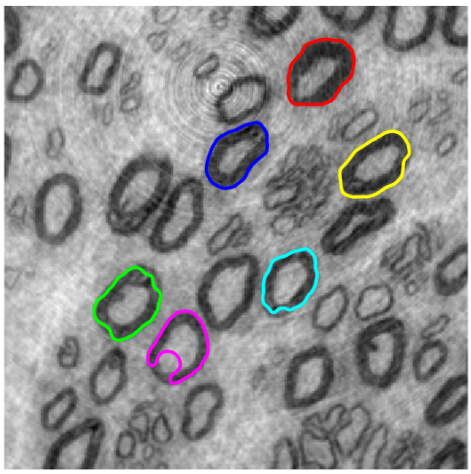
Nerve fiber segmentation using deformable models

For segmenting the nerves fibers a deformable model was chosen as the method used. The deformable model relies on the Mumford-Shah functional and the Chan-Vese algorithm for solving the functional, and uses snakes (parametric) as curve representation.

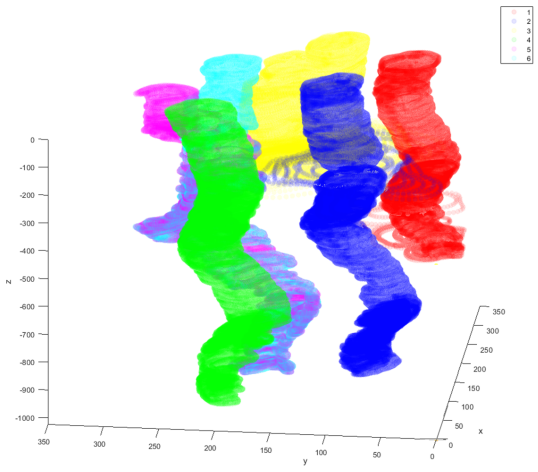
The nerve fibers were segmented and tracked one at a time, where inputs to the developed function was the initial placement of the circular snake. The function then deformed the snake in five iterations for each image slice. For the next image slice, the last snake from the previous slice was used as the initial snake. The function was able to track a nerve until a node of Ranvier, where the nerve fiber disappears. A simple method for continuing the algorithm is implemented: if the snake gets out of bound in the image, then the snake from 20 slices back is used as initializing snake. When the nerve fiber begins to "grow back", the nerve fibers are segmented correctly again.

Six nerve fibers were segmented. Parameters used in the algorithm were as follows $m_{in} = 30$, $m_{in} = 150$, $\alpha = 0.2$, $\beta = 0.2$ and $\tau = 0.005$. Below is seen a slice of segmented nerve fibers, and a 3D model.

Nerve 3 (yellow) at some point is close to another nerve, and the algorithm then includes the other nerve in the segmentation. Nerve 5 and 6 (magenta and cyan) at some point are detected as the same nerve. This problem should be accounted in further improvements of the segmentation algorithm. In several cases (especially after a node of Ranvier) the algorithm only segment part of the myelin sheath, and not the whole nerve fiber with the high intensity middle, which also should be corrected by altering the deformation criteria, which currently is based on mean intensities.



(a) 1^{st} slice with the six segmented nerves.



(b) 3D view of the segmented nerve fibers.

DTU Space
National Space Institute

Using Geometric Priors For Volumetric Segmentation

Pernille Kliving, Danjal Berg, Eigil Yuichi Hyldgaard Lippert
Technical University of Denmark, Earth and Space Physics Engineering

1 Introduction

Two methods, Markov Random Fields (MRF) and Chan-vase deformable models are used for volumetric segnemtation. The data are of myelinated nerves and the objective is to obtain a 3D model of the nerves. To visualise the nerves are a cross section showed in the following example.

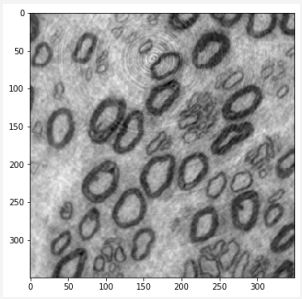


Figure 1: Example of one slice of the (1024, 350, 350) volumetric image.

2 MRF

In MRF are energy terms calculated for the one-clique and two-clique potential. The one-clique potential is the likelihood energy and the two clique potential is the prior. The sum of these energies gives the posterior energy that we want to minimize for a given configuration.

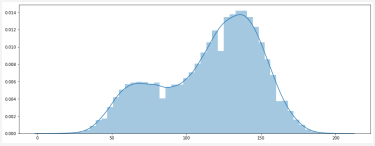


Figure 2: Histogram of one slice of the (1024, 350, 350) volumetric raw image.

The images of the myelinated nerves are very noisy. Values close to 0 are black and close to 255 are white. To preprocces the images are all values of the image above a pixel intensity of 100 set to 255 to achieve a white background and better contrast.

Graph cuts are used to get a smooth result. A smoothness parameter β is set to 100 in the $[x, y]$ direction and 500 in the z direction.

The smoothing parameter are able to remove most of the remaining noise but not everything. Dependent on the value of the smoothing parameter is there a trade-off. If the myelinated nerves are very close to each other they "merge" together as can be seen in figure 3.

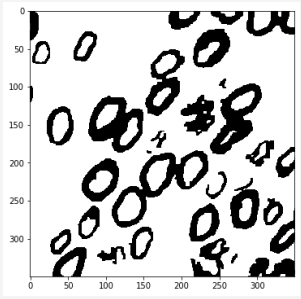


Figure 3: Graph cut of one slice of the volumetric preprocessed image.

3 Deformable models

The first 500 nerve images was normalised to floats between 0 and 1 as a preproccing step and used in the following. The approach used is a Chan-Vese approach combined with a parametric curve as described in the course notes. The parameters used for the process was: $\alpha = .25, \beta = .05$, number of curve points was $n = 75$. For the first slice of the nerve image the step size was 30, with 40 iterations, and for the subsequent slices the step size was 20 and the number of iterations reduced to 10. All curves were initialised with different radii to enclose the nerve of interest but not hit its neighbors.

In Figure 4 we see how the evolution of the snakes enables a 3d model of the nerves, with the curves which performed poorly in the background and the good ones in front. The cyan and blue curves did really well, and it can be seen how they both catch a node of Ranvier each, for the cyan curve this occurs around $z = 400$ and for the blue curve around $z = 200$. The performance of the curves was mainly assessed through the video in the appendix, but three snapshots from the video are seen in Figure 5.

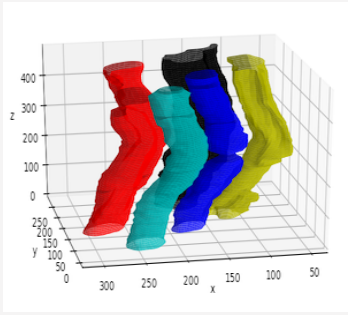


Figure 4: The evolution of curves segmenting nerves

Especially the yellow and black curves are not performing very well and gets stuck on neighboring nerves. The red encloses the neighbor in $z = 250$ and between that point and $z = 480$ it has a short period of enclosing the nerve perfectly. This was also observed for the yellow curve, although briefly.

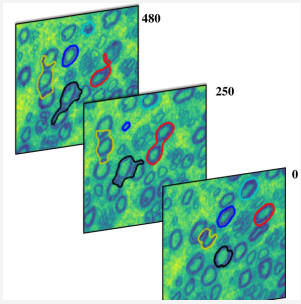


Figure 5: Segmented nerves at slice 0, 250 and 480

Various things where tried out to improve the result for example an additional term to the update term which made a step in the negative gradient direction, which proved to cause too many instabilities in the end. Another idea was to stop updating after some condition was met. The most promising condition was computing the mean in a thin band just inside the curve, and stopping the updates as this mean became stable. While it seemed to work for a single curve, it proved difficult to implement for all curves due to complexity and probably lack of generality of this fix.

02506: Advanced Image Analysis Exercise 7

Charlotte Friis Theisen, 01/04/2020

1. Introduction

The purpose of exercise 7 is to find interesting properties of human peripheral nerves. Of interest are the *nerve density count*, the *myelin density*, the *average nerve area* and the *average nerve radius*. These will be examined using two approached, namely a Markov random field (MRF) model and deformable models (DM).

2. Method

2.1 3D -MRF-model

The 3D MRF model is used to segment the image of the nerves into two classes. Class one is myelin and class two is everything else (includes background and axon). The MRF-model is used with graph-cuts to get the optimal segmentation. The pixel intensity of class 1 is determined to be approximately 40 and of class 2 it is determined to be 150. The likelihood energy is used as weights from sink and source to each node and all neighboring nodes are linked by an edge. This includes neighbors in both x-, y- and z-direction. The weight of these edges are in the x- and y-direction set to 100 and in the z-direction to 1,000. To incur a higher degree of smoothing in the z-direction. Only a subset of the first 100 images was used to reduce computation.

2.2 Deformable models

The deformable models are implemented using snakes to trace eight different clearly defined nerves throughout the volume. The model is initialized as round circles with 100 points equidistantly along the snake. They are manually initialized as shown in Figure 2. The same pixel intensities are used to determine the threshold which is calculated as described in the course note. The force is also calculated according to the course note. The pixel intensity values under the snake is determined as the intensity of the nearest pixel by rounding the non-integer snake-values. To regularize the snake without shrinkage an implicit smoothing was used, where alpha was set to 0 and beta to 20. The stepsize is set to 2. The new snake is calculated again according to the course note and the points are redistributed equidistantly along the snake and intersections are removed. The algorithm is run 150 iteration on the first/bottom image to fit all snakes well to the nerves. Afterwards the algorithm is run 10 iterations for each image allowing the snakes to adapt slightly to any changes in the nerves.

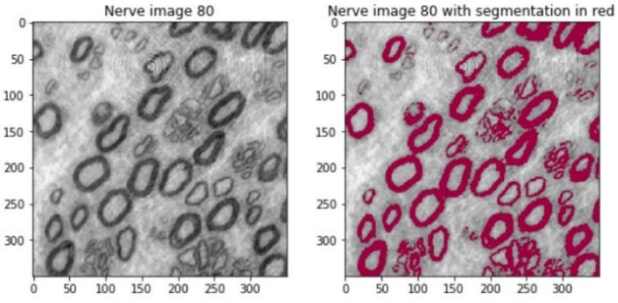


Figure 1: Result of MRF model on image slice 80.

