Semi-automatic detection of vesicles in 3d FIBSEM data

Mariuta Nicolae

Synopsis

1. Working title: Semi-automatic detection of vesicles in 3d FIBSEM data

2. Abstract: For this project it has been used a dataset of images obtained using 3d FIBSEM over a part of rat's brain cells. The purpose is to detect the shape of a vesicle in 3d by clicking in the middle of it in one of the 2d slices. The snake algorithm will be mainly tried in different ways, by applying it on several 2d planes of change it to work in 3d. Other algorithms like region filling will be considered.

3. Introduction: For the research of how the information is transmitted between two neurons, it is very important to know what the shape of the vesicles is during the propagation of signal. The 3d FIBSEM was used to obtain a 3d image from the rat's brain and the aim of the project is to detect the shape of the vesicles as accurate as possible.

4. Problem analysis: For the project will be used a part of the dataset found on the website of Computer Vision Laboratory (http://cvlab.epfl.ch/data/em). The dataset consists of several 2d images which form a 3d block from part of the rat brain. For the processing of images will be used the snake algorithm because it was used in previous project over the same dataset and had some good results.

5. Objectives: The objectives of the project is to detect the shape of vesicles in 3d as good as possible and for these will be considered several combinations of parameters for the processing algorithms and apply those algorithms in different ways: use the 3d version of algorithms, apply them in 2d on several vertical and horizontal sliced to find how the best results are obtained.

6. Hypotheses: The reason for this project is to find if the vesicles do not have shape of sphere as it was believed longer time ago. For this reason, the shape that is detected will be approximated to an ellipsoid. If the shape of ellipsoid is not close to the shape of sphere, then the assumption will be correct.

7. Limitations: The whole dataset is very large and hard to load it into memory and make the application run fast, and for this reason was considered only a small part of the original dataset, but the algorithm should be adaptable to make and application that to load all the images in dataset.

Because the vesicles are more or less visible and defined boundaries it is almost certain that the application will sometimes fail at detecting some of the vesicles, but it will be changed and adapted so it can work in most of the cases.

8. Methodology and methods: For the implementation of algorithm and making experiments it will be used Matlab as environment for working. For being able to find easier the best results, functions from the Matlab library and external libraries will be used for processing and visualizing the results.

Contents

1. Vesicles

Synaptic vesicles are particles from neuron which are very important for propagating the nerve neurons. The vesicles store neurotransmitters that are released regularly by a voltage-dependent calcium channel. The vesicles are located in the axon area named “button”. [6]

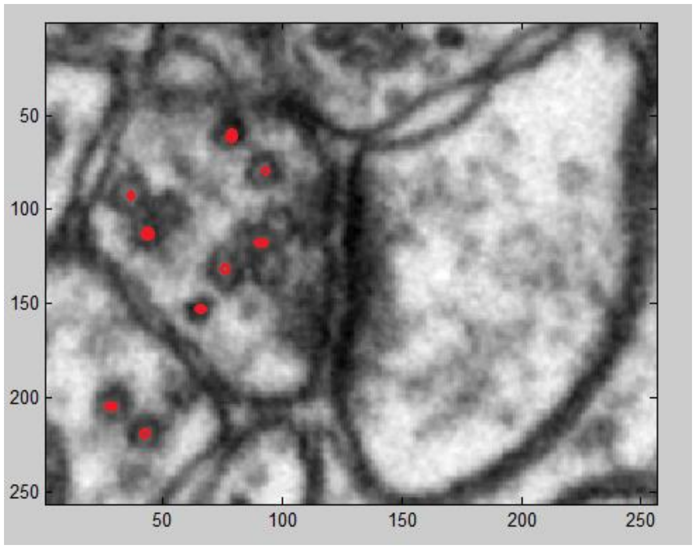


Fig. 1 Image from dataset representing contact area between two neurons and vesicles which have pointed with red circles

In the image above some vesicles are shown in a sample taken from the dataset that is used in the project. The darker area in the middle is the contact area between two neurons at the moment when transmission of an impulse is happening.

It is believed that the vesicles are changing their shape during the transmission of impulse so the purpose of the project is to detect the 3d shape of the vesicle along the consecutive images. The dataset contains several neurons out of which some were active and others were not active at the moment when the rat died so being able to find the shape for each of these vesicles would help very much to find if the assumption is correct.

Another thing which can be noticed in the image is that the boundaries of the vesicles are not very clear which will make it harder to find their shape very accurately.

2. Active contour model(Snakes)

Active contour model is a technique to find the objects outline which is efficient for noisy 2D images. The snake model is used in many applications like object tracking shape recognition, segmentation and edge detection. [7]

Snakes is a technique that matches a deformable model to an image by means of energy minimization. The snake itself is a deformable curve influenced by constraint and image forces that pull it towards the contours of object.

The main disadvantage of this method is that it requires that to have information about the shape of the contour before applying the algorithm. But it is very suited to this application which requires interaction with the user that has to click inside the vesicle so the starting point for the snake is known. An approximate for the final shape of the snake is also know(the vesicle is sphere) so finding the shape for the initial circle is easy to do.[7]

In comparison to other feature attraction techniques, snakes have several advantages:

- They autonomously and adaptively search for a minimum state
- External image forces act upon the snake in an intuitive manner
- Incorporation Gaussian smoothing in the image energy function introduces scale sensitivity
- They can be used to track dynamic objects[7]

The snake algorithm consists of several steps:

- Initial curve is created with shape of ellipsoid, a closed contour which will start growing until it finds area with high energy
- Image is convolved with Gaussian for smoothing and then it is calculated the gradient representing the external energy. Smoothing with Gaussian has the purpose to increase the energy of edges in the image.
- Read the value of the external force at the snake control points and update the position of the control points according to the gradient descent equation that was derived

- This is done in a loop with several iterations. In the current application it was chosen 200 iterations because that offered the best results. If the number of iterations is too low then the snake does not grow enough to detect the high energy edges and if the number of iterations is too high then the computation time is too long[7]

The position of the snake is represented parametrically by $v(s) = (x(s), y(s))$ and its energy functional is written[8]:

$$E^{*}_{snake} = \int_0^1 E_{snake}\big(v(s)\big)ds = \int_0^1 E_{int}\big(v(s)\big) + E_{image}(v(s)) + E_{con}(v(s))ds \qquad (1)$$

Where E_{int} represent internal image of the spline due to bending, E_{image} gives rise to the image forces and E_{con} gives rise to external constraint forces[8].

The internal spline energy can be written as:

$$E_{int} = (\alpha(s)|v_s(s)|^2 + \beta(s)|v_{ss}(s)|^2)/2 \qquad (2)$$

The spline energy is composed of first-order term controlled by $\alpha(s)$ and the second-order term controlled by $\beta(s)$. The first-order term makes the snake act like a membrane and the second-order term makes the snake act like a thin plate. Adjusting the weights $\alpha(s)$ and $\beta(s)$ controls the relative importance of the membrane and thin-plate terms[8].

