Deformable models

Marie Diez, Lucrezia Tosato, Lucie Jandet

12 novembre 2021

1 Detailed explanations on each method

This first section aims to give explanations of the working principle of two deformable model methods, through the manipulation of their implementation in python. The influence of each parameter and its influence on the result will be explained.

There are different ways of representing deformable models :

— Parametric curves

— Level Set

These models aim to minimize an energy based on 2 possible criteria :

— Attraction of the gradient by the contours

— Homogeneity of regions

— Other types of additional constraints : regularity, balloon force...

In this pratical work, 2 approaches are used (several other compositions of representation and criteria are possible)

1.1 First method

The first method is a parametric representation with the criteria of the attraction of the gradient by the contours : the "parametric active contour" method. The function from skimage for this method is the following :

```
snake = active_contour(gaussian(im, 0.1), init, alpha=0.5, beta=5, w_edge=20,
    gamma=0.001)
```

- im : image we are working on
- gaussian(im,0.1) : gaussian filter applied with a defined standard deviation, in this example 0.1
- init : initial contour (or snake)
- alpha : weight given to the control of the tension (or the length) of the curve
- beta : weight given to the control of the curvature of the curve
- w_edge : controls attraction to edges
- gamma : time stepping parameter

The goal is to minimise the following energy formula :

$$E_{\text{total}} = \int_0^1 \left(E_{\text{int}}\left(v(s)\right) + E_{\text{image}}\left(v(s)\right) + E_{\text{ext}}\left(v(s)\right)\right) ds$$

With :

$$E_{\text{image}} = g(|\nabla f|)$$

α **and** β A parallel with the course helps to understand more the method. The principle is to define a curve $v(s) = [x(s), y(s)]^t$ with s $\in [0; 1]$. Then this curve evolves under internal and external forces represented by the minimization of an energy which is the sum for the internal energy of the curve, the energy of the image (attraction by high gradients) and the external energy which depends on the problem. Alpha and bêta are parts of the internal energy :

$$E_{int} = \alpha(s)(\frac{dv}{ds})^2 + \beta(s)(\frac{d^2v}{ds^2})^2 \tag{1}$$

where the first derivative of the contour represents the tension of the curvature and the second derivative the control of the curvature. Practical examples are shown below to understand more the influence of α and β. The Figures are plots of the heart, with in red the initial contour and in blue the final contour, aiming to fit the left ventricle of the heart. The final contour is respectively calculated for α values of 2, 5 and 10.

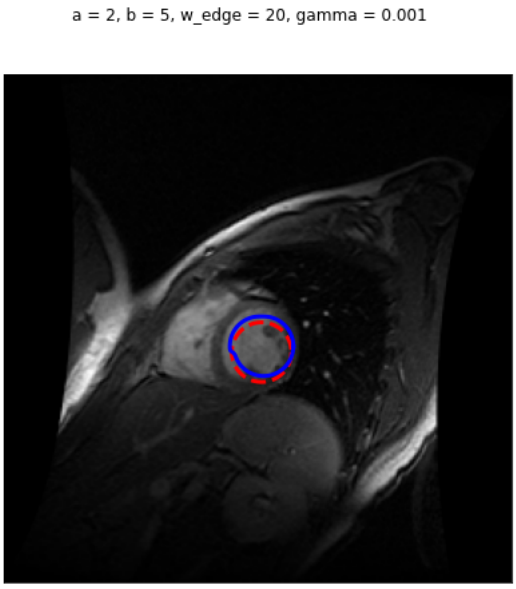

FIGURE 1 – $\alpha = 2$

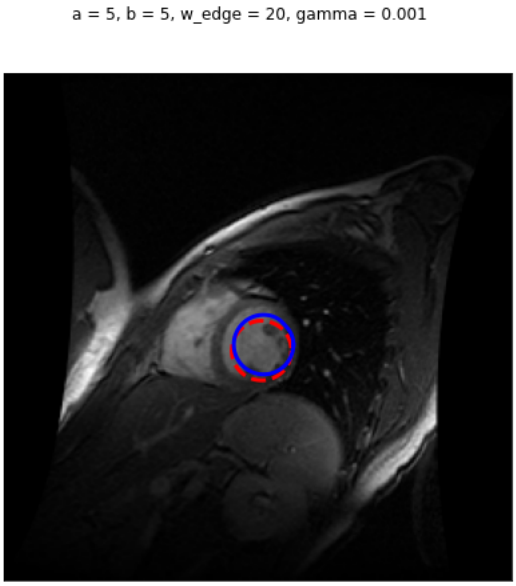

FIGURE 2 – $\alpha = 5$

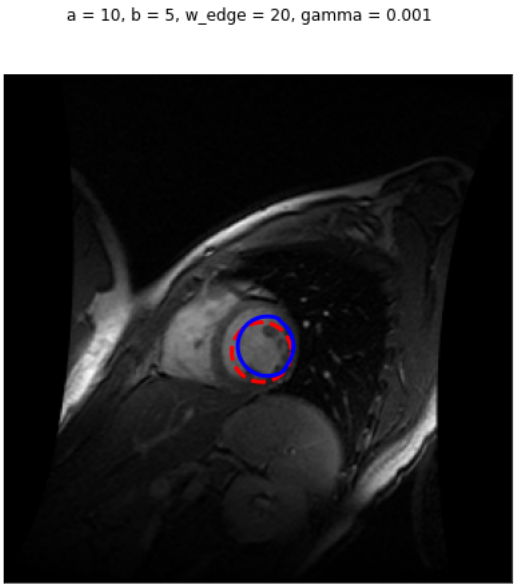

FIGURE 3 – $\alpha = 10$

α is the regularization of the shape, it control the length of the contour, so the more alpha is big in front of beta, the lower the expansion of the contour is. High values of alpha make the snake contract faster for regular object.
The Figures below are plots with a variation of beta, respectively with β values of 2 and 10.

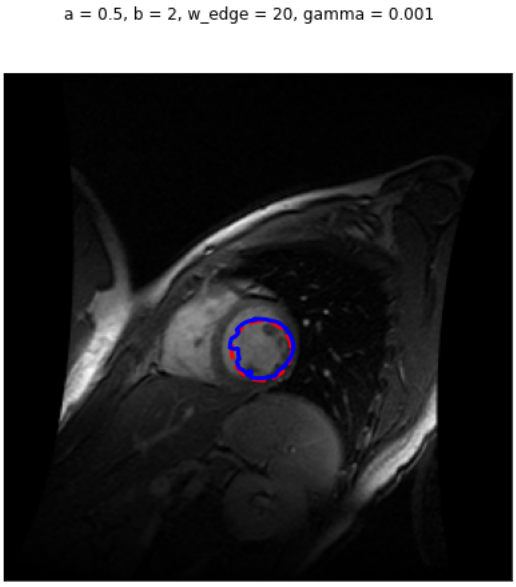

FIGURE 4 – $\beta = 2$

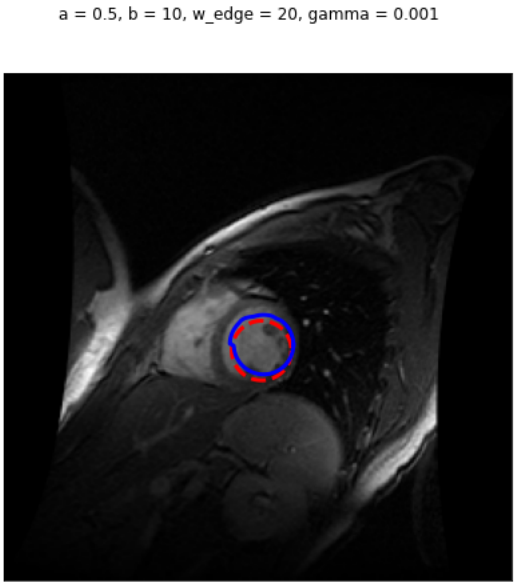

FIGURE 5 – $\beta = 10$

Beta control the curvature of the contour, increasing the value of beta produce contours which are more regularized, it makes the snake smoother.
Alpha and Beta are linked to each other, in fact if we reduce the curvature we will reduce the length of the contour and vice versa, it is therefore relatively difficult to adjust these two parameters.