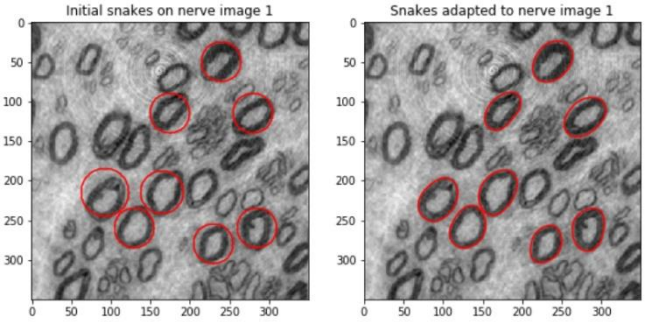


Figure 2: Deformable model. Initial state and snakes adapted to nerve image 1.

3. Results

The result of the MRF-model is showed in Figure 1. Here two slices of the volume are visualized. On the left is the original slice and, on the right, it is overlaid with the segmentation in red. As seen the segmentation is quite good, only missing a few of the very small nerves. Some cluttering is apparent in the areas with small entangled nerves. The deformable models are fitted well to the first image (see Figure 2) and adapts to the changes in the nerves throughout the volume. However, by inspecting every 20th image in the volume it is clear that most of the snakes at some point diverges from the nerve, typically because the nerve becomes narrow. The snake captures the narrow part but does not adapt well as the nerve radius increases again. Only the well-behaved snakes are used in the following calculations.

The myelin density is estimated from the MRF model by calculating the pct of pixels segmented as myelin, this is found to be 25.9%. The average nerve area is divided into average myelin area and average axon area. This is found by applying the snakes as a mask to the segmentations, such that only the areas inside snakes are used. This gives a myelin area of $9.55 \cdot 10^6\ nm^2$ and an axon area of $28.2 \cdot 10^6\ nm^2$. If we by approximation assume the nerves to be circular, this is equivalent to an average axon radius of $21.2 \cdot 10^2\ nm$ and average myelin thickness of $3.32 \cdot 10^2\ nm$.

4. Discussion and conclusion

Both of the models could be improved. The parameters could be tweaked to get improved performance. In the deformable models the image could be unwrapped to calculate a better direction of movement and the in and out pixel intensities could be dynamically updated. In Figure 3 an example of when the deformable model fails is shown. However, we also get information about this probably being a node of Ranvier, which is also of interest.

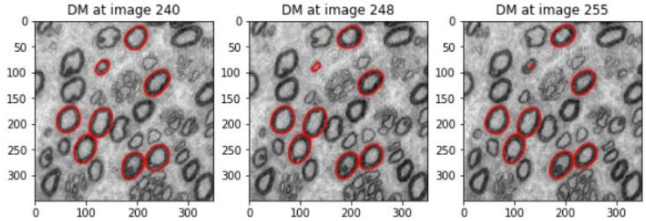
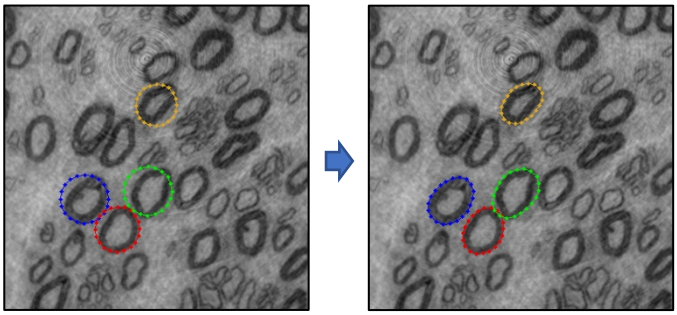


Figure 3: DM loosing track of a nerve.

Jacob Fiskaali Hertz

Cell segmentation by using deformable model

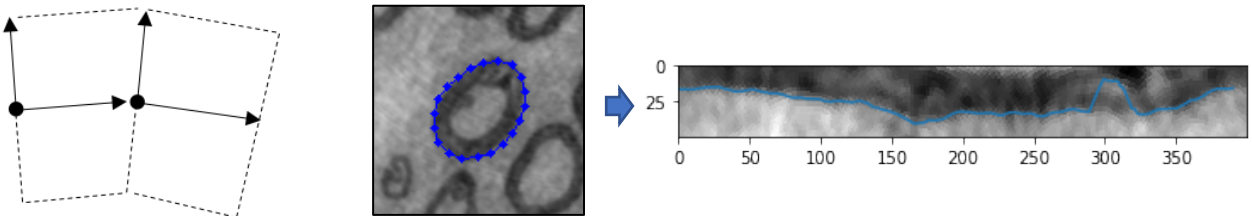
The snakes forming the deformable models are first initialized as approximated circles that are manually positioned around the cells to be tracked in the first image layer. Each snake has a set of 20 joints.



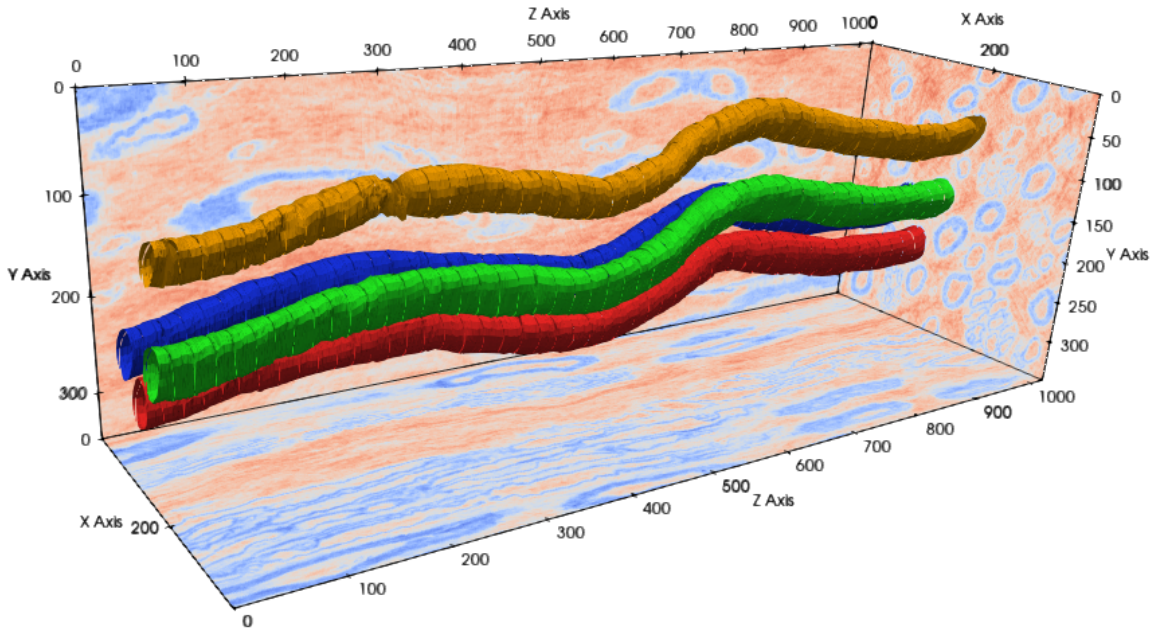
By using a fixed threshold for m_{in} and m_{out} with values of *70* and *140* respectively the snakes are pre-adjusted to fit the cells by running multiple deformation steps on the same initial image.

The procedure is now continued by applying a single deformation step for each image layer in the voxel, but in addition to regulating each joint in accordance to its corresponding image intensity, the snake is further adjusted to the edge of the cell by differentiating the image.

This is done by first unwrapping the snake along its normal vectors, applying a simple 5-by-5 vertical oriented Sobel filter and then extracting the maximum values. After smoothing the contour, the row-index of these max-values determines the normal forces for the snake adjustments.

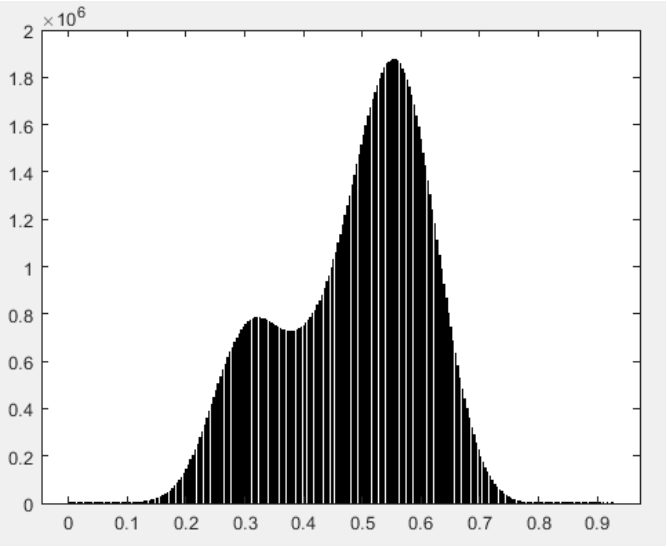


Running this algorithm on the four selected cells results in the following 3D model.



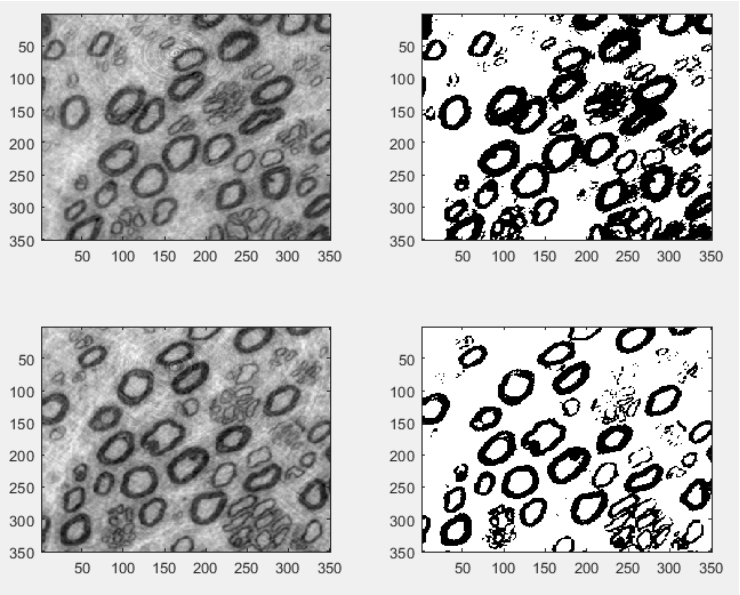
As seen in the orange cell contour, the method is able – though with some difficulties – to track the node of Ranvier. It's the unwrapping techniques that allows the snake to adjust to the rapid change in the size around this point. However, to improve the robustness of this method, when finding the unwrapped cell edge, the contour change should be taken into account instead of solely relying on the max values.

Using the MRF GraphCut function from week 5 I was able to get decent segmentations of the nerves. Setting each pixels neighbours to the adjacent pixels in the pixels own slice, and the slice above and below. After normalizing the data I made a histogram of the whole .tiff which can be seen below.



From the histogram I set mu for the two classes to 0.3 and 0.55. Through a bit of trial and error I set beta to 0.0005.