The total image energy can be expressed as a weighted combination of three energy functionals:

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term} \qquad (3)$$

- The image energy intensity is the simplest useful functional. If we set $E_{line} = I(x, y)$ then depending on the sign of w_{line} the snake will be attracted either to light lines or dark lines.
- If we set $E_{edge} = -|\nabla I(x, y)|^2$ then the snake is attracted to contours with large image gradients
- The curvature of level lines in a slightly smoothed image is used to find terminations of line segments and corners combining E_{edge} and E_{term} we create a snake that is attracted to edges or terminations[8]

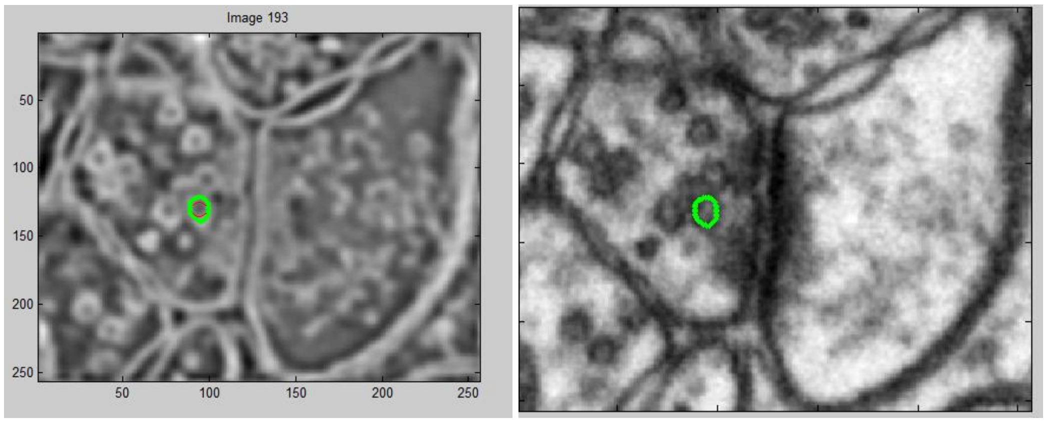


Fig. 2 First image contains the smoothed image with starting red snake and the green one at the current iteration in the algorithm and the second image shows the final result for snake drawn over the edges of vesicle

In the images above it is shown that the snake algorithm actually works in approximating the contour of the vesicles. In the first image it is shown the result for smoothing the original image which has the purpose to increase the energy of the boundaries. The snake is starting in the middle of the vesicle (the red circle) and it grows towards the exterior until it finds the boundaries with high energy and stabilizes. In the second image it is visible that the final snake is placed above the boundaries of vesicle.

The main challenge for the project is to find a way to make this algorithm to work at finding the boundaries of 3d vesicle. For this it will be tried to apply the vesicle over the 3 orthogonal directions or on several consecutive images until the vesicle ends.

3. Fit ellipsoid

As described previously, the points on curves detected by the snake algorithm are approximated with an ellipse that is also displayed in the 3d orthogonal slices viewer.   An example for applying this to a collection of 2d points is displayed bellow and is visible that most of the points are inside the ellipsoid. If these happen to be points obtained from applying the snake algorithms then the errors caused by noisiness of the image are ignored.

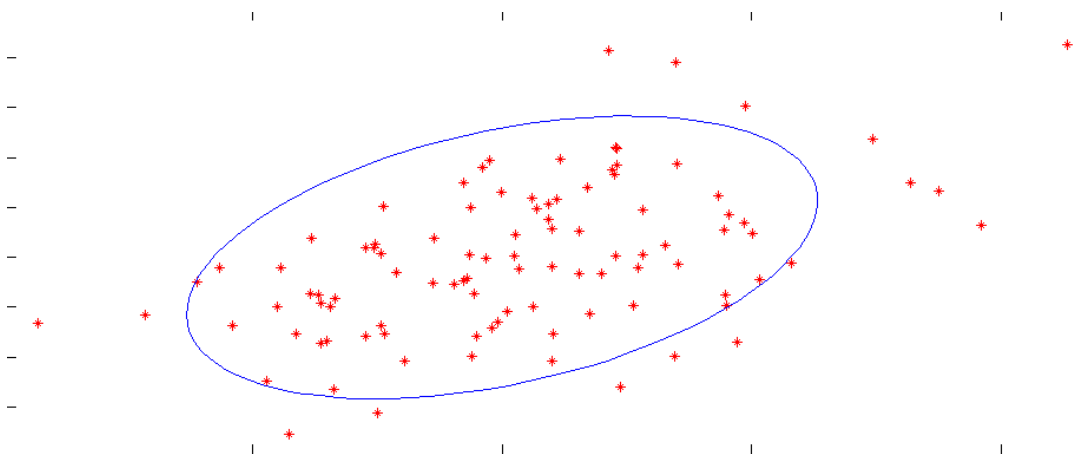


Fig.3 Examples of array of points in 2D that are approximated to an ellipsoid by using covariance matrix and mean if the points.

For calculating the ellipsoid it was used the algorithm and code described by Rajiv Singh [] which were adapted to make a function that fits 3d ellipsoid to an array if points and it contains the following steps[3]:

1. The covariance matrix is calculated for the points that are given[4]:

   $$\sigma(X,Y,Z) = E[(X - E(X)][Y - E(Y)][E - E(Z)]$$

   where E(X) is the expected value of X which is also known as mean of X []

2. The mean (average of the points from the given set) is also calculated
3. The eigenvalue and eigenvector is calculated based on the covariance matrix.
   a. The eigenvector is a non-zero vector *v* that when multiplied with square matrix *A* is a non-zero vector, yields a scalar multiple of itself (α):

      $$Av = \alpha v$$

   b. The number α is called the eigenvalue or characteristic value of A corresponding to *v*. In the current algorithm, A is actually the covariance matrix obtained in the step from above.[5]
4. The size of ellipsoid is calculated by using the equation:

$$S = N * \sqrt{E}$$

Where N is the standard deviation and E is the diagonal of eigenvalues. The result S is a vector with 3 elements: the sizes over the 3 orthogonal axes for the ellipsoid. The choice of standard deviation is important because it influences the size of ellipsoid which has to be accurate. Experiments with choosing different sizes for the ellipsoid will be done to ensure that the correct standard deviation is determined.

5. Now that the size of ellipsoid is determined, it has to be angled correctly according to the distribution of points in space. This is determined by multiplying the eigenvector with the coordinates of points from the edges of ellipsoid. The values for the coordinates that are obtained are summed with the mean value of the points and the rotated ellipsoid is obtained.[3]

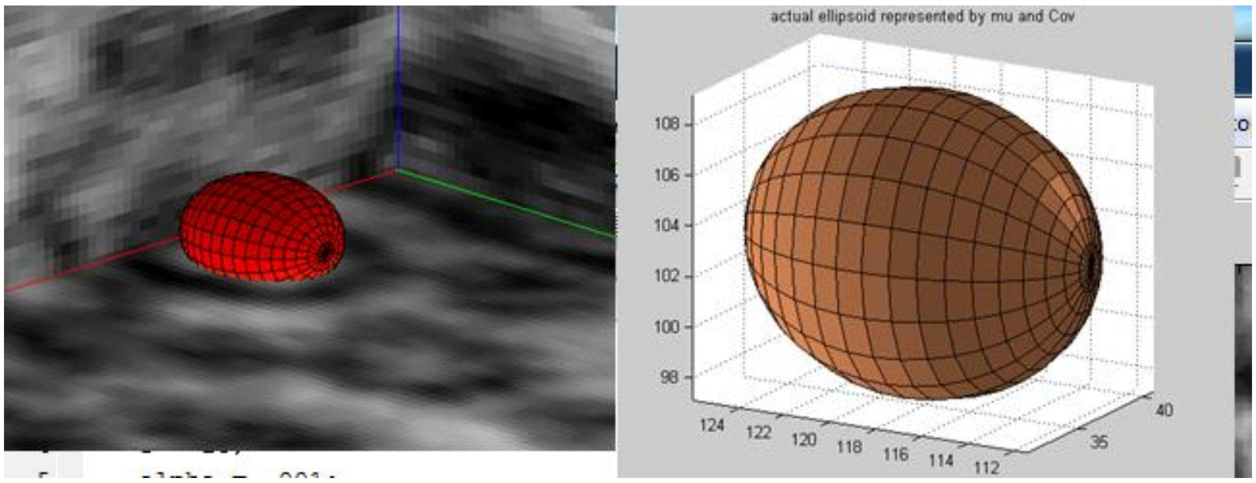


Fig. 4 Display of the ellipsoid that is obtained from the algorithm described above, by using the points obtained from the snake algorithm applied inside of a vesicle. The ellipsoid is also drawn separately from the dataset

As visible in the image above, after calculating the ellipsoid for the points on the curves that are obtained by the snake algorithm it is displayed on an orthogonal slices viewer which was created by Laszlo Balkay [1] and it helps with visualizing how well the ellipsoid fits inside the vesicles that have to be detected. The ellipsoid is also displayed on a separate 3d image viewer to visualize its shape.