gamma The course explains that the contour at the iteration t+1 is derived from the contour at iteration t trough the equation :

$$v(t+1) = (A + \gamma I)^{-1}(F(v(t)) + \gamma v(t)) \tag{2}$$

Where A is the matrix solving AV = F, with V the vector of the contour at different iterations and F the negative value of the gradient of P. P is the sum of the image energy and external energy. γ is the inernia of the computation. The higher γ, the faster the method will converge, but if the calculation step is too big, the result will be really inaccurate, as shown on the Figures below, with γ respectively of 0.01 and 0.001.

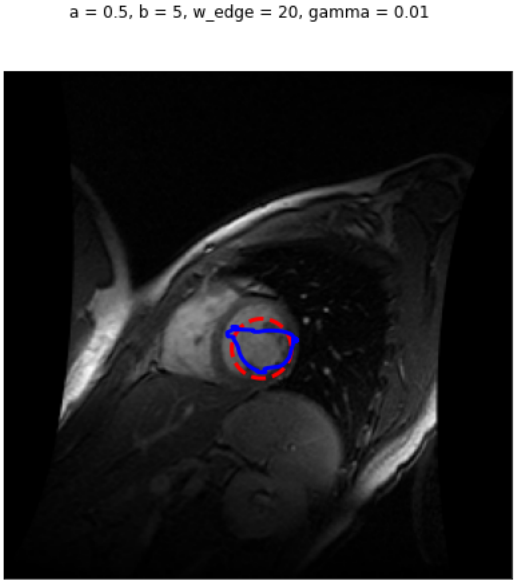

FIGURE 6 – $\gamma = 0.01$

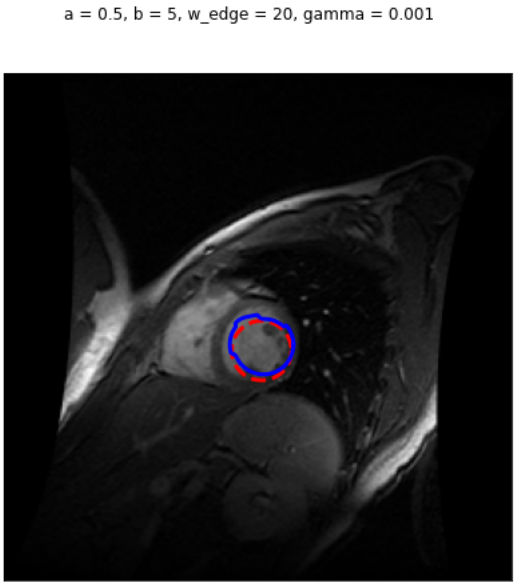

FIGURE 7 – $\gamma = 0.001$

wedge The wedge value indicates how much the contour will be attracted to edges. For very low value, the contour doesn't evolve in the right direction because it is not attracted enough by the contours, as shown on the Figures below with values of 1, 5 and 20. Conversely, negative values will push the model away from the edges.

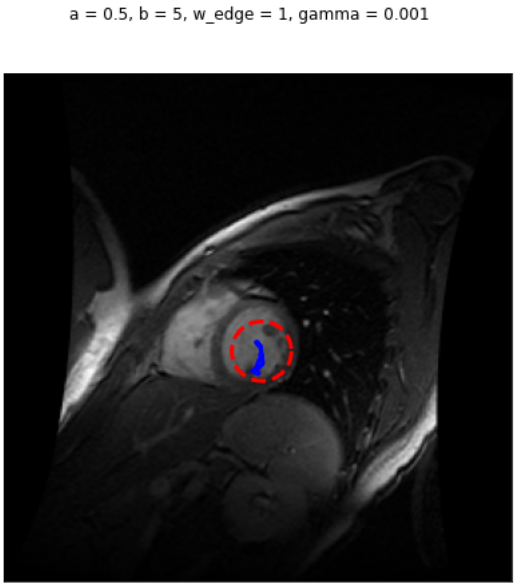

FIGURE 8 – wedge = 1

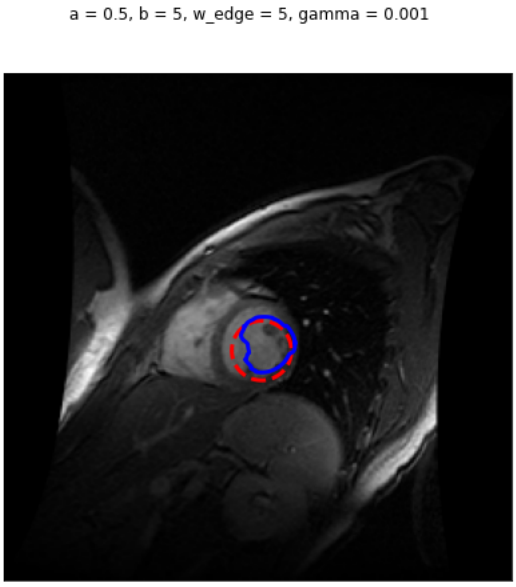

FIGURE 9 – wedge = 5

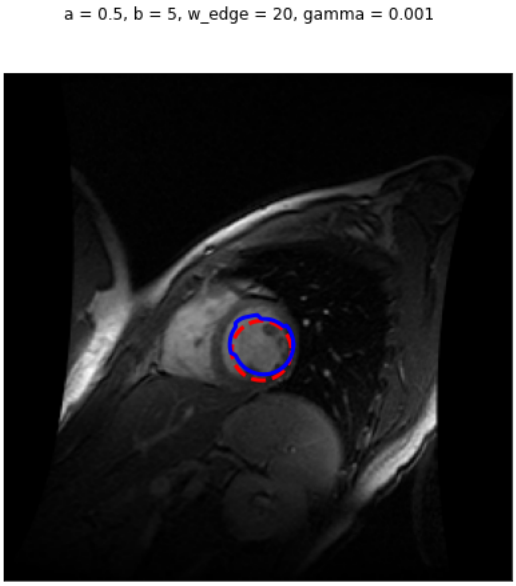

FIGURE 10 – wedge = 20

wline This parameter controls the attraction of edges to light (and negatively dark) regions. Deformable model in blue as a function of w_line with $w\_edge = 20$, $\alpha = 1$ et $\beta = 1$:

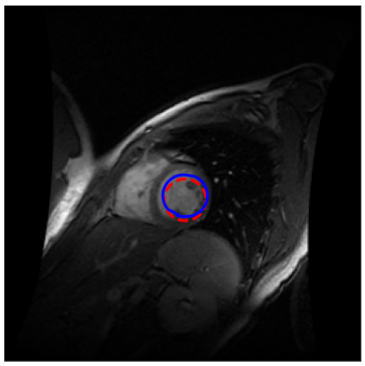

FIGURE 11 – $w\_line = 1$

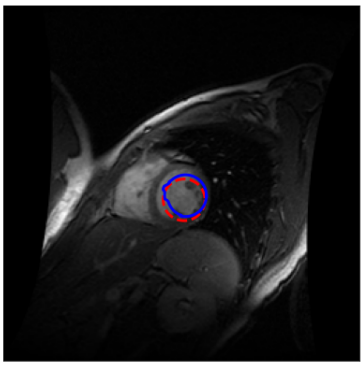

FIGURE 12 – $w\_line = -1$

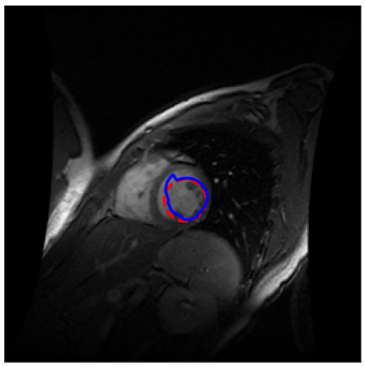

FIGURE 13 – $w\_line = 10$

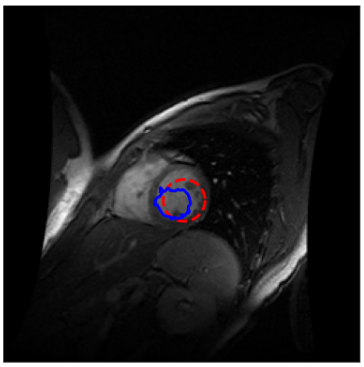

FIGURE 14 – $w\_line = -10$

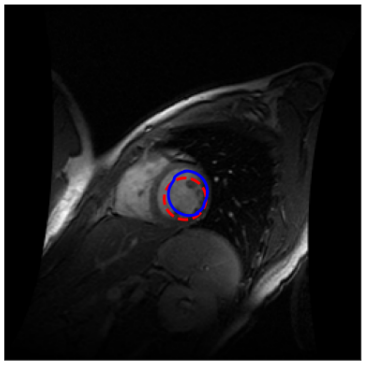

FIGURE 15 – $w\_line = 20$

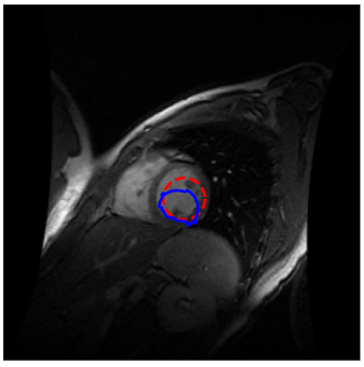

FIGURE 16 – $w\_line = -20$

max_px_move This parameter corresponds to the maximum number of pixels that can be moved in each iteration.
Blue deformable model as a function of max_px_move with $w\_edge = 20$, $\alpha = 1$ and $\beta = 1$:

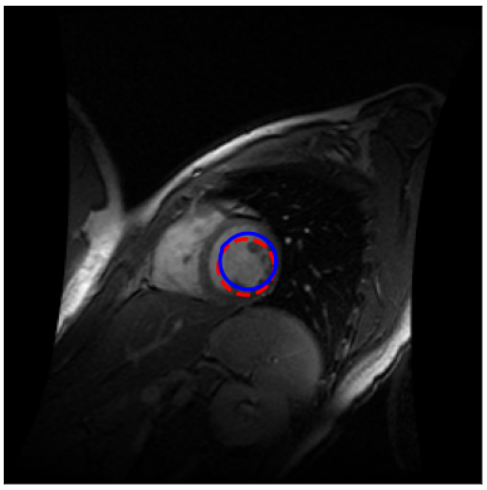

FIGURE 17 – $max\_px\_move = 0$

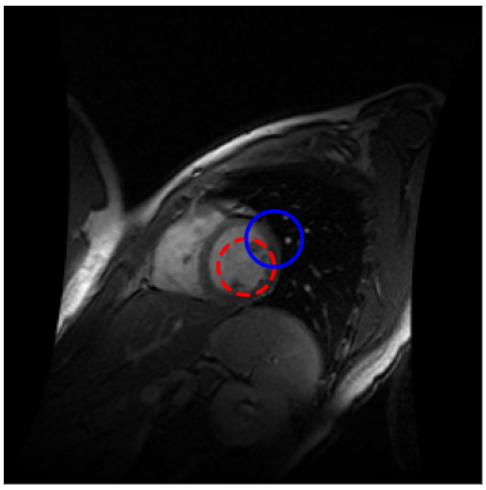

FIGURE 18 – $max\_px\_move = 5$

Active contour method can also be use as an open contour :

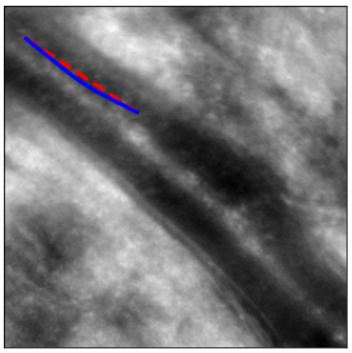

FIGURE 19 – Deformable model open in blue with $w\_line = -10$, $w\_edge = 20$, $\alpha = 20$, $\beta = 1$ and $\gamma = 0.001$

Conclusion on the first method We have seen how different parameters of the parametric active contour method may influence the direction of evolution of the contour. However, the main drawback of this method is that it will never give good results if the contour initialization is too far from the targeted shape.

External forces To solve this initialisation problem we can use some external forces :

— Gradient vector flow (GVF)
Starting from a simple initialization, the GVF approach will allow to propagate the gradients, which will allow to minimize the energy even starting from an initialization relatively far from the contours to be segmented.

— Balloon force
This external force will enable to push the initialization towards the contours.

Here are the results with an application of the balloon force, for that we had to use the function : morphological_geodesic_active_contour of scikit-image, this function uses the level set

representation and optimizes the gradient criterion :

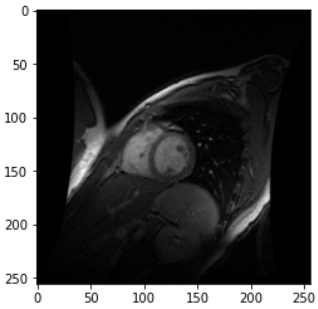

FIGURE 20 – Original image

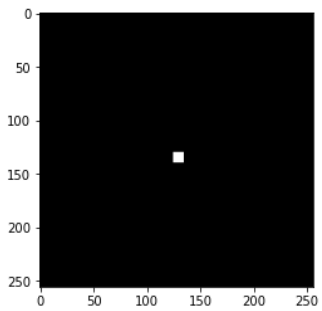

FIGURE 21 – Initial level set

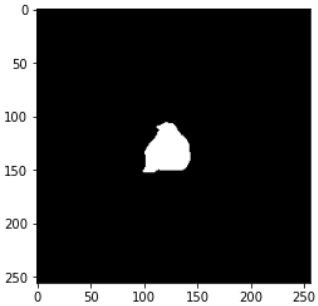

FIGURE 22 – Segmentation mask

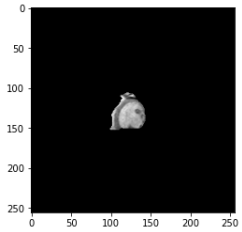

FIGURE 23 – Segmentation visualization

We can see that the balloon force has a good effect, starting from a small point in the object, the active contours get bigger until the final segmentation. The result is not very interesting, the object is well defined but the segmentation is not precise enough to be completely accurate. Pre-processing and improvement of the model parameters should be done.