I couldn't find a good way to plot the 3D segmentation in MATLAB, and did not look into ParaView. So I just plotted the first and last slice of the segmentation of get an idea of the quality of the segmentation. The segmentation of these slices can be seen below.



We can see that the big nerves are segmented well while the patches of smaller nerves are not segmented as well. Probably because of the darker background in these patches.

Week 7 - onepager - Deformable model

s164205

01/04 2020

Given the shape of the nerve cells, a strict Chan-Vese algorithm wasn't used, as the mean value inside the curve doesn't accurately represent the inside of what the curve will be bordering. Instead, intensity values are found manually and used as a constant.

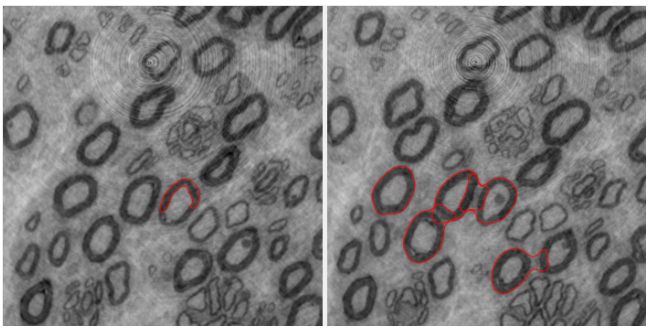


Figure 1: To the left, the curve 'imploding' on the axon. On the right, the curves including parts of respective bordering axons.

Solving this problem with one parameter set (per axon) turned out to be hard to fine tune. Choosing a high τ will result in the curve being to found of expanding to other axons as it moves up the nerve. Setting it low or increasing the smoothing makes the curve 'implode' i.e. form around the rim of the axon, instead of the whole structure.

Here, a measure was found, indicating the change of area created by the curve. When this measure exceeds a certain threshold, the algorithm is stopped. Were one to desire the full estimation of the axon, one would need to re-calibrate the parameters from this point.

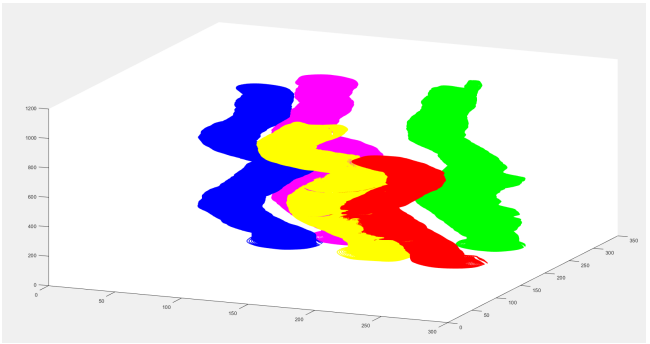


Figure 2: The final (badly plotted) estimation of a hand full of axons. Some are cut short, as the deformable curve no longer gives reliable information after a certain number of slices.

1 Introduction

This paper investigates volumetric segmentation of nerve cells by using deformable surfaces.

1.1 Data

The data in consideration for this exercise is a volumetric image of size 350 x 350 x 1024 of nerve fibres imaged using X-ray phase constrast zoom tomography.[1]

1.2 Method

The method deformable surfaces is used to segment the nerve fibres in each of the layers of the volumetric image. The objective of this method is minimize the segmentation energy.

2 Results

The inital segmentation of five of the nerve cells is seen in figure 1.

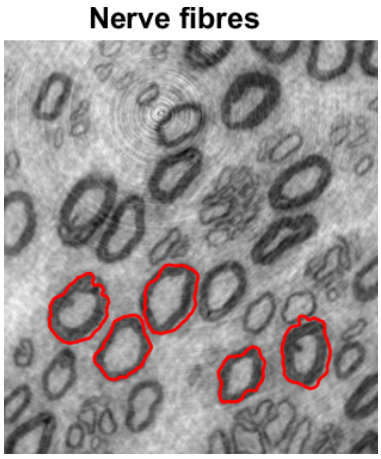


Figure 1: Intial Segmentation of five nerve cells in the first layer of the 3D image

As can be seen from figure 1 the shapes surrounding the nerve cells do not appear to be very smooth. The resulting segmentation of the volumetric image is depicted in figure 2.

[1]Dahl Bjorholm, Anders & Andersen Dahl, Vedrana "Advanced Image Analysis Selected Topics", pp. 59-61, March 25, 2020, DTU Compute

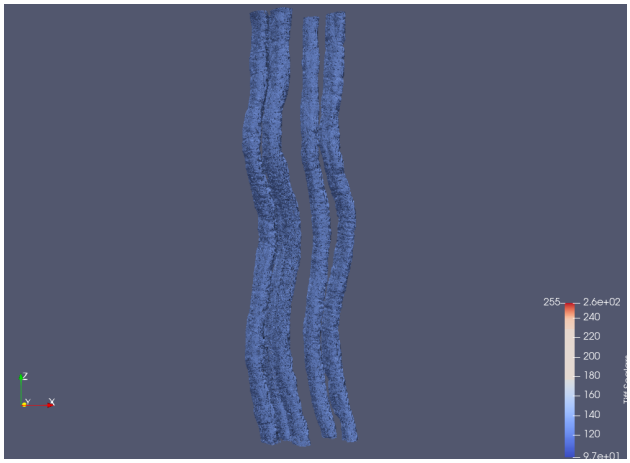


Figure 2: Segmentation of five nerve cells in a volumetric image. The five nerce cells are clearly seen as a group of 3 to the left and the pair of nerce cells to the right

2.1 Parameters

For the segmentation of the nerve fibres in the volumetric image using deformable surfaces the following parameters were used:

- For smoothing the curve, the following parameters were chosen: $\alpha = 3.5$, $\beta = 0.8$ and $\lambda = 1$
- Time step for displacement $\tau = 30$;
- Number of points = 157

3 Further improvements

For further improvements of the segmentation of nerve fibres in the volumetric image, the following can be optimized:

- Use of bilinear interpolation when finding the image intensity at snake coordinates, which are not integers.
- Increase of robustness by implemeting a feature such that the curve moves the point where the intensity in the normal direction is high. [2]

[2]Dahl Bjorholm, Anders & Andersen Dahl, Vedrana "Advanced Image Analysis Selected Topics", p. 61, March 25, 2020, DTU Compute

02506 Advanced Image Analysis - one pager

Nicolai Piet Dittmann (s170589), Mikkel Mathiasen (s174344)

March 2020

Markov Random Fields (MRF) is applied in order to segment nerve cells in a volumetric x-ray image.

A 3D implementation of MRF is applied, where neighbours in the plane and in z-direction, the nerves direction, are penalized if they are not from the same class. In order to make this approach a subset of the 1024 image slices are chosen (20-40 slices) and loaded in at once and flattened. Each neighbour in the z-direction are found through the image length and the coordinate in the given image.

The mean of the of the nerve cells and the background intensities are found through flattening the volumetric image and computing a histogram, as can be seen below.

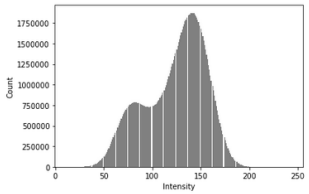


Figure 1: Histogram of image intensities

The mean of the background intensity is then computed as the highest peak in the histogram. The mean of the nerve cell intensities are computed as the second highest peak.

The slices have been normalized by dividing each image slice with the maximum pixel intensity in the given image slice.

In order to remove the small nerves, the intensity mean of the background and the nerve cells are fine-tuned. The mean of the background was lowered as this removed more of the small insignificant nerves.

The result of this segmentation strategy is shown in figure 2 below:

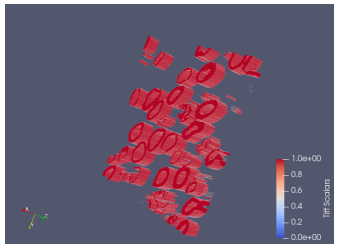


Figure 2: Caption

In order to optimize this segmentation more smoothing in the z-direction should be applied.

Markov Random Fields deliverable: Week 7

Frederik Hartmann, S174471

For this small deliverable, I worked with a binary segmentation of nerves in a 3D volumetric dataset. I predefined the mean-values of the myelin and background/axon (66 for myelin, 136 for background) from the histogram of the image. Those mean values I used to setup my s-t graph, by defining the weight from the source as the squared distance from the specific pixel-values in the image to the mean pixel-value of the background/axon. The sink weights were defined as the squared distance from the specific pixel values in the image to the mean pixel-value of the myelin. Those weights I use to define my terminal and internal matrices, that defines my s-t graph.

With this s-t graph set up, I used the graph-cut code given in week 5/6 (I can't remember), to find the optimal binary segmentation. The above procedure was done in every layer to get a 3D segmentation of the axons.

Below we see the results of doing the above.

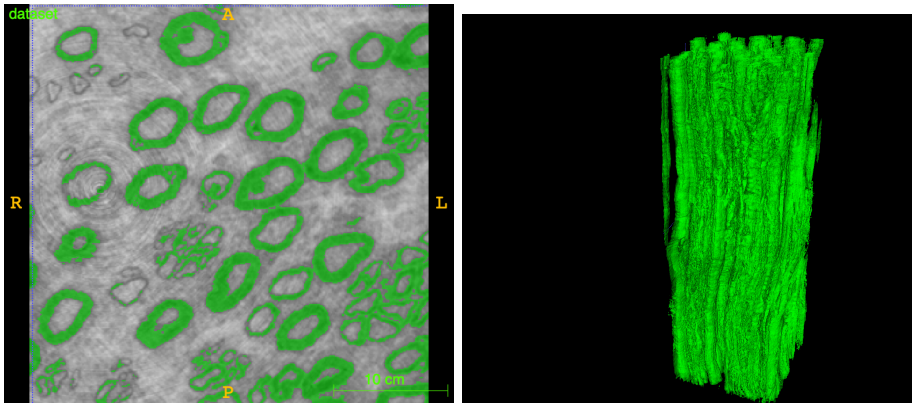


Figure 1: (Left) A binary segmentation of the first layer using the graph-cut method described above. To the right we see a 3D visualization of the binary segmentation done throughout the 3D dataset. It is hard to distinguish the individual axons from each other, since this method is binary (and not a multi-label approach).

As we see, the MRF problem solved with an s-t graph cut approach results in a pretty decent binary segmentation. However one might seek to use a multi label approach, since it is hard to distinguish the individual nerves from each other.