This method for calculating the ellipsoid using covariance proved to be a good tool into analyzing the efficiency of the algorithms that are created and displaying some results.

4. Algorithm 1: apply snakes over the three orthogonal directions

As the first algorithm for using the snake algorithm to detect the 3d shape of vesicles, it was considered to apply the snake over the 3 orthogonal planes that converge into the point that is chosen by the user.

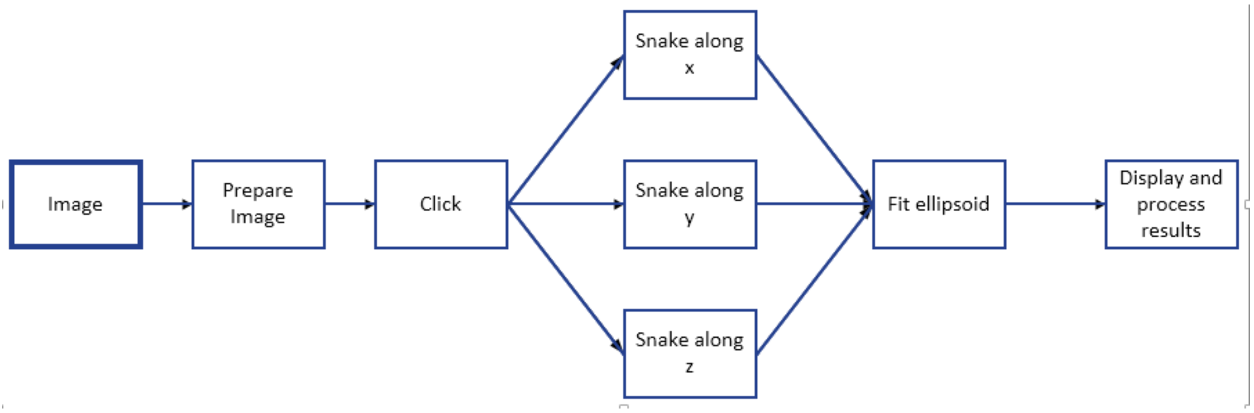


Fig.5 Diagram showing how the algorithm works

In the diagram above it is shown how the algorithm is implemented:

- In the first step the images are loaded from the file and the 3d matrix is created
- The image block is prepared by calculating image energy. This step is done by calculating the Laplacian for the 3d image. The Fast-Fourier Transform is calculated and then the second order derivative for all the dimensions (x,y,z) are calculated and the Gaussian is also convolved for smoothing. All these were done by using the function *scalen* which was created by Jon Sporring. As it is visible in the image from below, the high energy edges are very bright (have high energy value) while the areas in the middle of vesicle are black (have negative energy value)

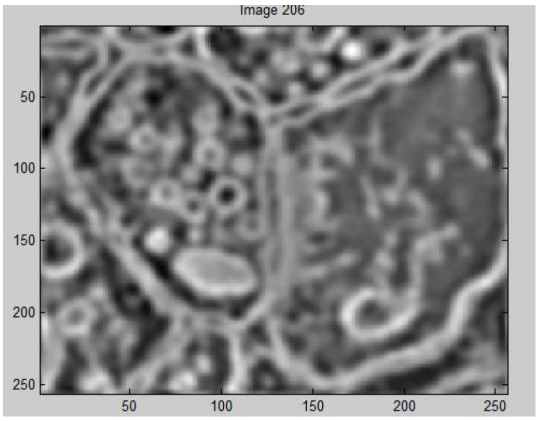


Fig.6 One slice in the dataset after the image energy was calculated for the whole 3d block

- The next step is that the user clicks inside of a vesicle. This step is important because, as described at chapter about Snakes it provides initial information so that the detection of vesicle boundaries can start. The result for this step is the coordinates (x,y,z) which are used as middle for the initial curve of the snake.
- In this step, the 3 orthogonal planes that are converging from the point that was clicked and snake is calculated for each of these planes. In the image below it is visible how the snake

determines the boundaries of the vesicles. Each curve is a collection of point which is used in next step.

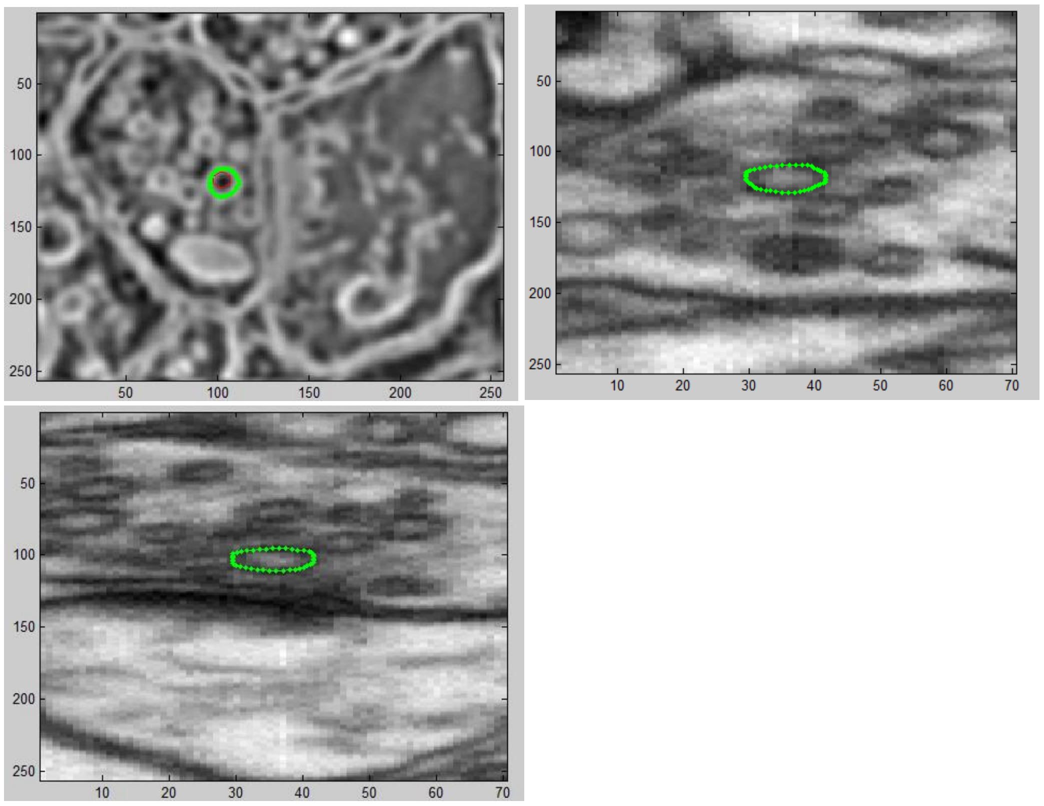
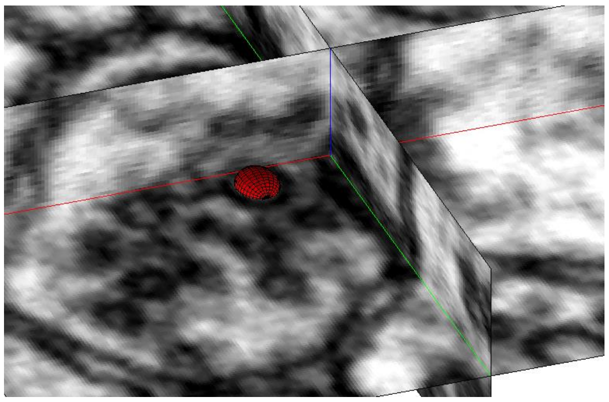
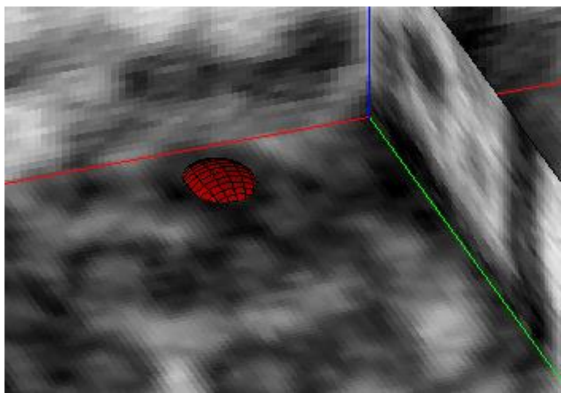


Fig.7 Snake algorithm applied over all the 3 orthogonal directions. Snake is drawn with a green curve

- After the curve for each plane is determined, for the points along each curve it is calculated the ellipsoid as described at chapter about fitting ellipsoid to an array of 3d points. In the image below it is show that the ellipsoid fits inside the vesicle on which was clicked. The images show from below and above that the ellipsoid ends almost on the slice on which the end of vesicle is.