1.2 Second method

The second method is an implicit representation using level sets, with a criteria on region homogeneity. It is called on python in the following way :

```
cv = chan_vese(image, mu=0.25, lambda1=5, lambda2=1, tol=1e-3, max_iter=200, dt
    =0.5, init_level_set=init_ls, extended_output=True)
```

- image : image we are working on
- mu : the smaller the value, the more details will be conserved for the global segmentation
- lambda 1 and lambda 2 : they give the relative range of values for each region of the segmentation
- tol : when the level set variation between two iterations is smaller than this value, it is considered that the final segmentation solution is reached
- maxiter : gives the maximum number of iterations allowed before the interruption of the algorithm
- dt : multiplication factor used to accelerate the algorithm
- initlevelset : initialization of the level set

The implicit representation using level sets to segment an image starts by defining a function Φ such as :

$$\begin{array}{rccc} \Phi & : & \mathbb{R}^2 \times \mathbb{R} & \rightarrow & \mathbb{R} \\ & & (x, y, t) & \mapsto & \Phi(x, y, t) \end{array}$$

Where x and y are 2D coordinates and t the time in the minimization process. A level set is define by the values of Φ at iteration t. It can be represented by an image with these values corresponding to pixels positions (x,y). The contour of the segmentation is defined as the points of the level set such as Φ is equal to zero, at a fixed time t. The level set, and therefore the contour, evolve with the time t. The evolution of Φ at each time is given by the solution to the equation :

$$\frac{\partial \Phi}{\partial t} = -F||grad\Phi||$$

with F the propagation speed.
To resolve this equation, the result of the minimization of the function (3) is needed. The function (3) to minimize is written in the piecewise constant case of Mumford and Shah approach. In this case, the image of the segmentation contains only two regions with constant values g_1 and g_2. The first two terms tend to minimize the distance between the pixels values of the image f and g. The last term tends to minimize the length of the segmentation contour.

$$U(\Gamma, g, f) = \lambda_1 \iint_{R_1} (f - g_1)^2 dxdy + \lambda_2 \iint_{R_2} (f - g_2)^2 dxdy + \nu \int_{\Gamma} dl \tag{3}$$

lambda1 and lambda2 This allows to understand the influence of the parameters λ_1 and λ_2. What matters is their relative value. If λ_1 is high in front of λ_2, the minimization process will be more accurate in the first region R1 and values "allowed" in R_1 will have to be closer to g_1 than values allowed in R_2 would have to be closer to g_2. As a result, more values will be admitted in R_2 than in the case where λ_1 and λ_2 would be similar. The higher is λ_1 the less we will accept pixels that are too different in the region R_1 and respectively for λ_2 and R_2.
Below are two segmentation images of the retina. The first one is a plot where λ_1 is equal to 5

and λ_2 is equal to 1 and the second one is a plot where λ_1 is equal to 1 and λ_2 is equal to 5. R_1 is the white region and R_2 is the black region. Furthermore, we can note that changing the relative value between λ_1 and λ_2 changes the size of the regions as indicated previously.

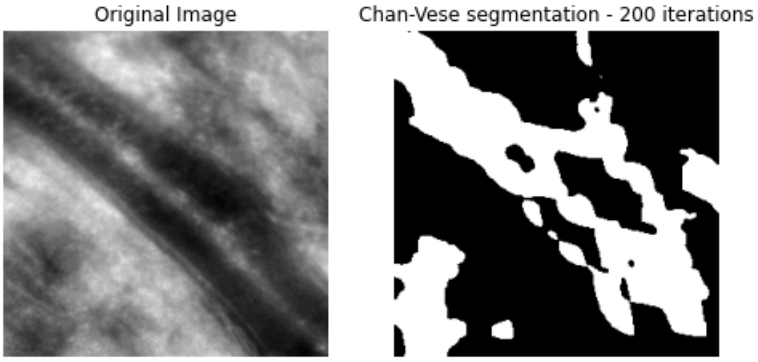

FIGURE 24 – $\lambda_1 = 5$, $\lambda_2 = 1$

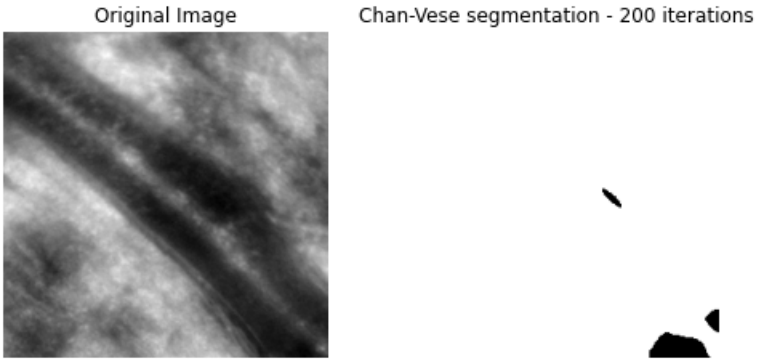

FIGURE 25 – $\lambda_1 = 1$, $\lambda_2 = 5$

mu The parameter μ is a weight that controls the length of the contour. Higher values will produce a more "rounded" level set while lower values will detect smaller, more accurate objects. Result with $\lambda_1 = 7$, $\lambda_2 = 1, tol = 1e-3, dt = 0.5$:

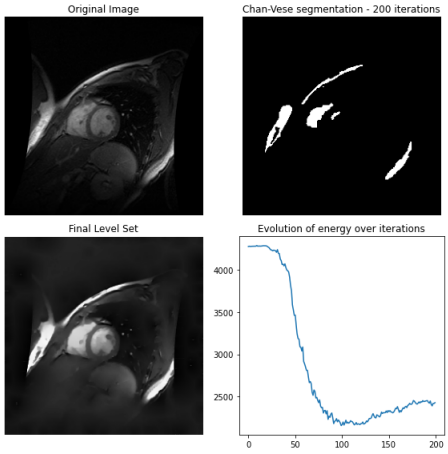

FIGURE 26 – $\mu = 0.1$

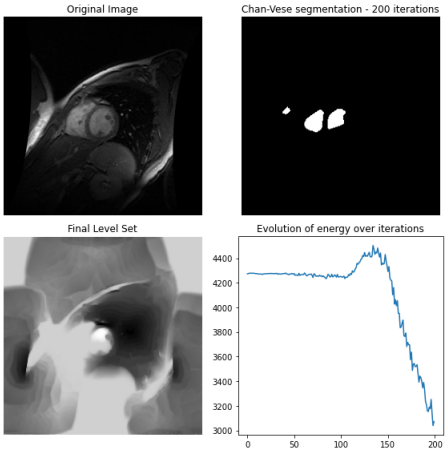

FIGURE 27 – $\mu = 0.3$

We can indeed see on the final level set that the level set corresponding to a lower μ is more precise.

init_level_set The initial level set is a very important parameter which allows to define where we start from. It must indicate the initial location in the object to be segmented, region R_1. As the iterations progress, the algorithm will add new values to the region thanks to the new calculation of the level lines. The segmentation corresponds to level 0.