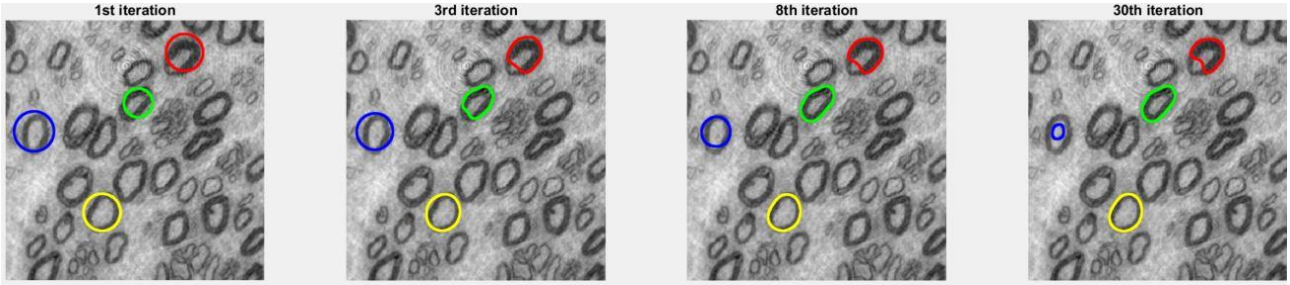
I refer to ITK-Snap as the 3D visualization tool used to create both figures in figure 1.

Deliverable, week 7

Jens Christian Bang Gribsvad, s174477

Deformable model approach

Below is illustrated my results of segmenting nerve cells by using deformable models in terms of snakes. Though the initial goal was to segment the nerves through a volume I only managed to segment the nerves in 2D:



From the above it can be seen the deformable model sort of works. The green and yellow snake succeeds in enclosing the myelin-boundary well while the red snake almost succeeds. Unfortunately, the blue snake doesn't seem to detect the axon at all as it just shrinks in and doesn't fold around the myelin. A difficulty of working with this data set is the change of myelin pixel intensity and the varying thickness of the myelin. As I have run my snakes on the same parameters this could explain the different outputs in detecting the axons.

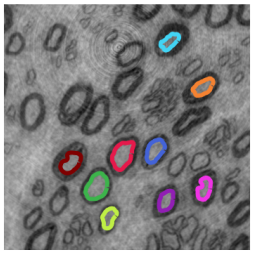
My approach:

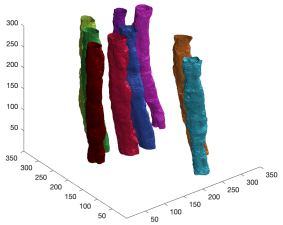
1. Determine the center of the initial circle for the snake along with the radius and number of points in the circle (I've chosen 200 points).
2. Compute the outward normal vectors for each point on the circle.
3. Locating the pixels inside and outside the circle and computing the mean value.
4. Creating an energy function consisting of the mean values and outward normal vectors (ie. Equation 6.1 in the notes).
5. Computing a regularization matrix (same procedure as from week 1).
6. Finally, the curve is updated by using equation 6.3 in the notes.

When working with this approach I found that the regularization parameters played a somewhat big role for how quick the snake would converge around the axon. I played around with it quite a bit and found the best parameters to be: $\alpha = 12$ and $\beta = 5$ (see page 12 for explanation about regularization).
Furthermore I just realized it might be a good idea in step 3 in my approach to just limit the outside pixels to a slightly larger area around the snake instead of take all the pixels in the in the image which is outside the snake. This adjustment could perhaps solve the issue with the blue snake as it would make the difference between the mean inside and mean outside larger.

Chan-Vese snake segmentation

For the Chan-Vese snake segmentation, the algorithm was initialized at the first layer. The two means of the inside (the bright inner axons) and the outside were set to $\mu_{\text{in}} = 0.4$ and $\mu_{\text{out}} = 0.3$ respectively in order to help the solution converge. The regularization matrix had the parameters $\alpha = 0.1$ and $\beta = 0.5$. Furthermore, the stepsize τ was set to 25.
The snakes were initialized on the first frame with 10 iterations in order to fit the inner axons, and then for each frame, one iteration was run with the snake from the previous frame used as starting point. This gave us the following segmentation of the nerves:



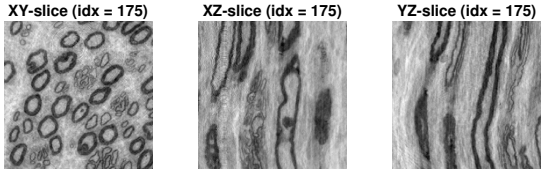


As seen above, we only included some of the axons since some of the snakes would have undesired behaviour i.e. "explode".

MRF segmentation

For the MRF segmentation, a full 3D MRF segmentation was implemented where each direction (x, y, and z) had its own smoothness prior parameter penalizing change in labels. This was done specifically to ensure increased smoothness of the segmentation along the z-axis which is the direction in which the nerves are elongated. The MRF segmentation problem was modelled as a directed graph and solved by finding the minimum cut in this graph.
Pixel values were used for the MRF segmentation rather than double values between 0 and 1. Due to this, the hyperparameters are fairly large: The smoothness prior parameters were set to 2200, 2200, and 10000 for the x-, y-, and z-direction respectively. The mean of the background (not nerves) and foreground (nerves) was set to 150 and 25 respectively in order to segment the darkest thickest nerves more easily.
Below, three center slices of the resulting MRF segmentation, one seen from each axis, have been shown alongside the corresponding original image slices. The 3D image was limited to a size of 350x350x350 in order to speed up computations. It can be seen that some work still needs to be done in order to make the segmentation really good. For instance, a median filter or morphological operations could be applied to the segmentation in order to remove noise and unclear nerves.

Week 7 - onepager Viktor Stenby (s174483) and Bjørn Dahl (s174480)



Danmarks Tekniske Universitet

02506 ADVANCED IMAGE ANALYSIS

Image Analysis with Geometric Priors

Nicolai Pleth	s174503
Mikkel Kofoed Pedersen	s174485
Lucas Alexander Sørensen	s174461

1 MRF Results

To get our MRF results for the nerves images, we used the template given, "gender_labelling.m". We read the first image of the nerves, and saved it as a 350x350 matrix with values between 0 and 1. We then chose our mean values as $\mu = [\mu_1 \quad \mu_2]^T = [0.44 \quad 0.27]^T$. These values were chosen to match the "background" and the myelin intensities. We then started checking for different values of prior term weight, β. We got our best result when choosing $\beta = 0.01$. This can be seen in the following figure:

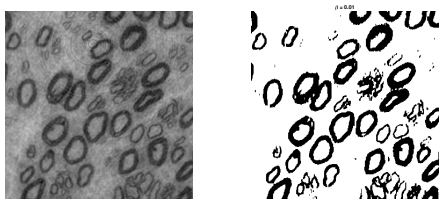
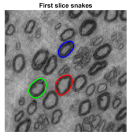


Figure 1: Visual result of MRF

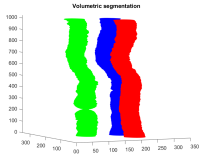
From this result we can directly get the myelin density, by summing up the number of pixels classified as myelin and divide this by the size of the image (350x350), for volumetric data, this should just be done for all images in the sequence. The remaining microstructural measurements would need further processing of the images, by use of a BLOB detection e.g.

2 Deformable surfaces

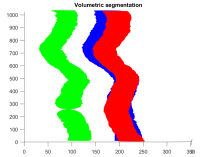
To get our deformable surface results, we chose three nerves to compute our snakes around. We computed the snakes individually, so the parameters could be different for each snake. As the form of the nerves is not the same for all images in the sequence, the parameters also needed to be changed for different sub-sequences. Through trial and error we found the parameters to give us the results shown in the following figure.



(a) Axons segmented using deformable curves visualized on a single slice.



(b) 3D visualization of axons segmented using deformable curves.

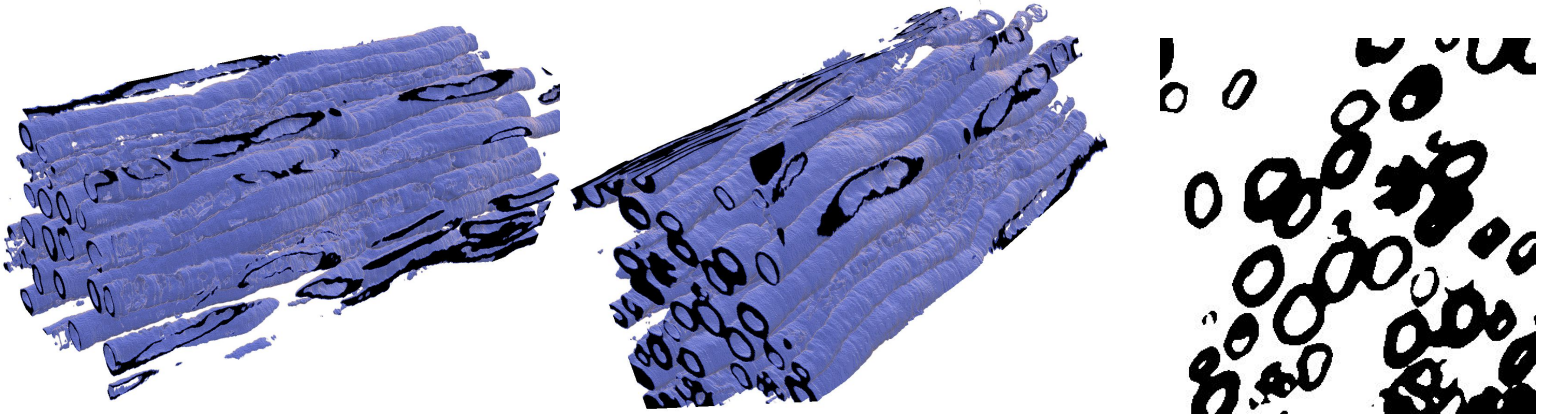


(c) Same 3D visualization of axons segmented, but from a different view point.

If we were to make deformable surfaces for all nerves, and make it on both the inside and outside of the myelin, we could get multiple microstructural measurements. We would have the nerve density count, as all nerves would be found. We could find the myelin density by use of the area covered in the rings formed between the inner and outer snakes. The average myelin could be calculated by use of the myelin density and nerves density count. The average axon density could be found by use of the area covered by the inner snake and the nerves density count. We could then approximate the average nerve radius, by assuming the nerves as perfect circles and using the average nerve area results.

02506 Advanced Image Analysis
Elika Araghi, Monika Frolcova

Binary segmentation of myelinated nerves using Markov random fields



Segmentation strategy

At first, we applied 2D gaussian smoothing to each slice to avoid modelling the small nerves which create clutter in the segmentation.
We estimated the mean pixel intensities of myelin and the background using a histogram of intensities.
Then for each slice, we used the MRF method and optimized it by building the s-t graph and applying the maximum flow algorithm to find the minimum cut to create the segmentation.
We have processed each slice independently and therefore we included only the neighbouring pixels within each slice in the smoothness prior.
After obtaining the segmentations, we exported the images as .png files and used ParaView software to create the 3D model.