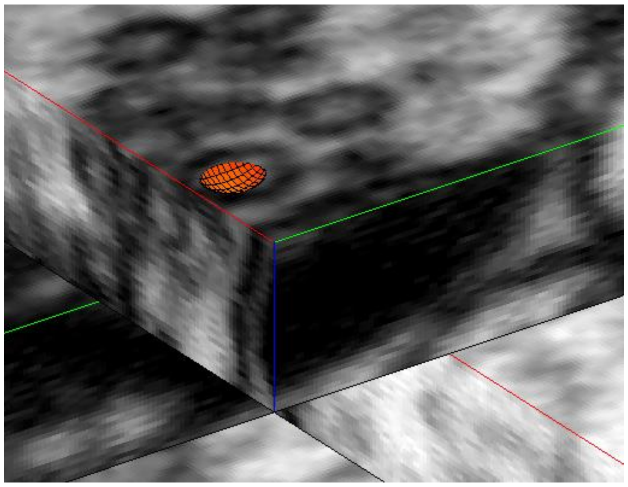


Fig.8 Ellipsoid displayed over the orthogonal slicer from different view angles with slices moved to see that the ellipsoid follows the boundaries of vesicle.

- In the last step the results are displayed on the orthogonal slicer that enables to visualize how the ellipsoid fits inside the vesicle. The results obtained are also processed further at the experiments stage to find how accurate is the ellipsoids are calculated

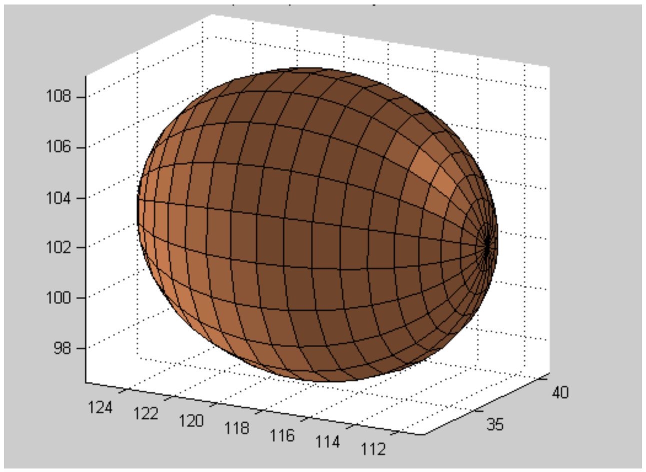


Fig.9 Ellipsoid displayed on a 3d viewer separately from the block with dataset

In the images above it can be notices that the ellipsoid does not fit perfectly and is actually smaller than the vesicle. This may happen because either the standard deviation for ellipsoid calculation was not chosen properly or other parameters are not at their good value. By using the results obtained and reading image energy values along the coordinates of ellipsoid it experiments will be done to find the optimal parameters for obtaining the best results.

5. Algorithm 2: Snake applied on several horizontal slices

For the second algorithm it has been used the remarks that the resolution of image along the vertical planes is not very good and because of that the snake algorithm may not work very well at detecting boundaries on vertical slices. For this it was considered to apply the snake algorithm over several consecutive horizontal slices until the end of the vesicle is found. For detecting the end of vesicle, the ellipsoid for the curve of each 2d plane is calculated and if there is too big change in its dimensions, the algorithm stops.

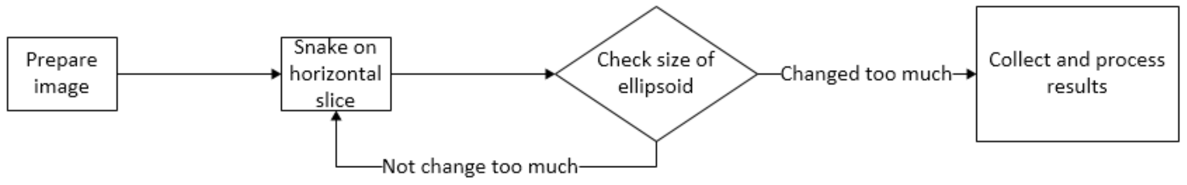


Fig.9 Diagram showing how the algorithm for processing several horizontal slices works

This algorithm is processing the image to detect the boundaries of vesicle by going through the following steps:

- The image is prepared as for the previous algorithm: Files are read, Image energy is calculated and the vesicle center is located by click from the user.
- The snake algorithm is applied for the current slice to find the boundaries of the vesicle
- The size of ellipsoid is checked and compared with dimensions obtained for the previous slice and if the sizes changed too much it means that the vesicle ended
- The current snakes takes as starting curve the curve from the previous image because this makes the algorithm run faster and more accurate (the shape is vesicle in 2d is not changing very much from one slice to another)
- After the boundaries of vesicles from the upper images are found the algorithm is also going down through slices starting from the one that was clicked.
- The curves from all the slices are collected and their points are concatenated to a single array for which the ellipsoid is calculated and the results are displayed and processed in the same way as for the other algorithm

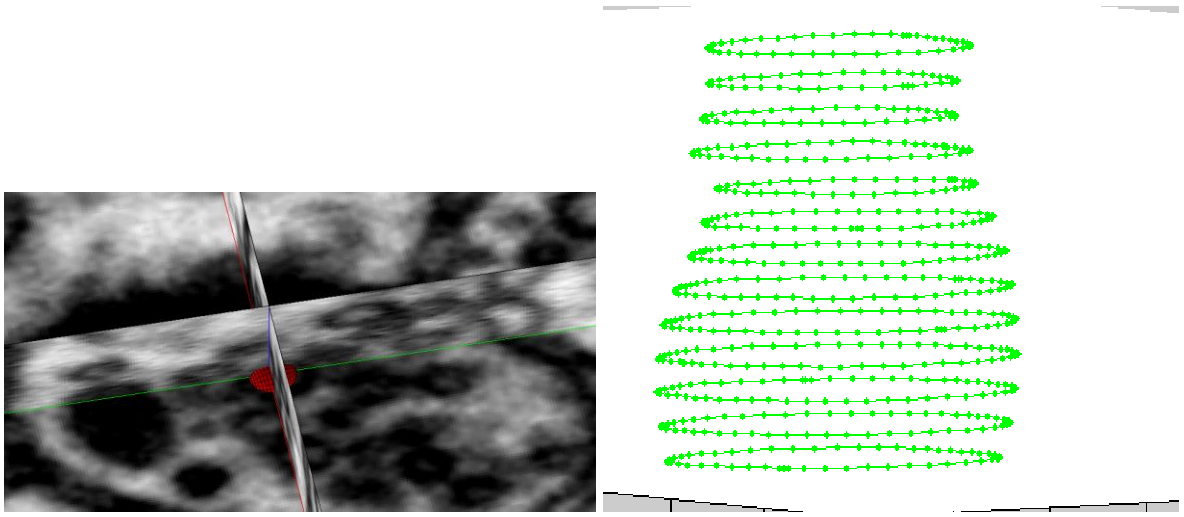


Fig.9 Display of ellipsoid that is obtained using the algorithm described previously over the orthogonal slider. And in the right image the curves obtained for each slice displayed on a 3d viewer.

