in *OrientationJ*. Thus, we must shift one or the other to compare them. If we shift back the first one of -90° we can see that the peak of the distribution takes place approximatively for angles between 80 to 100° , which is the same amount for the second histogram in the x-axis (corresponding to φ). The following pictures give the histograms for the second and third volume, and we can see the same results for each of them, with this -90° shift:

We will now focus on the color representation. If the histograms are similar, then the colors will be equivalent too. We can see on pictures 15 and 16 that colours are shifted (red in one picture correspond to blue in the other). Picture 16 is the maximum intensity projection of the output of our plugin for the angle φ . Only the first volume 12 will be shown here, the others give similar results:

According to these images, we can see that there is little differences if we compute the φ angle with *OrientationJ* or with our plugin and then take the maximum intensity projection of the coloured φ image.

Table 10: Histograms for the second volume

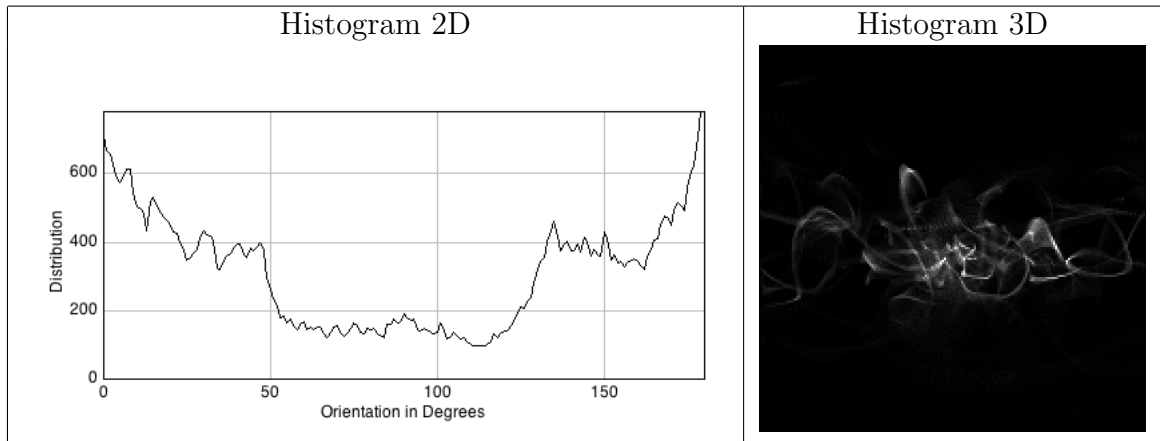
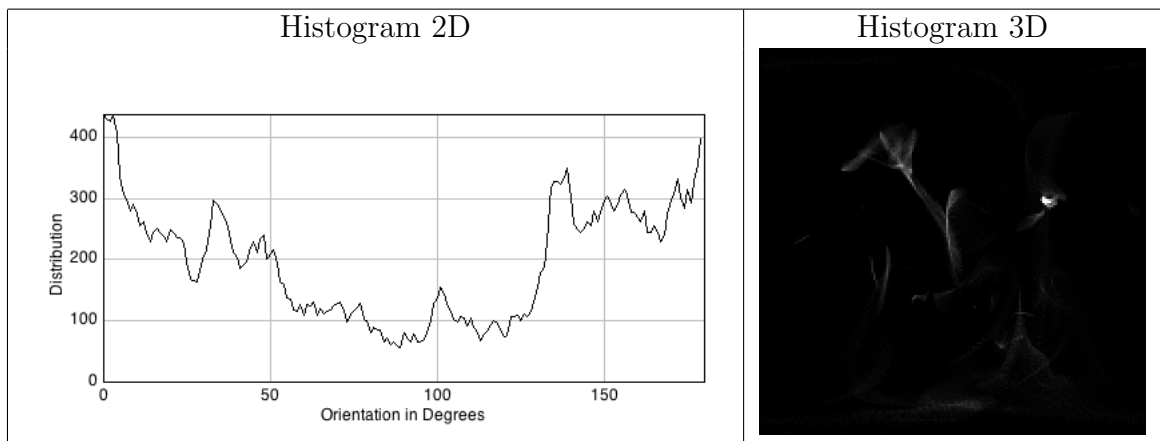