Possible improvements

3D smoothness

We have modelled smoothness prior using neighbouring pixels within the slices. We could further improve our model by taking the fully 3D approach and including the neighbours from the previous and following slices in the z direction.

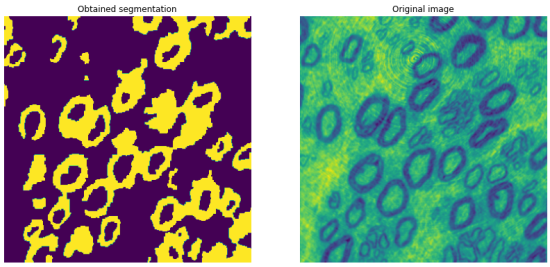
Segmentation of small nerves

We have used 2D gaussian smoothing on each slice to avoid modelling the smallest nerves. We could further improve the segmentation by applying other approaches such as changing the pixel intensities.
The differences in intensities of the foreground and background pixels could become more significant using non-linear transformation.

One-pager of Jules Belveze s182291

1 Markov Random Field results

One possible strategy to segment myelinated nerves is to use Markov Random Field. This will enable us to label each pixel as either background or stained myelin sheaths. To do so, I first plotted the pixel intensity distribution in order to estimate the mean intensity of each class μ_i. Afterwards, we formulate the likelihood energy and construct the corresponding graph: labels are the sink and the source, the weights between neighbouring pixels are assigned by using 2-clique potentials and the weights to the source and sink correspond the probability of the pixel to belong to each class. We show below the best obtained segmentation after tuning the smoothing parameter β and the two mean intensity values. We can see that



the obtained is a bit rough. It succeeds at identifying big isolated and cluster of small nerves but fails to separate nerves that are close to each other. In addition, the noisiness of the segmentation was limited by the factor β. The segmentation could probably be improved by removing the small nerves, which would prevent the formation of clutters. However, this segmentation can give a good approximation of the myelin density.

2 Deformable surfaces results

In this section we use a different approach for segmentation that uses deformable models. In order to obtain the desired segmentation we initialize a snake as a circle and iteratively deforms it with respect to both external and internal energies. The results when trying to segment 4 nerves are shown below, the left one depicting the best obtained segmentation. Even though, the results are promising it requires a lot of parameter tweaking. The right image illustrates what happen when the circle are initialized with a bigger radius. We can see that the deformable surfaces tend to wrap neighbouring cells. In opposition to the results obtained with MRF, this segmentation could be used to extract the average nerve area and the average nerve radius.

In addition, deforming the surfaces along the $z - axis$ also gave promising results with the right parameters.

Week7_MRF

March 31, 2020

1 Week 7 - Advanced Image Analysis

Authors: Nikolay Dobrev and David Vinje March 2020

1.1 1 Introduction

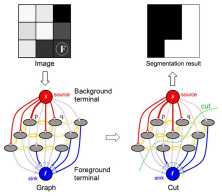
In the following exercise we will look at segmentation of myelinated axons basedon volumetric data. To achieve this we will use Markov Random Fields (MRF) optimised using a graph cut algorithm implemented in Python.

1.2 2 Methods

Optimizing a MRF can be done in different ways: manually trying out differ-ent labels configurations (not very efficient), using iterated conditional modes,Gibbs sampling algorithm, or, as we will use, a graph cut algorithm.

```
[1] from IPython.display import Image
    Image("1.jpeg",width=600)
```

[1]



The goal of the graph cut is to minimize the cost separating the source (also known as foreground) from the sink (or background), which in this case is separating myelin sheated axons from a background. The cost of one such cut equates to the posterior energy of the corresponding configuration.

1.3 3. Results

Four microstructural measurements were extracted from the volumetric data.

```
[2] import numpy as np
    import PIL
    import maxflow
    from scipy import ndimage
    from skimage import measure
    import matplotlib.pyplot as plt
    from scipy.signal import find_peaks
```

We now load N frames from the tiff image

```
[3] number_of_frames = 100
```



```
[4] img = PIL.Image.open('nerves_part.tiff')
    frame = np.array(img)
    img_frames = np.empty((number_of_frames,frame.shape[0],frame.shape[1]))

    for i in range(number_of_frames):
        img.seek(i)
        img_frames[i,:,:] = np.array(img)
```

We found the mean intensities so we can perform MRF segmentation

```
[5] uniq, counts = np.unique(frame, return_counts=True)
    counts, bins = np.histogram(frame.flatten(),len(uniq))
```

```
[6] mu = [53,133]
    labels = [1,2]
```

```
[7] beta = 1
    beta_s = 10
```

```
[8] single_image_length = len(frame.flatten())
    img_flatten = img_frames.flatten()
    N = len(img_flatten)
```

```
[9] G = maxflow.Graph[int](N,N)
    vertices = G.add_nodes(N)

    # create the edges between each neighboring and between the corresponding
    # vertices between the frames
    for z in range(number_of_frames):
        for i in range(single_image_length):
            if( i + 1 < single_image_length ):
                G.add_edge(vertices[z*i + 1],vertices[z*i + i+1],beta,beta)
            if( z + 1 < number_of_frames ):
                G.add_edge(vertices[z*single_image_length + i],
                           vertices[(z+1)*single_image_length + i],beta_z,beta_z)

    for i in range(N):
        G.add_tedge(vertices[i],img_flatten[i]-mu[0]**2,
                    img_flatten[i]-mu[1]**2)

    G.maxflow()
```

```
[9] -1.4187107
```

```
[10] segmentation = np.zeros(img_flatten.shape)
     for i in range(N):
         segmentation[i] = G.get_segment(vertices[i])
```

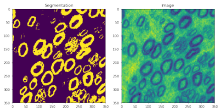
We now reshape back the segmentation to the original volume sizes

```
[11] first_segmented = np.array(segmentation[:single_image_length])
     first_segmented = first_segmented.reshape(frame.shape)
```



```
segmentation = segmentation.reshape((number_of_frames,
                                     frame.shape[0],frame.shape[1]))
```

```
[12] fig = plt.figure(figsize=(9,4.5))
     ax = fig.add_subplot(1,2,1)
     im = ax.imshow(first_segmented)
     ax.set_title("Segmentation")

     ax = fig.add_subplot(1,2,2)
     im = ax.imshow(frame)
     ax.set_title("Image")
     fig.tight_layout()
     plt.show()
```



1.3.1 1. Nerve density count: Number of axons per area of nerve-fibre crossection. Measured in number per area.

```
[13] def get_cutted_image(image,area):
         cutted_part = image
         if(area != 0):
             top_left = area[0]
             buttom_right = area[1]
             cutted_part = image[top_left[0]:buttom_right[0],top_left[1]:
         buttom_right[1]]
         return cutted_part
```

```
[14] def get_nerve_density_count(segmented_image,real_image,area):
         cutted_part = get_cutted_image(segmented_image,area)
```



```
    real_image = get_cutted_image(real_image,area)
    plt.imshow(cutted_part)
    plt.title("Cutted area")
    plt.show()

    filled_holes = ndimage.binary_fill_holes(cutted_part)
    plt.imshow(filled_holes)
    plt.title("We fill the axons")
    plt.show()

    nerves = real_image-filled_holes
    plt.imshow(nerves)
    plt.title("We segment only the nerves")
    plt.show()

    # we take only the axons and make the image binary
    axons = nerves[:-segmented_image]
    axons[axons > 1] = 1

    axon_area = (axons == 1).sum()

    props = measure.regionprops(measure.label(axons))

    area_threshold = 20

    valid_axons = []

    for prop in props:
        if(prop.area > area_threshold):
            valid_axons.append(prop)

    number_of_axons = len(valid_axons)

    plt.imshow(axons)
    plt.title("Binary axons with the centroids of the props found")
    for prop in valid_axons:
        plt.plot(prop.centroid[1],prop.centroid[0],'*')
    plt.show()

    return number_of_axons, axon_area
```


1.3.2 2. Myelin density: A fraction of nerve cross-section corresponding to myelin. Expressed as dimensionless fraction (a number between 0 and 1), or a percentage

```
[15] def get_myelin_area(segmented_image, area):
         cutted_part = get_cutted_image(segmented_image,area)
         num_of_myelin_pixels = (cutted_part == 1).sum()

         return num_of_myelin_pixels,cutted_part
```

```
[16] def get_myelin_density(segmented_image, area):

         num_of_myelin_pixels,cutted_part = get_myelin_area(segmented_image,area)
         num_of_all_pixels = len(cutted_part.flatten())
         return num_of_myelin_pixels/num_of_all_pixels
```

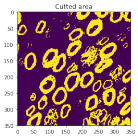
```
[17] top_left = (100,50)
     buttom_right = (300,250)
     area = [top_left,buttom_right]
     print("Myelin density in the given area is ",
           ,get_myelin_density(first_segmented,area))
```

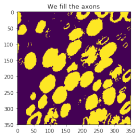
Myelin density in the given area is 0.34305

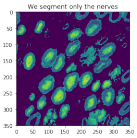
1.3.3 3. Average nerve area: Average area of nerves, broken down into average axon area, and average myelin area. This measure is very related to average nerve radius.

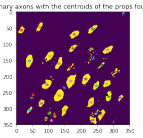
```
[18] def get_avg_nerve_area(segmented_image,real_image,area):
         myelin_area, cutted_part = get_myelin_area(segmented_image,area)


         number_of_axons, axon_area =
     ,get_nerve_density_count(segmented_image,real_image,area)

         avg_axon_size = axon_area/number_of_axons
         avg_myelin_size = myelin_area/number_of_axons

         return avg_axon_size,avg_myelin_size
```

```
[19] get_avg_nerve_area(first_segmented,frame,1)
```





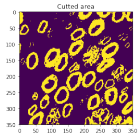


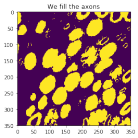

```
[19] (207.86486486486494, 950.9189189189)
```

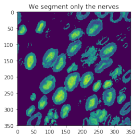

1.3.4 4. Average nerve radius: Average radius of nerves, broken down into average axon radius, and average myelin thickness. This measure is very related to average nerve area.

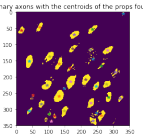
```
[20] def get_avg_nerve_radius(segmented_image,real_image,area):
         avg_axon_area, avg_myelin_area =
     ,get_avg_nerve_area(segmented_image,real_image,area)

         # A = pi * R
         avg_axon_radius = np.sqrt(avg_axon_area / np.pi)
         avg_myelin_wall_thickness = np.sqrt(avg_myelin_area/np.pi) - avg_axon_radius


         return ( avg_axon_radius, avg_myelin_wall_thickness )
```

```
[21] get_avg_nerve_radius(first_segmented,frame,1)
```










```
[21] (9.737213730493174, 7.460685360533336)
```

Volumetric Segmentation

s182820 // s182706

April 2020

Brief Description of Results

We have attempted to implement both binary segmentation using MRF and volumetric segmentation using deformable models. The results for binary segmentation can be seen in figure where we have shown three consecutive images from the image stack. This was obtained using a beta value of 30. It is clear that we do obtain segmentations of the myelin sheets, but we also observe quite a lot of noise in the segmentations.