As it can be seen in the image above it is visible that in the upper part the last 3 snakes seem to be obtained in images where the vesicle is no longer present. And it in the down part of the image it seems that the calculation of curves stopped before the slice on which the vesicle ends. This may prove that the algorithm is not working that well.

At first sight it seem this algorithm is less effective than the previous one because it fails quite often in detecting the end of vesicle properly. But more tests will be done with varying the number of runs for the snake algorithm in each slice and the value of deviation which should decide that dimensions of 2d ellipsoid for each snake has changed too much. Perhaps, if setting the parameters properly then the algorithm may work a lot better.

6. Experiments

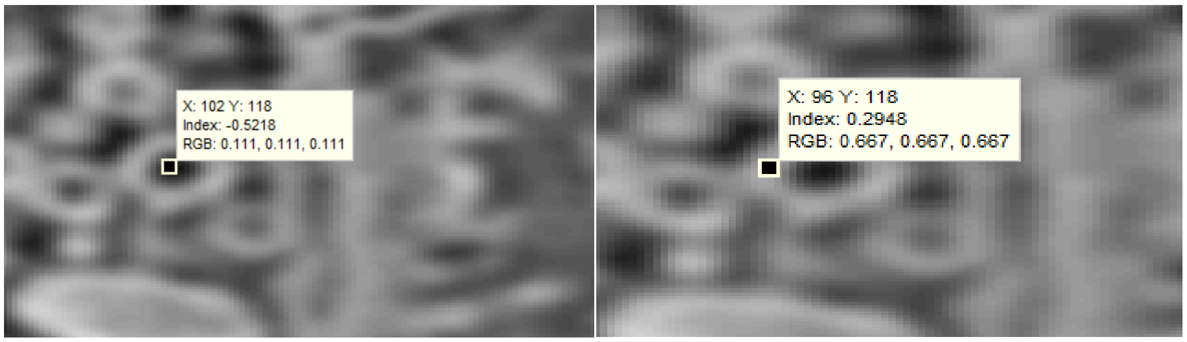
For the experiments it was created a program which wraps the two algorithms and allowed to click the middle of all vesicles in the dataset by navigating through all horizontal slices. The collection of points is used to run the algorithms described previously and change different parameters at each run.

In order to determine how accurate the calculation of ellipsoid that fits over the vesicle is, it has been decided to calculate the energy of all points that are obtained from applying the snake algorithms using the two methods.

At the beginning, on each algorithm the energy is calculated along the curves obtained from the snakes applied on the 3 orthogonal to find the optimal values for α, β that are the terms which control the way that snakes calculates the internal spine energy; the parameter sigma which is the deviation of the Gaussian that used for smoothing the image before the snake is applied and the number of steps in which snake reads the value of external forces to change the shape to fit along the edges of vesicle. After this it will be calculated energy along the edges of ellipsoid that is obtained to see how close it is to the edges with high energy of the vesicle. After these are checked then the algorithm that applies snake on horizontal slices will be tested to find at which deviation of 2d curve sizes the algorithm should stop.

In the image below it is displayed the value of image energy in two points: one point on the dark area inside the vesicle which is negative because it has low energy value and one point on the boundaries of vesicle which has positive value. This confirms that if the energy on snake curve is high then it is more accurately placed over the boundaries of vesicle.

For calculation on energy to all snakes for each vesicle in the image the energy obtained for the points on each vesicle are summed and then averaged to be compared with energy obtained for other parameters. The results will be displayed graphically to prove the choice of parameters.



X: 102 Y: 118
Index: -0.5218
RGB: 0.111, 0.111, 0.111

X: 96 Y: 118
Index: 0.2948
RGB: 0.667, 0.667, 0.667

Fig.10 Reading value of energy (Index) at two points: one inside the vesicle and one on the edge of vesicle

6.1. Number of updates for the snake position

First tests were done to find the optimal number of runs of last step from the snake algorithm: the updates of the control points according to the gradient descent equation. This is done to ensure that the number of updates that are done is enough to make accurate detection for the boundaries of vesicle.

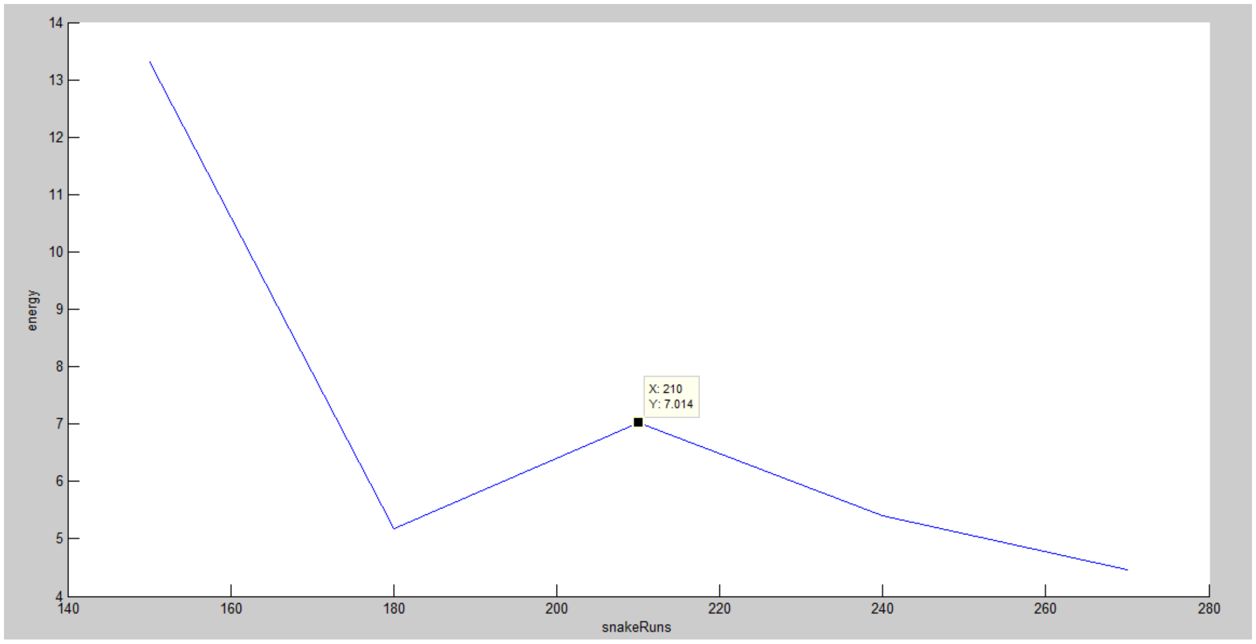


Fig.11 Average energy for points on snake curves for different number of runs for the snake position is updated as part of its own algorithm.

In the graphic above it is shown that it is the best to make 210 updates of the snake control position. After this value the energy average is lower because the snake tends to grow over the boundaries of vesicle and ends up to find other appropriate edges in the image. Perhaps the value is higher for very low number of updates because of noisiness but even visually on the orthogonal slices it was visible that value 210 is the best choice.

6.2. Size of terms alpha and beta for calculating the snake internal spine energy

The next experiments were done to find the optimal values of parameters α, β to see which are the best values to make the snake detect the edges of vesicle correctly.