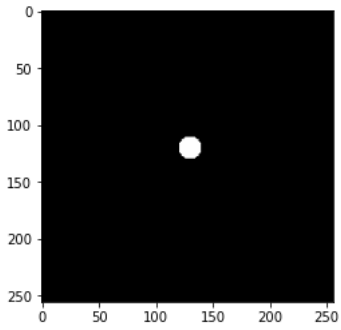

FIGURE 28 – initial level set

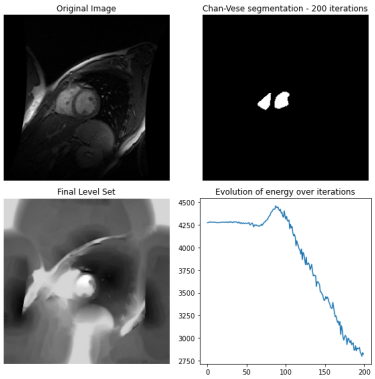

FIGURE 29 – Result

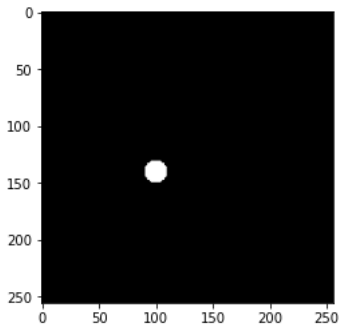

FIGURE 30 – initial level set

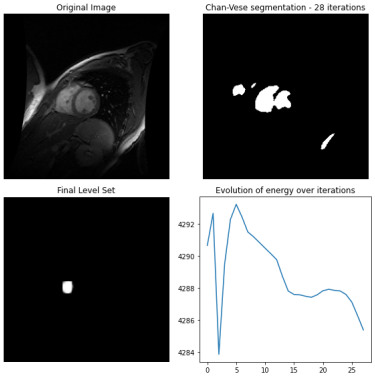

FIGURE 31 – Result

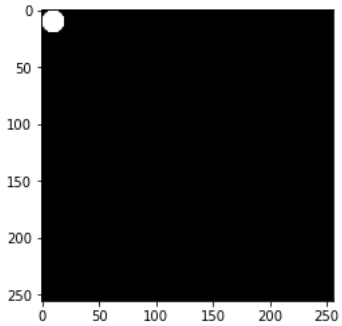

FIGURE 32 – initial level set

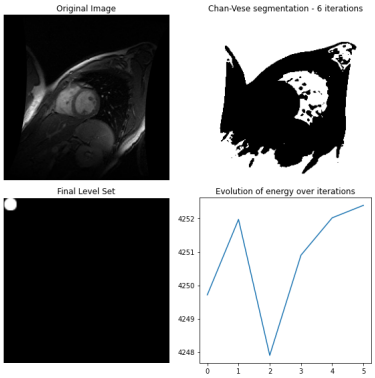

FIGURE 33 – Result

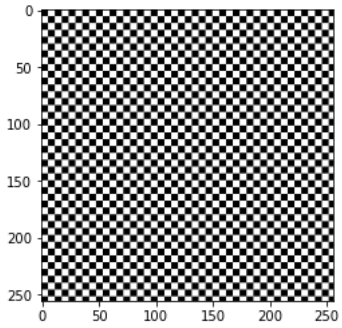

FIGURE 34 – initial level set

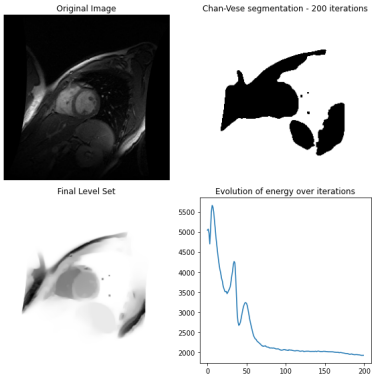

FIGURE 35 – Result

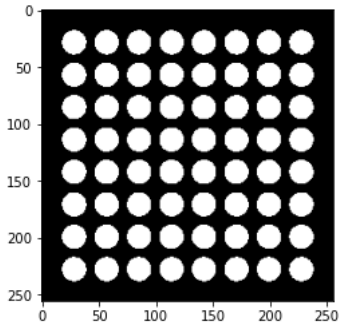

FIGURE 36 – initial level set

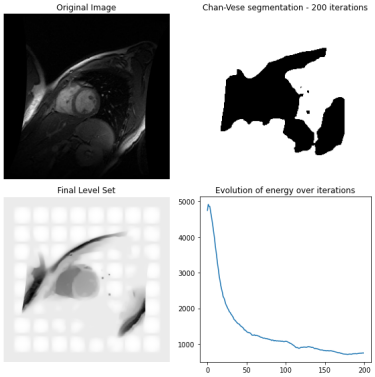

FIGURE 37 – Result

The results are very different depending on the initial level set provided, in fact it is the starting point of the energy minimization, if it is badly placed (relative to what we are trying to segment) we risk not finding the local minimum associated with our object, but falling into another local minimum. For example, if we want to segment the 2 ventricles, the first 3 examples above are quite different in terms of results. The first one gives a good segmentation, the second one would need to slightly revise the parameters or slightly better place the initial point, the results are on the whole good because the starting point is located in the object to be segmented. However, the third result is very bad, because the starting point is not at all placed in the right place for the segmentation. This point would have to be in the object to be segmented for the algorithm to work properly. The last 2 results will not be interesting for the segmentation of the ventricles (with these parameters), however the checkerboards seem to locate the clear regions (in black in the segmentation). Indeed, starting from an uniform initialization, the minimization will add to the region R_1 (in white) all the pixels that are relatively close to each other in order to keep a certain homogeneity (set by λ_1 and λ_2). So in the end we get 2 regions with on the one hand the lightest regions and on the other hand the darkest. The choice of the initial level set depends on what we want to segment.

max_iter This parameter corresponds to the maximum number of iterations before the algorithm is interrupted. We can make the same comments as before in the sense that if the number is too low the algorithm will not have time to converge and if this one is too large, one risks losing time of calculation necessary.

dt This parameter is a multiplicative factor to speed up the algorithm. However, large values may prevent convergence.