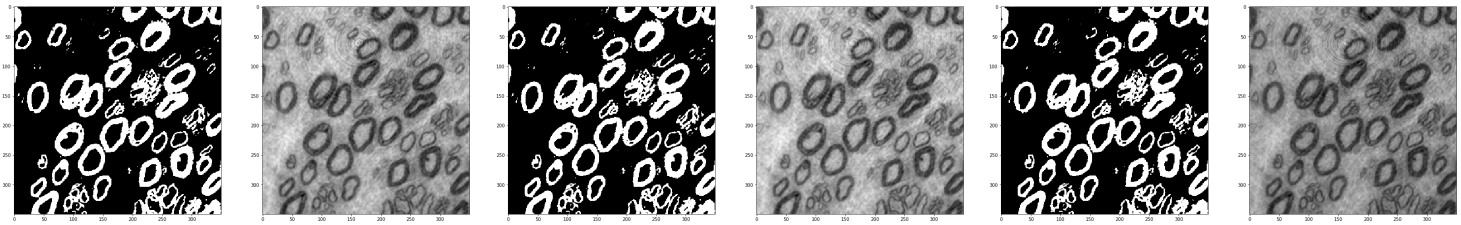


Figure 1: Three consecutive images from the image stack with their respective binary segmentations using MRF.

Hereafter we performed volumetric segmentation using deformable models.This was performed using fixed values for m_{in} and m_{out} estimated from the pictures. Note that the model is made as to sketch the inside of the myeline. To make this work the snakes has to be initialized with extreme care. The results can be seen in figure.

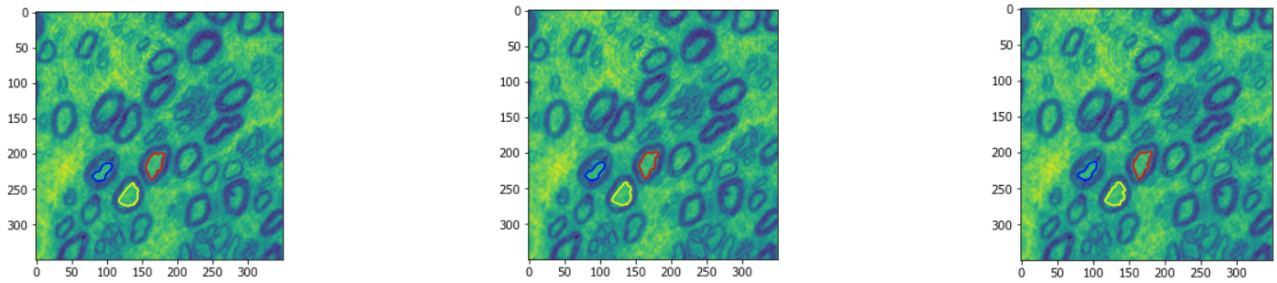


Figure 2: Three consecutive images from the image stack with their respective volumetric segmentations using deformable models.

Extraction of Microstructural measurements

We have not been able to extract the microstructural measurements within the timeframe, but here discuss how we would perform each of them:

- *Nerve density count* - We expect that nerve density count could be performed simply by counting the number of blobs in a cross section. We would set a threshold on blob size in order to remove artifacts occuring from noise.
- *Myelin Density and nerve area* - Using deformable models, we would count the white area (Myelin) within a volumetric segmentation and then divide the number of white pixels by the total number of pixels within the volumetric segmentation. The white pixels would be the area of the myelin, and the area of the axon would be the $total\_area - myelin\_area$.
- *Average nerve radius* - Average nerve radius can be found by doing scale space detection on each cross section as we did in exercise 2. We would first perform scale space detection to find the radius of a whole nerve. Hereafter we would perform scale space detection to find the radius of the axon. The thickness of the myelin is then $total\_radius - radius\_axon$.

Jakob Lønborg Christensen (s183985)
01/04-2020

Myelin and axon volumetric segmentation

I attempted 2 methods, the first being a Chan-Vese snake segmentation of the nerves. I chose to segment the bright inner axons as the snakes were too eager to segment multiple nerves if the darker myelin was included. Even with data transformations it seemed impossible to prevent this, and i would have to code no overlapping snakes to stop it. The results were decent for axon segmentation, however about half of the snakes lost their nerve and expanded. The ones that did not lose their nerve are plotted. I used means of mu_in = 0.4 and mu_out = 0.3.

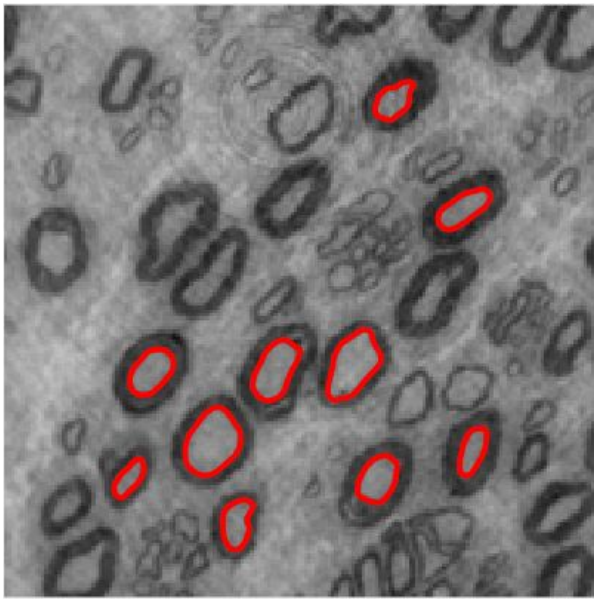


Figure 1: One layer of the Chan-Vese snakes that did not explode.

The second strategy was a 3D MRF segmentation which worked to visualize all the dark myelin but could not separate nerves that were too close. I used a higher beta value (0.01 compared to 0.002) in the Z-direction as the nerves were elongated in this direction. For MRF i used double pixel values of mu_1 = 0.48 and mu_2 = 0.24.

The data was downscaled for both these problems as it was quite expensive for my laptop.

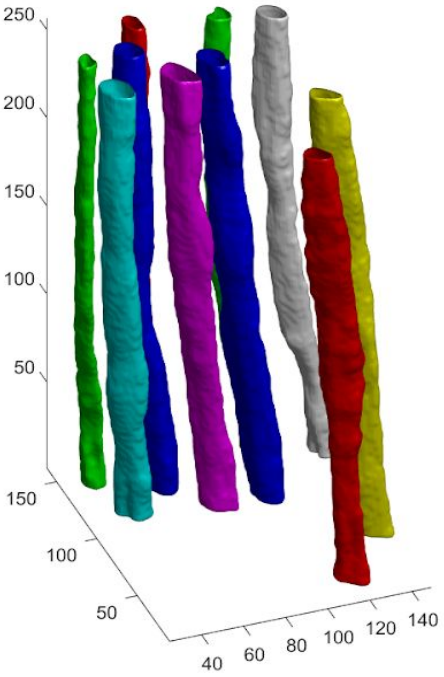


Figure 2: The 3-D segmentation of axons using Chan-Vese snakes.

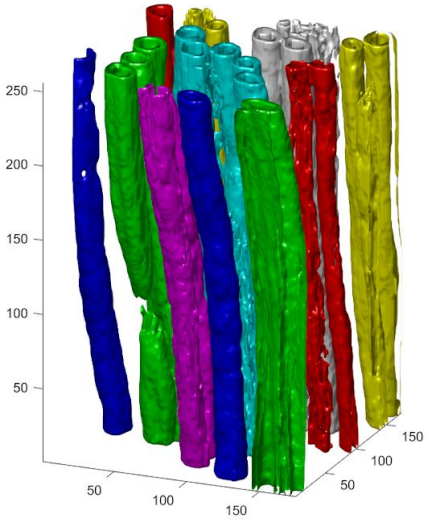


Figure 3: The 3-D segmentation of Myelin using MRF. Nerves of the same color were connected.

Deformable Models

Fabian Mager - s190212

March 2020

In the following, I will briefly describe my approach to the segmentation of nerves. The segmentation has been achieved using deformable models, referred to as "snakes". The snakes have been initialized using rotated ellipses, as the nerves have rather an elliptic than a round shape. First, the intensities to calculate the external forces have been hard coded. Looking at the normalized grey-scale image, the values have been set to $m_{In} = 0.4$ and $m_{Out} = 0.55$. A "inner snake", deforming along the inner myelin surface was then initialized. After deforming, the updated intensities have been calculated, using the area formed by the inner snake as m_{Out}, and the area formed by the outer snake minus the area formed by the inner snake as m_{In}. It is worth mentioning, that in order for the inner snake to "press" against the myelin ring, m_{In} and m_{Out} have to be switched for the calculation of F_{ext}. A regularization matrix B has been used for the deformation of snakes. It has been observed that the inner snake is needs to be less regularized, as it is restricted by the closed form of the myelin rings anyway. Setting the parameter for the outer snakes seems to be more difficult. The values for B are: $\alpha_{in,out} = 0.02$, $\beta_{in} = 0.02$ and $\beta_{out} = 0.11$. When nerves seem to "touch" each other, the outer snakes tend to overlap. This issue could not be avoided. However, the inner snakes seem to be more consistent. I used this property and restricted the outer snake not to overlap with the inner snake. This seemed to work quite well, as it was holding the outer snake in place, which eventually recovers as the nerves grow apart. I specified a margin between the outer and inner snake of three times the norm-vector perpendicular to the outer snakes.
The results as shown in the figures below show room for improvement. Similar to the restriction described above, one could restrict the outer snakes making overlapping between them impossible. Another way might be to set a maximum margin between the inner and outer snake. Furthermore, optimal values for B need to be investigated. The current choice is quite arbitrary, and found due to trial and error. Micro-structural properties were not included.

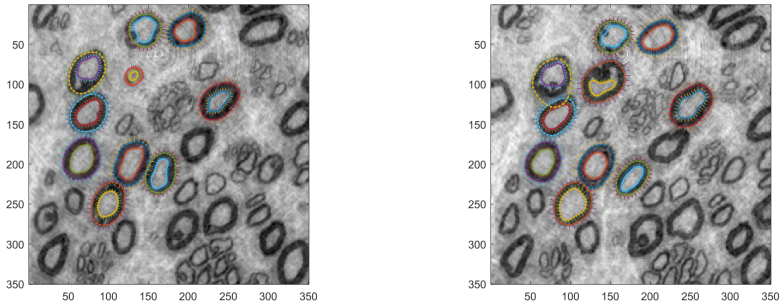


Figure 1: Segmentation of snakes, including norm-vectors perpendicular to the curvature. The image on the right shows the common problem of overlapping snakes.