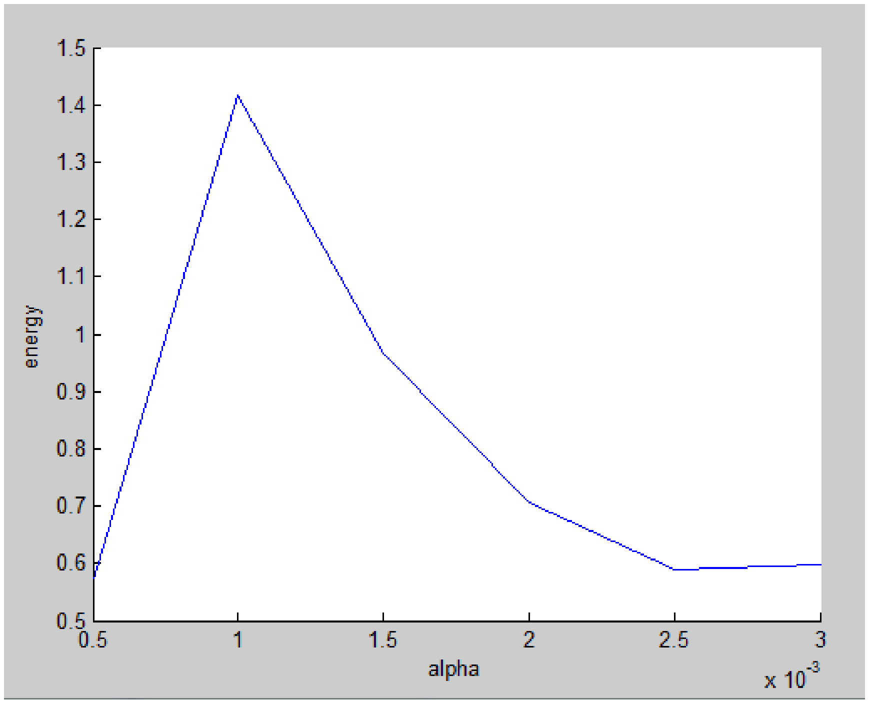


Fig.12 Average energy obtained by varying the parameter alpha when updating the vesicles position

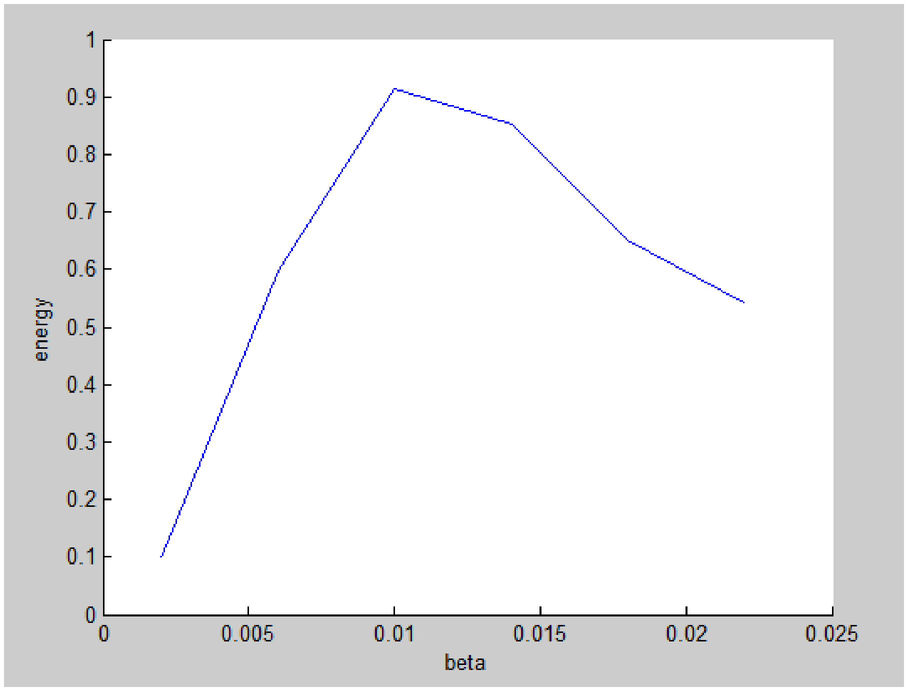


Fig.13 Average energy obtained by varying the parameter beta when updating the vesicles position

The experiments and graphs from above show that by choosing value 0.01 for parameter *beta* and value 0.001 for parameter *alpha* the vesicle edge is obtained the best detection for edge of vesicle on the 3 orthogonal planes.

6.3. Standard deviation for Gaussian smoothing

Further tests were done to find the parameter *sigma* which is the deviation of Gaussian filter that is used for smoothing the image before calculating the gradient and apply the snake algorithm. The choice of parameter sigma is important because the smoothing determines how higher the energy of edges in image is than the surrounding areas and if the energy of vesicle boundaries is higher then the snake algorithm is more effective in finding the shape of vesicle.

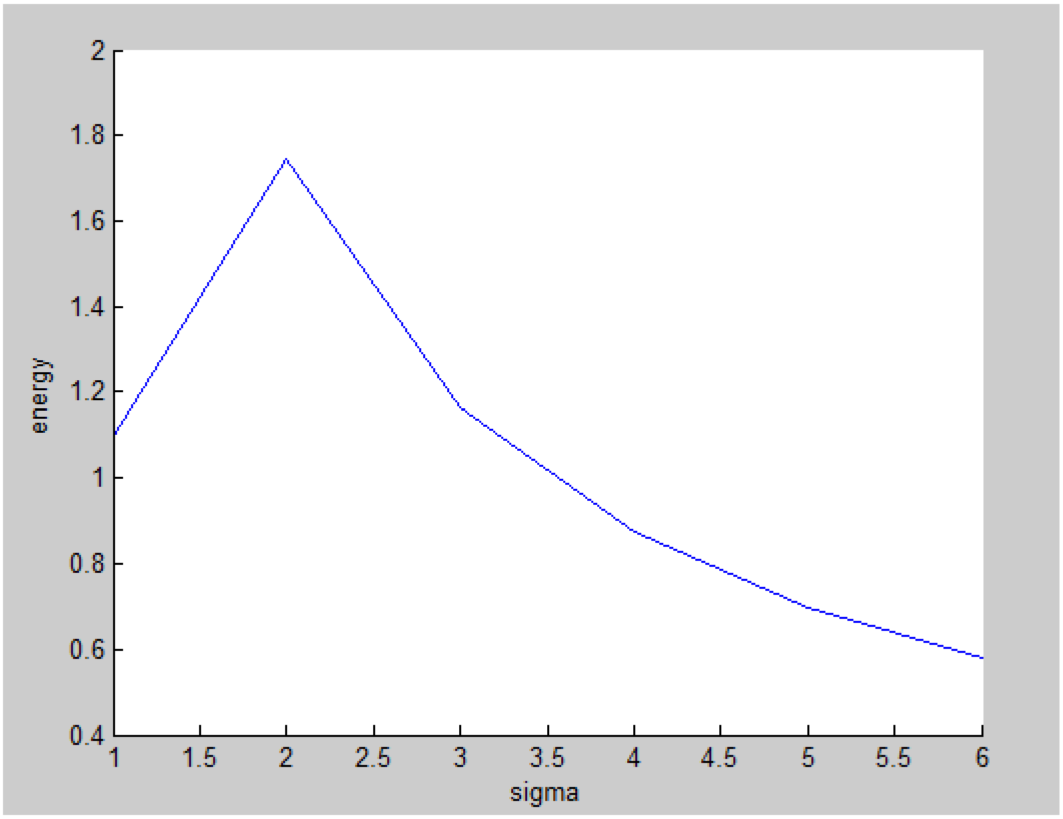


Fig.14 Average energy of detected vesicle boundaries for varying the standard deviation of Gaussian filter that is used for smoothing the images

The experiment has shown that the value 2 for sigma is the best choice for Gaussian standard deviation that is used for preparing the image to apply the snake. For higher value the image becomes too blurred and the boundaries of vesicles are harder to distinguish and it seems it does not work at all for lower values.

Further tests are done by reading image energy in points calculated for the ellipsoid that is fit to points obtained by the 3 snakes. The purpose of this experiment is to find the standard deviation of algorithm that calculates the size of ellipsoid so that it fits better on the edges of vesicle.

It has been found that the value 2 for the deviation is the best for calculating the dimensions of ellipsoid that to fit inside the vesicle.