Table 11: Histograms for the third volume



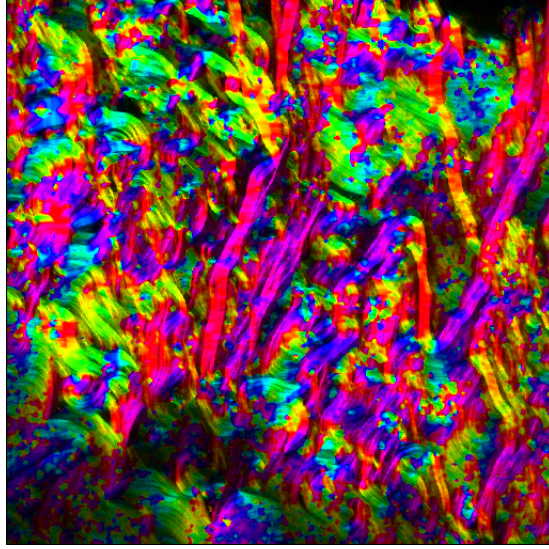


Figure 15: Color representation of the angle φ obtained with *OrientationJ*, with the M.I.P. of the volume as input.

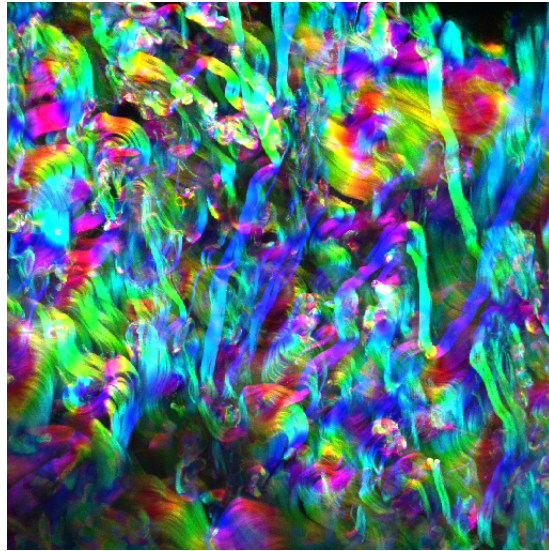


Figure 16: Color representation of the angle φ obtained with our plugin. Display only represents φ (section 4): The hue channel contains φ , saturation is set to one, and the value channel contains the normalized input.

Picture 17 now shows the coloured image depending on angle θ of the third volume 14. Again, the maximum intensity projection is displayed, with saturation set to one to give a better representation (it could have been set with the coherency value to attenuate area where filaments are not well represented).