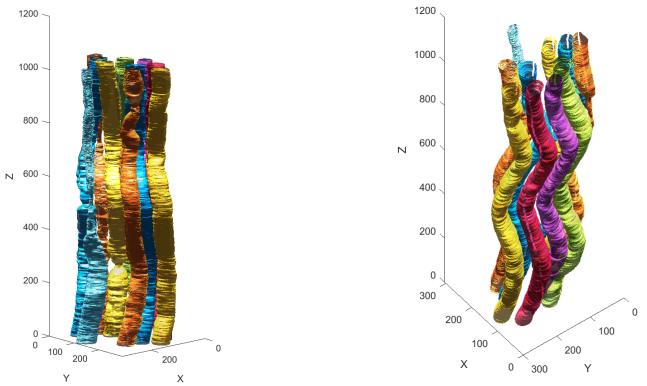


Figure 2: 3d view of segmentation results after all 1024 images.

Ionela Marinuta s190388
Platon Woxler s196660

Nerve Segmentation with MRF

For the task of segmenting the myelin sheaths on the provided nerve fibers we used MRF with graph cut optimization. We processed the volume slice-by-slice, without smoothness in the *z* direction.

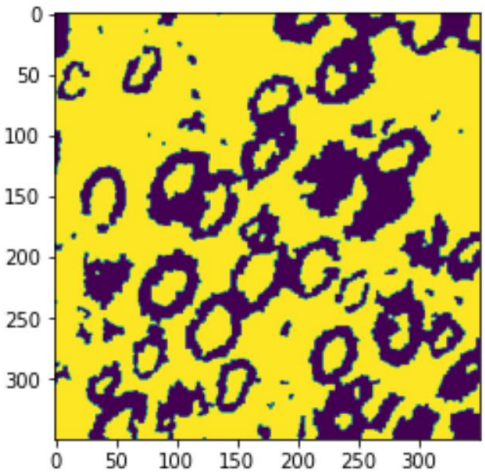


Figure 1 Segmentation of single slice

Before starting the segmentation we plotted the histograms of a selection of slices to see how well the myelin intensities are separated from the surrounding intensities. We visually determined that the mean intensities do not vary highly along the slices. We discussed implementing some kind of peak detection of the histograms so that the means could be calculated for every slice, but ultimately decided to use hardcoded values.

We tried a few Beta penalization values for the MRF algorithm and settled on 100 as it gave the best result in terms of maxflow as well as the most satisfactory visual inspection. Another thing we tried was to increase the contrast of the images before segmenting, however that yielded poorer results that also reflected in the maxflow capacity.

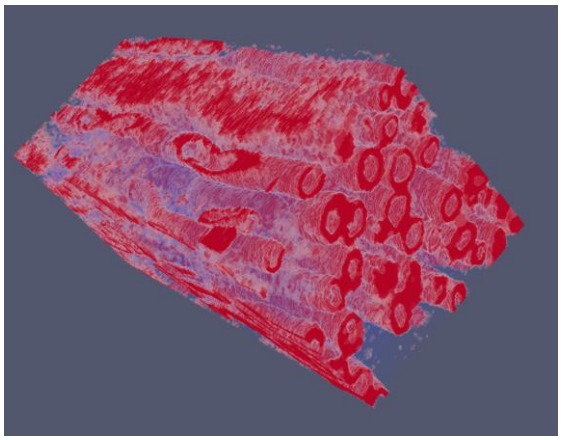


Figure 2 Segmentation with MRF

Trying to show a 3D-view of the data gave us pretty poor results due to the picture containing a lot of noise. It still shows some potential though and some of the visualization “problems” might be solved by just knowing one’s way around the visualization tool.

Interestingly, the clearest segmentation we got came from simply thresholding the image on intensity 100, which can be seen in figure 3. This would probably not work on just any data but happened to give a better segmentation in this case.

Future improvements:

- compute mean intensities for each slice
- smooth along the z-axis
- try ICM optimization

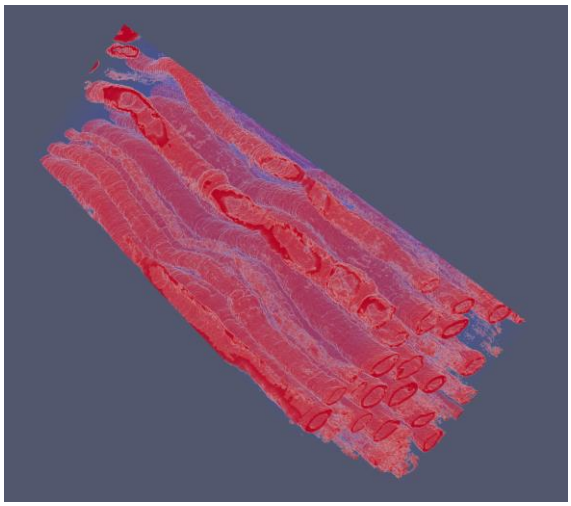


Figure 3 Segmentation using thresholding

MRF result on nerves

Paul Senty (192701)

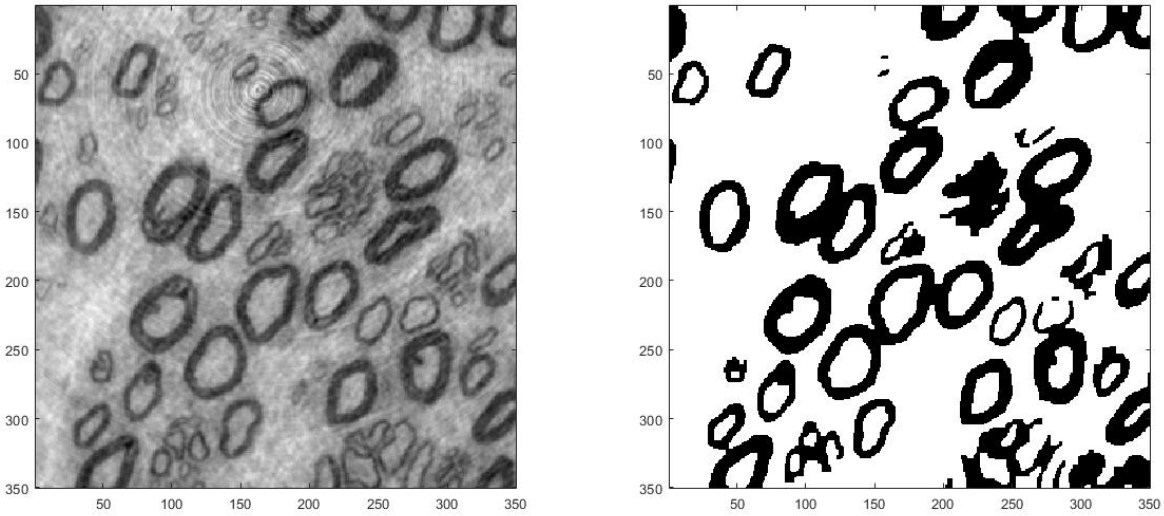
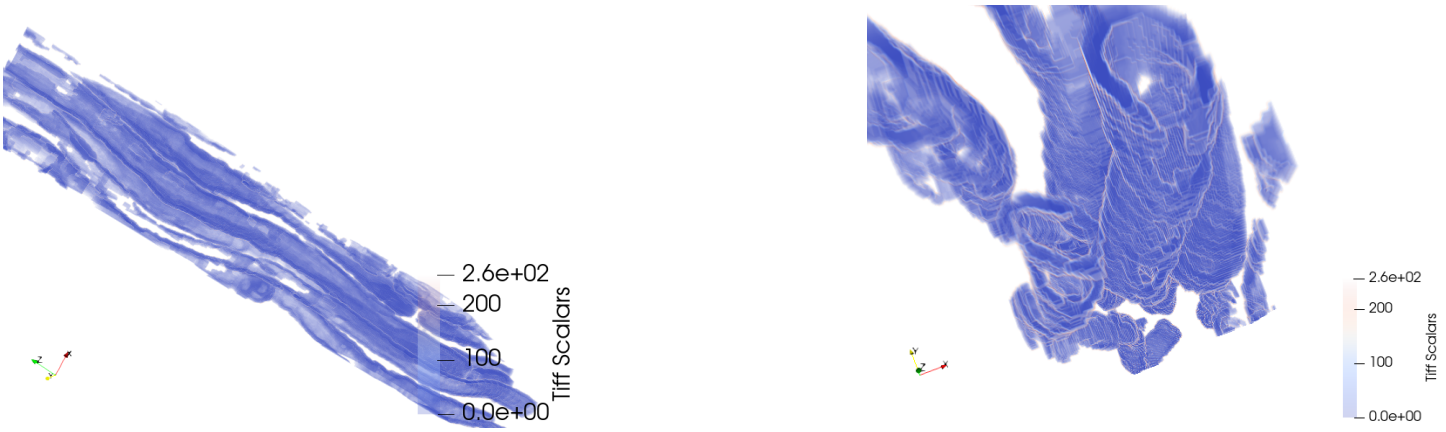


Figure 1. Slice visualization of the original image and the segmentation



(a) View from side **(b)** View from above

Figure 2. 3D visualization of the nerves segmentation on Paraview

These results were obtained by using Markov random fields or, more precisely, through a graph cut t-s algorithm. I assigned each pixel to one of the folowing two classes : myelin or non-myelin.

The image histogram is bimodal, the first peak can be interpreted (appoximated) as the mean intensity of the myelin area and the second (and higher peak) as the mean of the rest of the image. I defined the distance of each pixel to both of these classes as the difference in intensity between the pixel and the peaks (it could be improved by considering approximating the histogram as a sum of two Gaussian distributions with different standard deviation and then take the deviation into account to compute the distance).

For the s-t graph-cut algorithm, we need to define two matrices : the terminal energy and the internal energy. The terminal energy matrix can easily be deduced from the distances explained previously. The internal energy links each pixel in three dimensions to its 6 nearest neighbors (the value of the link β is the same for all the neighbors and is adjusted to have a good trade-off between internal and external contributions). We see easily on the slice that the algorithm struggles to identify groups of thin nerves as distinct nerves as the myelin area is not necessarily closed. We could add constraints (new cliques) to force it to be closed and define a more subtle terminal energy (distance to local peaks on the histogram instead of global peaks for instance).