Representation of level lines in 3D To better understand the representation of the level lines and the final segmentation obtained we have displayed the final level set in 3D, with $\lambda_1 = 7, \lambda_2 = 1, \mu = 0.25, tol = 1e-3, dt = 0.5$:

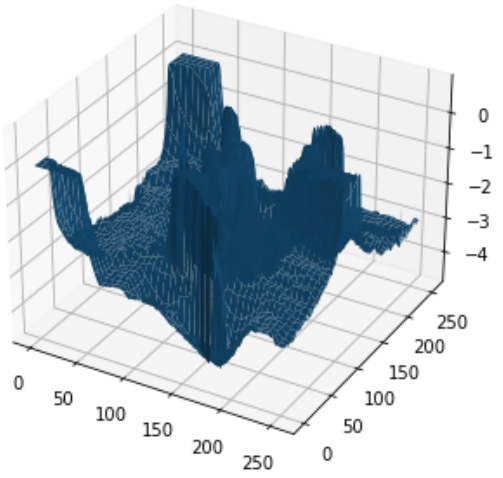

FIGURE 38 – 3D representation of the final level set

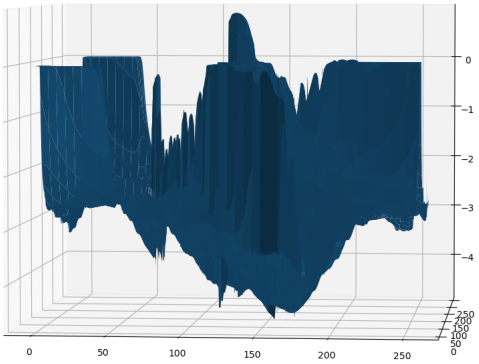

FIGURE 39 – 3D representation of the final level set

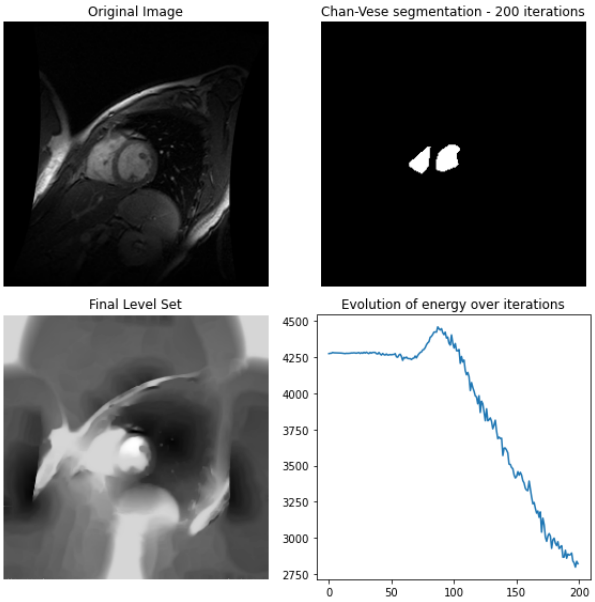

FIGURE 40 – Result

It can then be seen that the final segmentation corresponds to level 0 of the final level set.

2 Segmentation of some anatomical structures

2.1 Segmentation of ventricles

Both ventricule in heartMRI

1. With **active contour** :
   $\alpha = 5, \beta = 5, w\_line = 5, w\_edge = -20, \gamma = 0.002$:

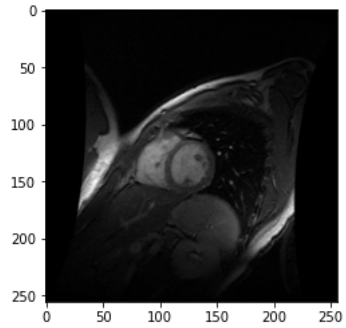

FIGURE 41 – original image

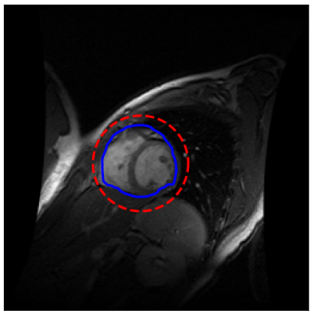

FIGURE 42 – Results

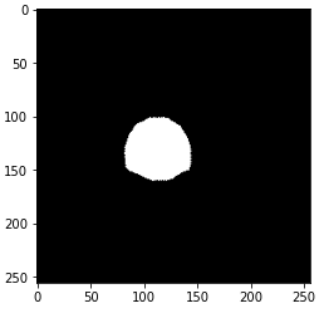

FIGURE 43 – Segmentation mask

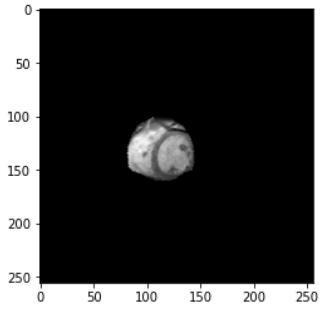

FIGURE 44 – Segmentation visualization

The result is interesting, the algorithm seems to have difficulty in understanding the exact contour at the top of the ventricles but the result seems relevant. We chose to start from an external contour because as the algorithm is based on the attraction by the contours, it would be more difficult to segment the ventricles from an internal initialization given that the 2 ventricles are separated by a contour, it would be necessary to set up 2 initializations but their fusion risks to be more complex, especially as the representation is parametric. The simplest solution here is to start from an external contour.

2. With **chan_vese** :
   Using the chan_vese algorithm based on the homogeneity criterion of the regions, one can think of simplifying the image to make it more homogeneous. For this we can use morphological operations, for example the sequential altered filter. We have chosen not to reconstruct between iterations in order not to preserve the black valleys within the ventricles which would disturb the homogeneity of the regions. Here is a relatively interesting result for N=15, with $\lambda_1 = 3, \lambda_2 = 1, \mu = 0.25, tol = 1e-3, dt = 0.5$:

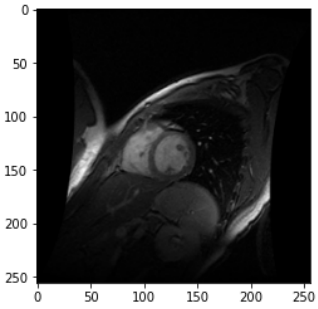

FIGURE 45 – Original image

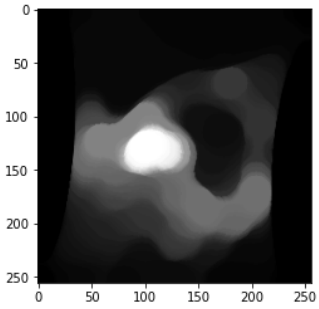

FIGURE 46 – ASF on original image

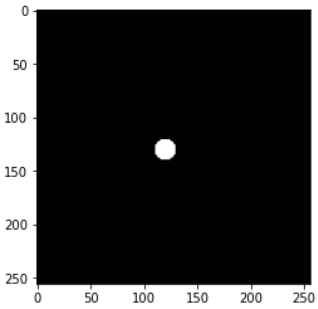

FIGURE 47 – Initial level set

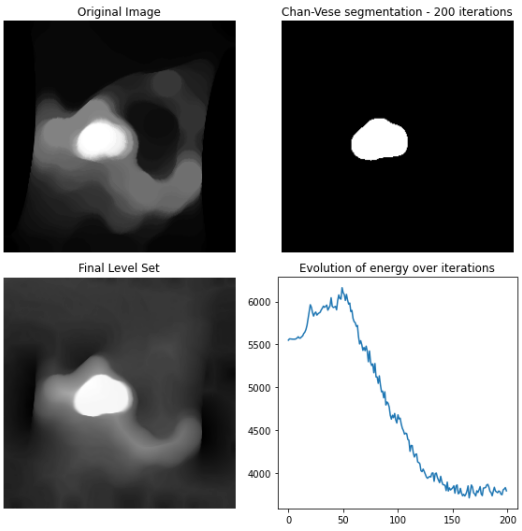

FIGURE 48 – Result

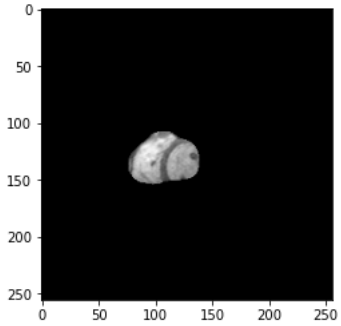

FIGURE 49 – Segmentation visualization

The result is interesting, however it would lack some precision if we wanted to have a perfect segmentation, moreover the calculation is relatively long due to the N iterations of the sequential alternate filter.