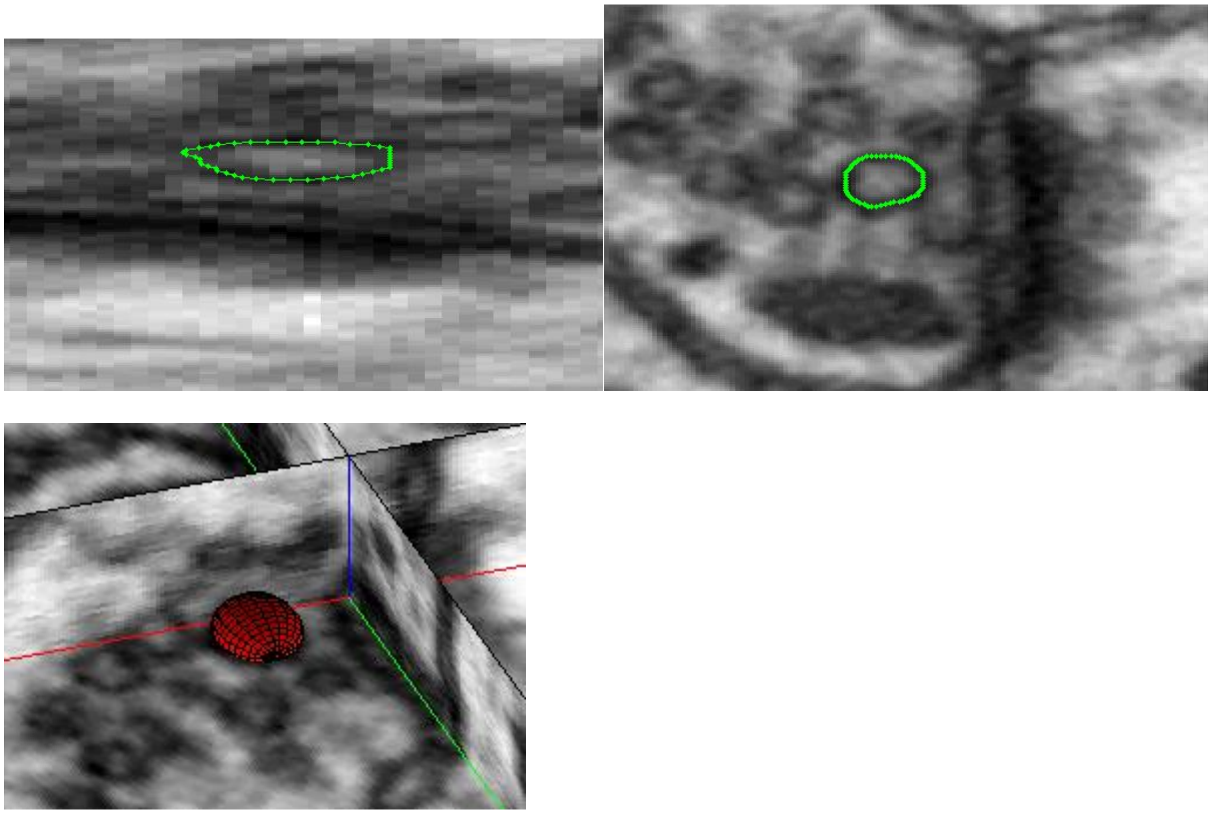


Fig.15 Result for applying the snake algorithm and fitting ellipsoid by using the parameters that were found above to work the best

6.4. Parameters for Algorithm 2

Now the optimal parameters have been found for running the first algorithm inside the final application. Detection of vesicle was tried again with these parameters. This time it seems the boundaries of vesicle along the perpendicular orthogonal planes are looking almost the same but the ellipsoid is fit a lot better inside the vesicle and it is no longer a lot smaller than the vesicle.

After this stage of experiments, the first algorithm is working properly but further tests have to be done to check if the algorithm that detects vesicle shape by running snake on several horizontal slices can be improved by changing the parameters that were used and make it work better and suitable for comparison between these two algorithms.

Since for the parameters that were previously tested, the optimal values were found, now it is only needed to find which is the deviation on 2d ellipsoid size which should make the algorithm to stop if the vesicle is no longer present in the current slice.

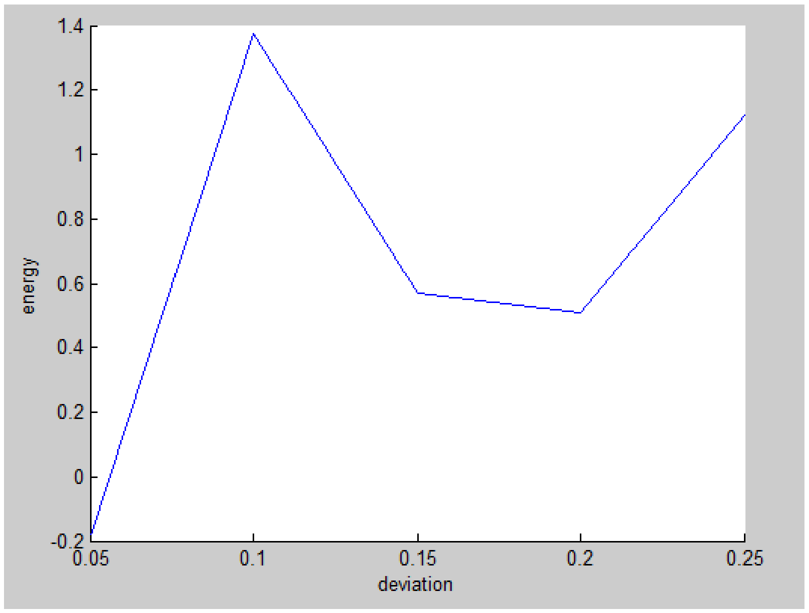


Fig.16 Average energy for the curves points that were detected by running the second algorithm calculated using different values deviation of ellipsoid along 2 consecutive slides

In the graph that was obtained it is shown that value 0.1 is optimal for the deviation at which the snake algorithm should stop because the vesicle is no longer present in the slice.

By running again the algorithm with the value that was found it appears that some improvements happened.

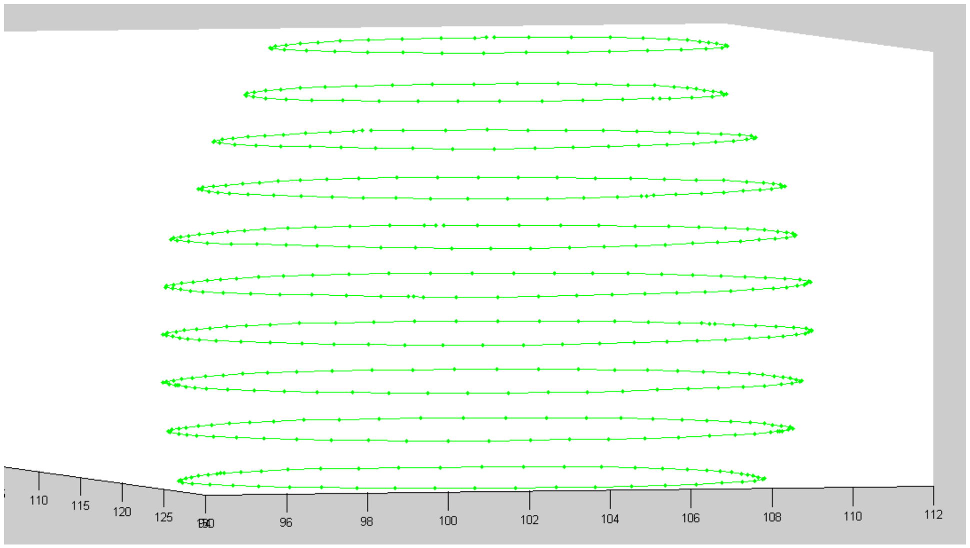


Fig.17 Curves obtained for applying snake algorithm on several horizontal images by using the second algorithm with deviation 1