DTU 02506 - Segmentation of myelinated nerves

Miklós Kristóf Jásdi - s192748

March 31, 2020

1 Problem statement

Modern technologies, such as X-ray tomography have given us the ability to produce volumetric images of human tissue. One potential use of the aforementioned technology is the analysis of the axons of diabetes-affected peripheral neurons. While researchers have the tools to extract the 3D image of the given tissue, they are particularly interested in the radius, trajectory and organization of axons, that can not be directly inferred from the raw volumetric scan. In this short report I represent an image processing approach to extract the segmentation of these nerves for easier analysis, based on an example volume shown in Figure 1.

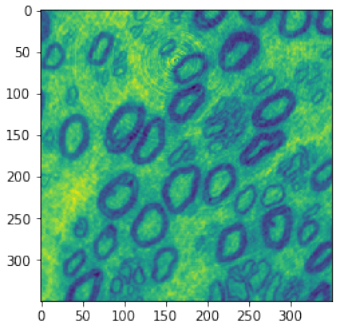


Figure 1: The example volume at Z=0.

2 Proposed solution

There are multiple approaches to this problem, such as Deformable Models, or Markov Random Fields (MRF). I have chosen the latter technique, as it requires less manual initialization to yield acceptable results. The MRF technique involves the representation of the segmentation task as an optimization problem, where each pixel and its neighborhood is assigned a certain amount of energy. It is up to us to define a suitable energy function for the given domain. In image segmentation, it comprises the prior energy and likelihood energy. The former represents the difference between a given pixel and its neighborhood, while the latter represents how far the given pixel's intensity is from the mean intensities (μ) that characterize the segmentation classes. We can obtain the best segmentation by minimizing this energy function.

There are multiple methods that can accomplish this, but for our two-class (nerve and background) segmentation task, I have used the so-called max-flow min-cut algorithm. As an input for this algorithm, the image can be modelled as a graph, where each of the nodes represent a pixel. Each node is connected to all other nodes representing its X, Y and Z pixel neighbors. The weight of these edges (β) denote the prior term. Each node is also linked to a distinct source and terminal node, via edges that represent the likelihood term. The algorithm yields an s-t graph cut with the minimum edge weights, representing a segmentation with the optimal energy.

Due to my limited access to computing power, I had to downscale the volume by a factor of 2 along the X and Y axes, and take a section between Z=0 and Z=300. After several test runs, I have settled at $\beta = 500$, $\mu(myelin) = 50$, $\mu(background) = 140$ to obtain the final segmentation shown in Figure 2.

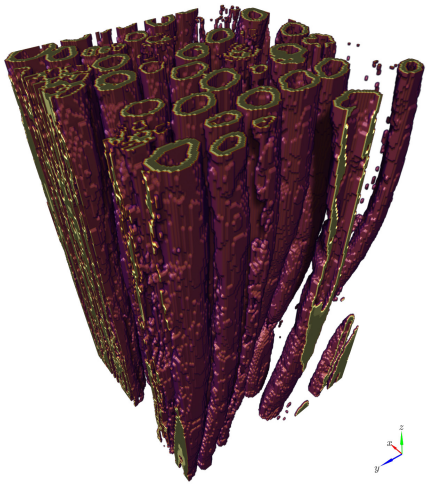


Figure 2: The completed segmentation.

Volumetric Segmentation of nerves using deformable model

Xiaoran Ma

(I)

The deformable model function that I define has the following format:

deform(D,img,iter_time,tau,alpha,beta)
returns the updated coordinates of points

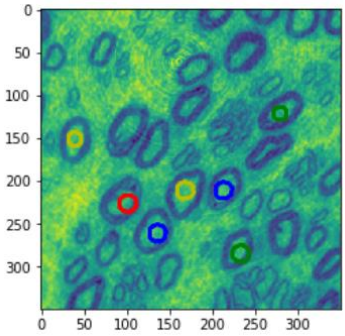
·The meaning of parameters of the function:

D:
The initialized coordinates of the points.
img:
The target image
iter_time:
The iteration time of deform algorithm
tau:
parameter of model which indicates the step of updating points.
alpha & beta:
parameter of the smoothing process

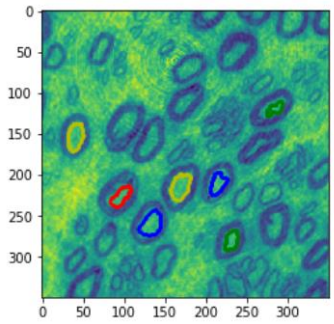
(II)

Steps for building a 3D visualization of nerve cells

① **Initializing points which is close to the center of a single cell**



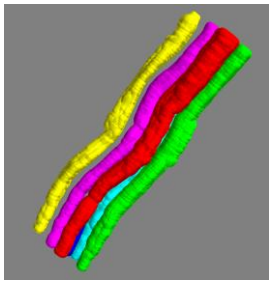
② **Conducting the iteration of deformable model by 300 steps on the first frame. (The iteration time is relatively large to reach a good result because it's the first frame, in which the initialized points are in shape of circles)**

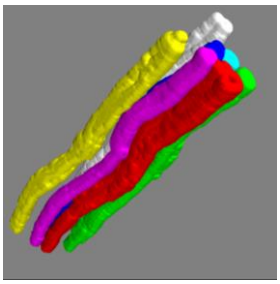


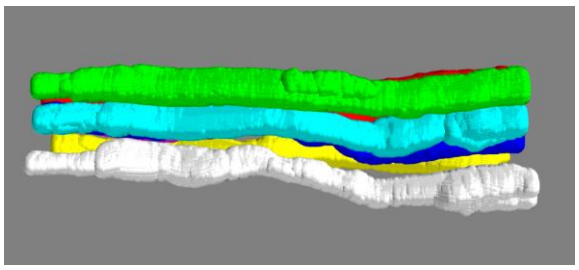
③ **Reading in the next frame, using the points of the former frame as the initial points of the new frame. (The iteration time is set to 10 because there is just slight change between two neighbor frames)**
Adding the result of the returned updated coordinates of points to a 3D array.

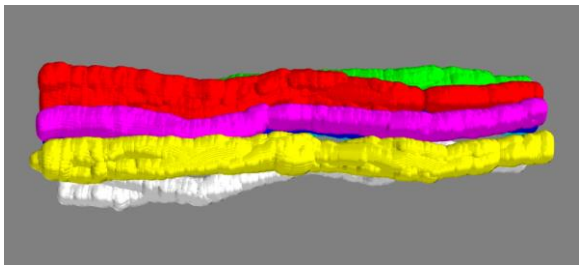
④ **Repeating the third step frame by frame until it reaches the last frame, and all the results are saved in the 3D array.**

⑤ **Visualizing the results which are saved in the 3D array by using a library called mayavi.**









Week 7 - Minireport

Joscha Erbis 22/03/2020

Method

In this report, I explore the adaptability of a 3D-MRF-implementation for the provided 3D-nerve-image. The goal is to achive the best possible segmentation and to illustrate the impact of different parameters of the algorithm. In order to increase the distinguishability between the nerves and the background, the contrast of every slice was increased using the `imadjust`-function in MATLAB. To reduce complexity, only a subset of slices was used (every fourth slice, 20 slices in total). The 3D-implementation was done by including neighbourhood across slices (pixel same location in next slice) into the 2-clique-potential in the prior-energy. Different weights are used for in-slice and slice-to-slice neighbours. This leaves us with four parameters to tune: The mean intensities of the foreground and the background μ, the in-slice-neighbour-weight β_s and the slice-to-slice weight β_z.

Results

Different mean intensities

The first sequence in Fig. 1 shows the first frame with different values for the mean-intensities of the fore- and background. For visualization, the segmentation is plotted in white on top of the original image. As it can be seen in the first row, reducing the foreground-intensity to lower values leads to a cleaner segmentation of the bigger nerves. Segmenting smaller nerves as well leads to high levels of noise and to bigger nerves growing together. After a good foreground level is found, the segmentation is fine-tuned with the background intensity in the second row. As it is highlighted with the red ellipsoids, varying this parameter can help in reducing the number of grown-together segmentations.

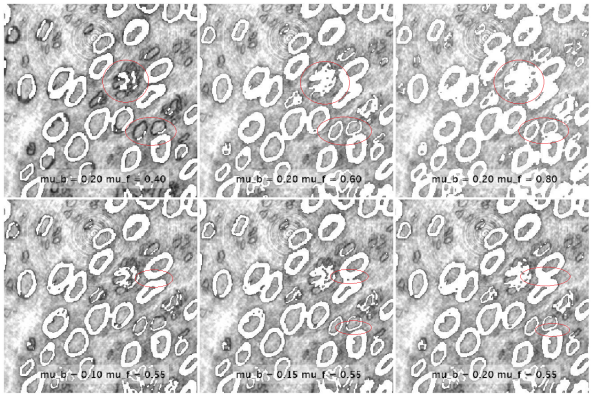


Figure 1: First frame with different combinations of the mean intensities. Better segmentation with less noise from smaller nerves is obtained with smaller values.

Different β_z-values

The impact of the slice-to-slice weight β_z becomes apparent in the sequence in Fig. 2. The first row shows the first, middle and last slice of the set segmentat with a low β_z-value of 0.05 and is compared to a segmantation with a higher value of 0.2 in the second row. A high value punishes changes in a pixel´s class between slices, so the segmentation will react slower to changes in the image. This can be seen in the red marked areas in the last frame. In the second row, the underlying myelin becomes visible as a dark shadow

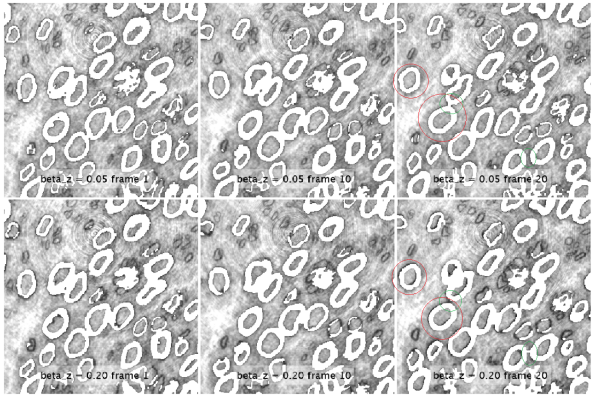


Figure 2: First frame with different combinations of the mean intensities. Better segmentations with less noise from smaller nerves are obtained with smaller values.

because the segmented shape does not follow the movement between the slices. No general statements can be made whether a smaller or higher value is better to prevent the growing together of segmented nerves since in both rows examples of nerves growing together can be found (see green markers).

Conclusion

With the MRF-segmentation-framework, a relatively well segmentation can be achieved. By extending the implementation to 3D, the segmentation can be further improved. However, segmenting both the smaller and bigger nerves without them growing together is impossible. It is believed that even better results can be obtained by implementing different energies which not only consider the pixel intensities but the local image structure as well.