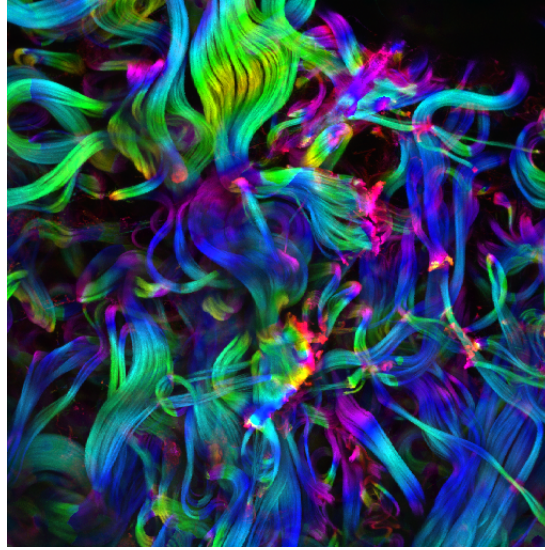
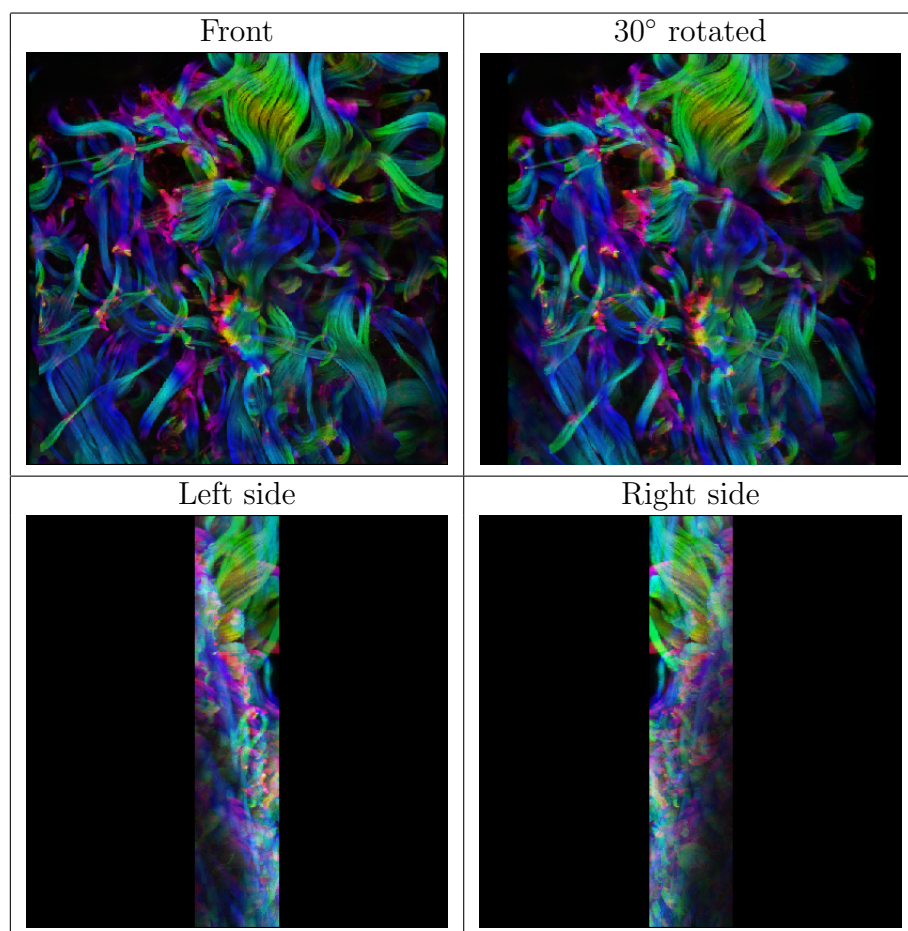


Figure 17: Max. intensity projection of the color representation of the angle θ obtained with our plugin. The hue channel contains θ , saturation is set to one, and the value channel contains the normalized input.

We cannot compare this result with other existing plugins, so we will see here the 3-D projection of this volume, and look at the original image 14 to compare orientations (this volume has clearly a preferential orientation in the angle θ , it will be easier to see it):

Table 12: Three images of the 3-D projection of the coloured image representing angle θ , rotated around the y-axis.



The left and right side of this projection allow to see the angle θ in the x-z plane, as if we were looking for angle φ in the x-y plane. We can see that two preferential orientations occur, coloured in green and blue. Looking back at the histogram, we can notice that a peak takes place for $\theta = 70^\circ$, and if we increase the brightness of the image, another one appears for $\theta = 10^\circ$.