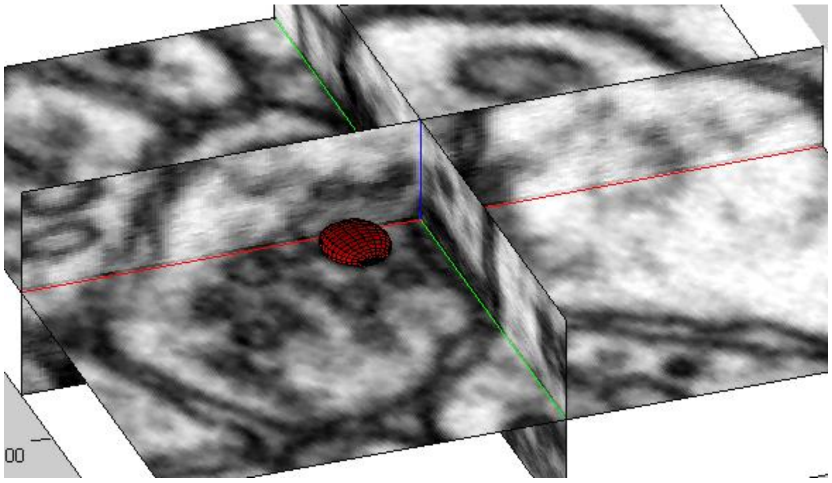


Fig.18 Ellipsoid resulting from the second algorithm displayed over the orthogonal slicer

In the images above, especially the one displaying the curves for each horizontal image it is visible that on the upper part the results no longer include wrong additional curves which were sometimes wrongly calculated for different value of deviation chosen for the 2d ellipsoid. It also appears that the curves calculated below the slice on which was clicked are also calculated better than at the chapter describing the algorithm, where the algorithm was stopping too early.

6.5. Comparison between the two algorithms

For making a comparison between the two algorithms the total energy of points on ellipsoid from the image energy are calculated for the same set of points and plotted on a same graphic.

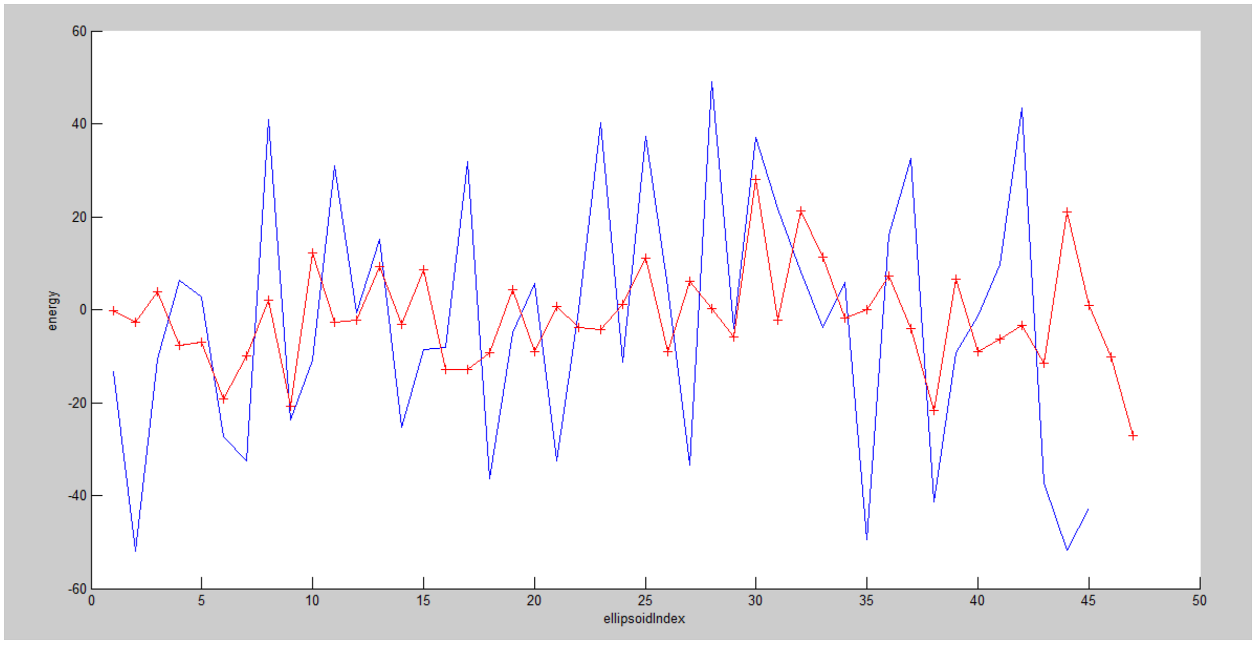


Fig.19 Energies of ellipsoids obtained from the first algorithm displayed with blue color and for the second algorithm it is displayed with red color

In the graphic it seems that for the algorithm that is applied on the 3 orthogonal planes there it is a lot larger difference at total energy on the surface of ellipsoids that are calculated. It also happens that some points obtained with the first algorithm are missing because it happen errors especially because the snake algorithm does not work properly on the vertical slices.

7. Conclusion

This project offered two different algorithms for detecting the 3d shape of the vesicles in dataset of images that form a 3d block. Both algorithms are using Active Contour Models(snakes) as 2d algorithm for detecting the shape of vesicles. Snake algorithm has the advantage of having good results in this project where the user knows that the shape of snakes is approximately spherical and also offers the algorithm the coordinates of a point in the middle of vesicle.

The first algorithm is calculating the vesicle shape by running the snakes on the 3 orthogonal planes converging to the point that is clicked. This algorithm has proven to offer quite accurate detection for the shape of vesicle by only running the snake 3 times. The main drawback is that the resolution of 3d block along the vertical planes is not that good and sometimes the snake algorithm fails on vertical plane because it cannot find the energy of the boundaries.

The second algorithm is calculating the vesicle shape by running the snakes on several horizontal slices adjacent to the plane on which the click was done. The main advantage of this algorithm is that it has not given any error at running the snake algorithm because the resolution of horizontal planes is better and edges of vesicles are easier to detect. A drawback for this algorithm is that it sometimes noisy curves that are present on the next slice where snake is calculated are making the algorithm not to stop properly. This mostly happen because the shape of vesicles on top and bottom is usually different. Sometimes upper boundary is detected too late and when going down through slices a thinner part of vesicle makes that the deviation of snake curve to be too large and algorithm stops too early.

In terms of speed the first algorithm is a lot faster because it only runs the snake three times while the second algorithm makes 7 or more runs of the snake and this can make the application too slow. But in case of efficiency both algorithm have drawbacks from some points of view.

Perhaps different criteria for the second algorithm to decide on which slice to stop with snake calculation could make it very efficient. And for the first algorithm perhaps calculating the snake on even more planes which to converge to the point on which the click is done at different angles could make the algorithm more efficient and snake edges detection a lot better.

Another improvement that could be done to this project could be in the stage of experiments where perhaps by only calculating the total energy of the boundaries that were calculated is not enough to decide which parameters are the best for the calculations. One strategy could be to take a smaller set of vesicles and somehow to manually draw their 3D shape and then run the algorithms with different sets of parameters and find how big are the differences.

Finally, the project has offered two different solutions for solving the problem that was given and each of them seems to work better for detecting the shape of other vesicles. With some improvements of the algorithms and the way that experiments are done, the results can be significantly improved and the algorithms may be used for making very accurate applications.

8. Bibliography

[1]Orthogonalslicer by Laszlo Balkay : http://www.mathworks.com/matlabcentral/fileexchange/5934-orthogonalslicer

[2]Images dataset from CVLAB (http://cvlab.epfl.ch/data/em)

[3]Obtain 3d ellipsoid using covariance : using the original demo by Rajiv Singh (http://www.mathworks.com/matlabcentral/newsreader/view_thread/42966)

[4]Information about covariance from Wikipedia (http://en.wikipedia.org/wiki/Covariance)

[5]eigenvalue and eigenvector from Wikipedia (http://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors)

[6]Information about vesicles from Wikipedia (http://en.wikipedia.org/wiki/Vesicle_%28biology_and_chemistry%29)

[7]Information about snakes algorithm from Wikipedia (http://en.wikipedia.org/wiki/Active_contour_model) and

[8]Michael Kass et al., "Snakes: Active Contour Models", International Journal of Computer Vision, p. 321-331, 1988

[9]Functions *snake.m* and *scalen.m* from Jon Sporring