Finally, let us have a look at the clustering plugin on one of these histograms. Even if the histogram is not as clear as the one for the simple lines, the clustering algorithm can retrieve several mean directions and label them. Figure 18 shows the output images for the second volume 13:

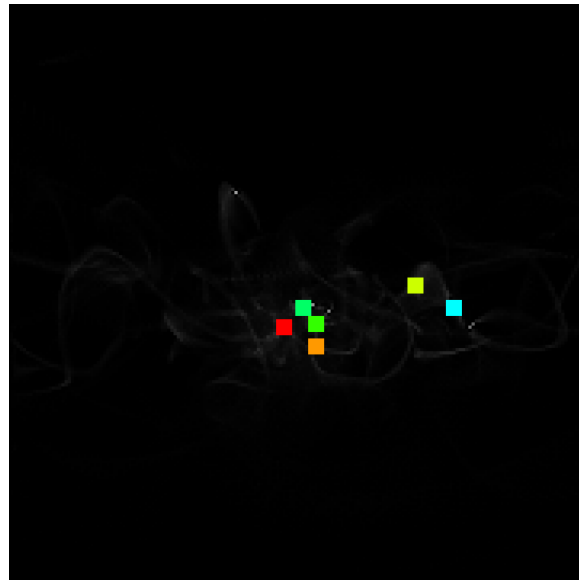


Figure 18: Labelled clusters on the histogram of the angles obtained for the second volume. When the algorithm stops, 6 main orientations are pointed out.

7 Conclusion

This work described a way to implement an automatic analysis tool to detect orientations of fibres in 3-D images using local neighbourhood analysis. Several steps were necessary to achieve this aim:

- Computation of the Gradient Structure Tensor (GST):
 - Computation of the gradients in all three directions, using a cubic splines method.
 - Smoothing of the matrix' components, using a Gaussian smoothing with a given σ .
- Computation of the eigenvalues, which is a cubic equation resolution, with three real positive roots.
- Computation of the eigenvectors, or at least, the ratios of these eigenvectors.
- Computation of the two angles φ and θ .

Once these steps achieved, an bidimensional histogram was build to show principal orientations of the input image. On these histograms, a QT-clustering algorithm can be run to automatically label the most represented orientations. Finally, a color representation was implemented to allow the direct visualization of the angles.

We have seen that this automatic analysis of the angle works perfectly on test images, and thus, can be used to detect orientations of more complex structures like collagen fibres. The goal of the biologists was to analyse if there exists principal directions in these collagen structures. The results shown at the end of this work point out that some orientations are preferential, because the histograms were composed on grouped points on some specific coordinates, and the coloured images confirmed this fact.

Further work can be done, in particular for the clustering algorithm. The QT-clustering algorithm works quite efficiently to detects these clusters, but the maximal distance parameter has to be given with care. A K-means or equivalent could be implemented if the user know the number of clusters a priori. It would be more efficient and would have less computational cost. A tracking of one filament of collagen could be implemented too. It would be helpful to check if the given orientation is correct, and to allow the measure of its lengths, to give the information about its deformation.

References

- [1] M. Axelsson. An evaluation of scale and noise sensitivity of fibre orientation estimation in volume images.
- [2] F. Daniels. Quantification of collagen orientation in 3d engineered tissue, 2006.
- [3] F. G. Faas and L. J. van Vliet. 3d-orientation space; filters and sampling.
- [4] J.-M. Geusebroek, A. W. M. Smeulders, and J. van de Weijer. Fast anisotropic gauss filtering. *IEEE transaction on image processing*, 2003.
- [5] M. V. Ginkel. Image analysis using orientation space based on steerable filters, 2002.
- [6] B. Jähne. *Digital Image Processing*. Springer, 2005.
- [7] S. Pajevic and C. Pierpaoli. Color schemes to represent the orientation of anisotropic tissues from diffusion tensor data: Application to white matter fiber tract mapping in the human brain.
- [8] M. Unser. *Image Processing, Volume 2*.
- [9] G. M. van Kempen, N. van den Brink, and L. J. van Vliet. The application of a local dimensionality estimator to the analysis of 3-d microscopic network structures.
- [10] Y.-C. Wu, A. S. Field, M. K. Chung, and B. Badie. Quantitative analysis of diffusion tensor orientation : Theoretical framework. *Magnetic Resonance in Medicine*, 2004.