Left ventricule in heartMRI

1. With **active contour** :
   Here are 2 results with internal and external initialization :

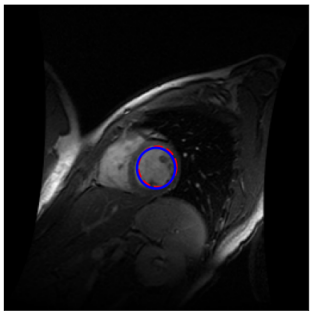

FIGURE 50 – $w\_line = -5$, $w\_edge = 20$, $\alpha = 1$, $\beta = 1$ and $\gamma = 0.001$

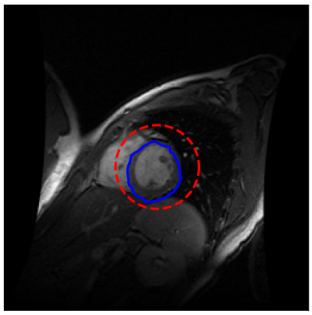

FIGURE 51 – $w\_edge = -20$, $\alpha = 5$, $\beta = 5$ and $\gamma = 0.001$

Depending on whether or not the contour of the ventricle is to be retained, one of the initialization will be better than the other.

2. With **chan_vese** :

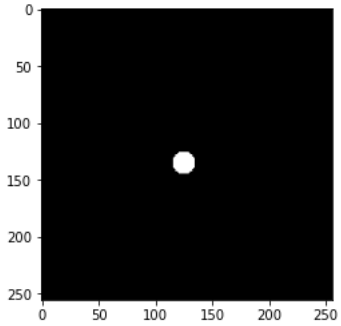

FIGURE 52 – Initial level set

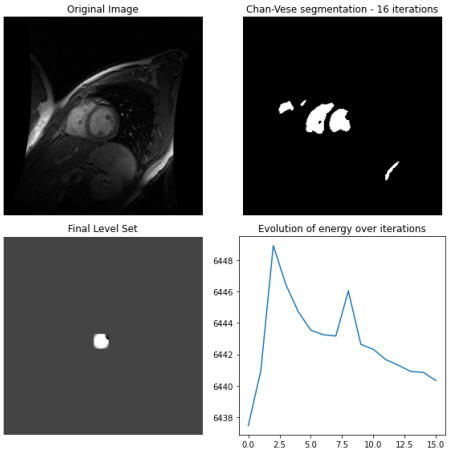

FIGURE 53 – $\lambda_1 = 11, \lambda_2 = 1, \mu = 0.25, tol = 1e-3, dt = 0.5$

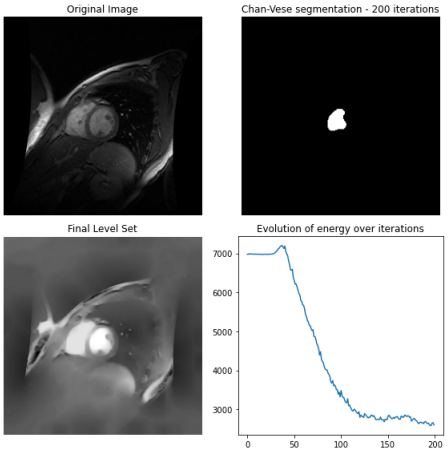

FIGURE 54 – $\lambda_1 = 12, \lambda_2 = 1, \mu = 0.25, tol = 1e-3, dt = 0.5$

One can think of using dilation as a pretreatment to remove the small black spots that impact the final result, however dilation will add too many elements in the object region and make the segmentation of the ventricle even more difficult. It is more difficult in this case to tell the algorithm to segment only one ventricle, as these are approximately on the same level line. The active contour algorithm seems to be more suitable for this segmentation.

2.2 Segmentation of the horn of ventricle

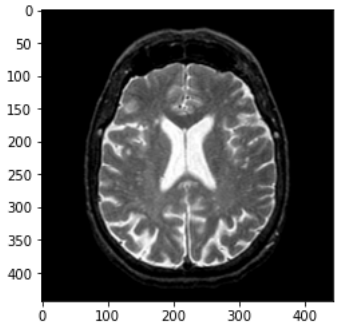

FIGURE 55 – Original image

Let's imagine that we want to segment the middle part. Segmentation based on the homogeneity criterion seems interesting here. We can then start by applying a sequential alternating filter with reconstruction, we have chosen to do a reconstruction step between each iteration because we have no major irregularities in the region to be segmented and we want to preserve the details of the region. This allows us to make the regions of this image more homogeneous :

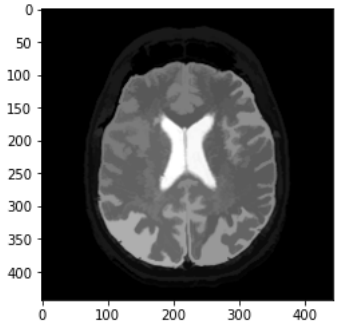

FIGURE 56 – ASF N=10

We will use **Chan_vese** method.
An initial point in the object to be segmented is chosen :

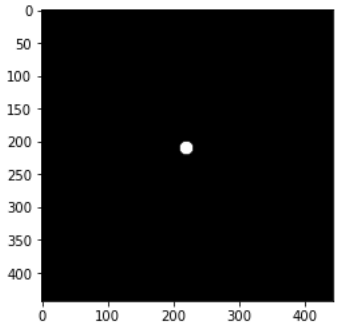

FIGURE 57 – Initial level set

Results with $\lambda_1 = 5, \lambda_2 = 1, \mu = 0.25, tol = 1e-3, dt = 0.5$:

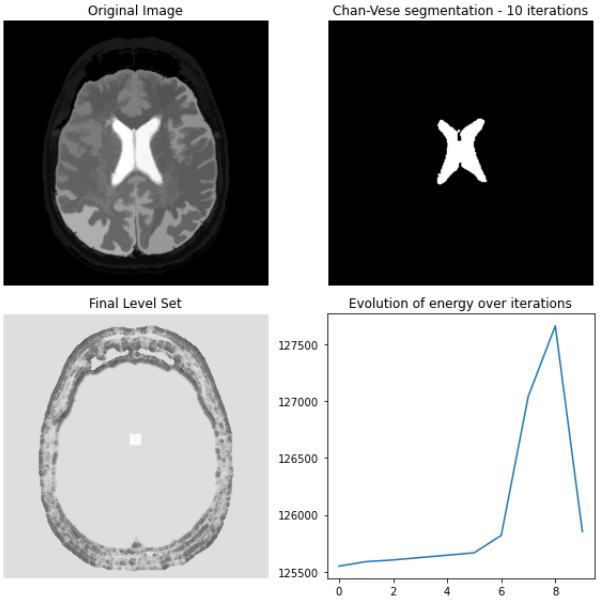

FIGURE 58 – Result

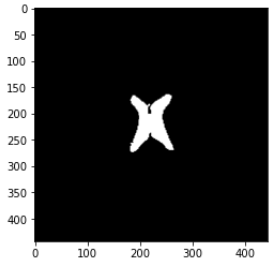

FIGURE 59 – Segmentation mask

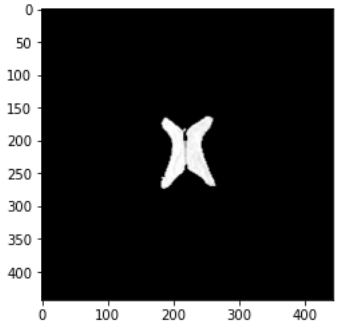

FIGURE 60 – Segmentation visualisation

2.3 Hemispheres of Brain2

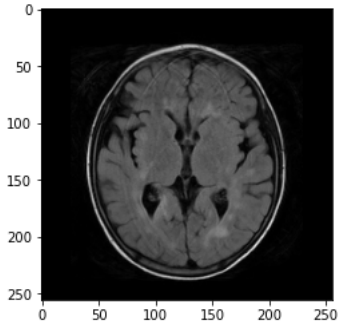

FIGURE 61 – Original image

Let us imagine that we want to separate the hemispheres of the brain. The simplest solution seems to be the **open active contour**. Let us first place a straight line between the two hemispheres as an initial marker :

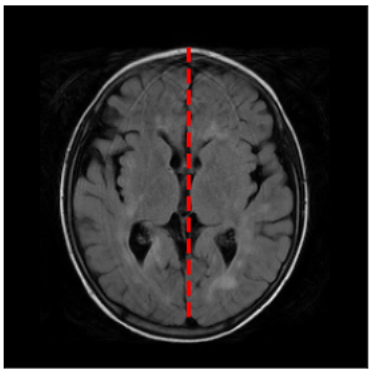

FIGURE 62 – Initial marker

What we would like is a curve that follows the small irregularity of one of the hemispheres to segment them correctly, here is the result obtained with $bc = fixed, \alpha = 0.3, \beta = 0, w\_line = 2.5, w\_edge = 30, \gamma = 0.001$:

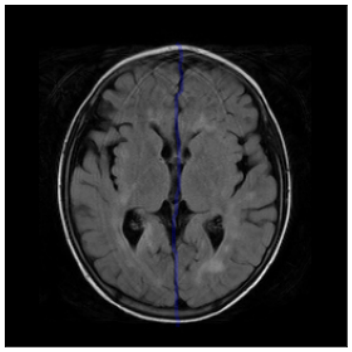

FIGURE 63 – snake result

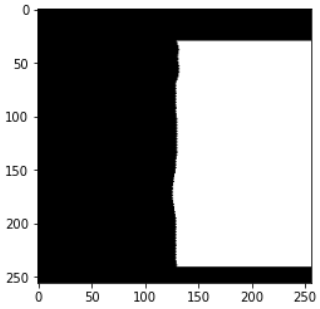

FIGURE 64 – Mask

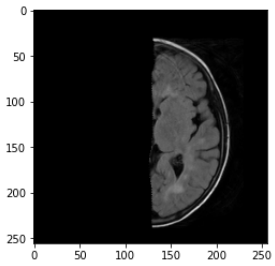

FIGURE 65 – Right hemisphere

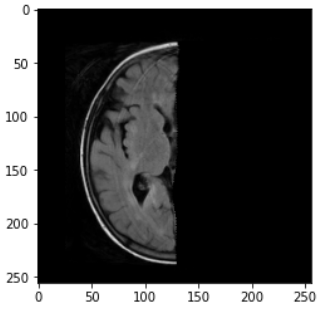

FIGURE 66 – Left hemisphere

2.4 Segmentation of the Brain

Active contour segmentation

We already use this algorithm called active contour segmentation also known as snakes. It is initialized using a user-defined contour or line, around the area of interest, and this contour then slowly contracts and is attracted or repelled by light and edges.

The choices of radius and position of the initial circle have a great impact on the results. In any case the final result is not very precise, indeed we notice from the Fig.67 and Fig.68 that the brain can not be properly segmented because it does not have a circular shape, it would be necessary to use an oval shape to segment it properly.

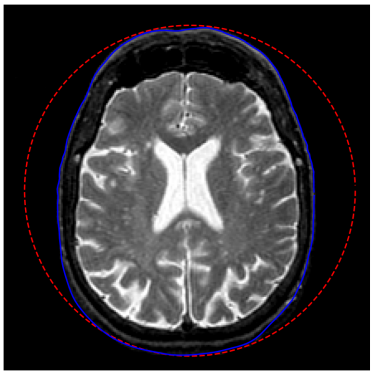

FIGURE 67 – center=(225, 225), radius=200

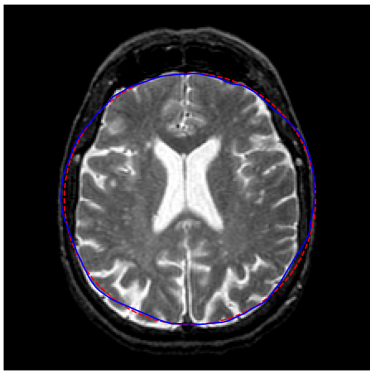

FIGURE 68 – center=(235, 225), radius=152

3 Other methods

3.1 Random walker segmentation

User interactively labels a small number of pixels which are known as labels. Each unlabeled pixel is then imagined to release a random walker and one can then determine the probability of a random walker starting at each unlabeled pixel and reaching one of the pre-labeled pixels. By assigning each pixel to the label for which the greatest probability is calculated, high-quality image segmentation may be obtained.
Since we are working with a supervised algorithm, the position we give it at the beginning is fundamental for the final result. As we can verify from the two figures 69 and 70, the results are very different if we use different sizes for the seed.

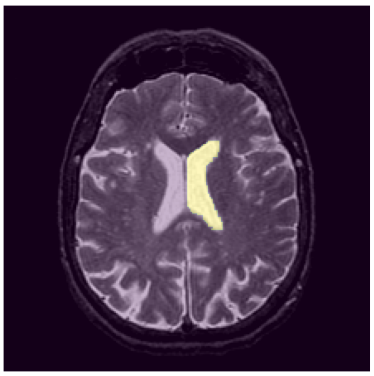

FIGURE 69 – Seed size 5

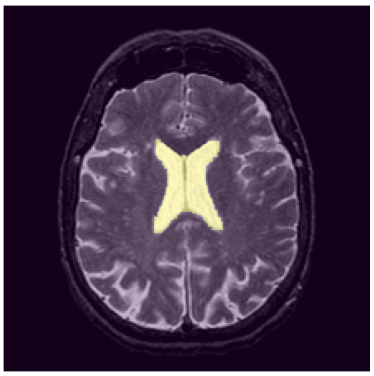

FIGURE 70 – Seed size 8

3.2 SLIC (Simple Linear Iterative Clustering)

SLIC algorithm actually uses a machine-learning algorithm called K-Means under the hood. It takes in all the pixel values of the image and tries to separate them out into the given number of sub-regions.
Since each pixel in the image is now characterized by the value of the class assigned to it, we can create another completely black image, locate the value of the class we are interested in for segmentation, and report in the new image only the values that correspond to those in the class(es) searched for, in white(Fig.72).

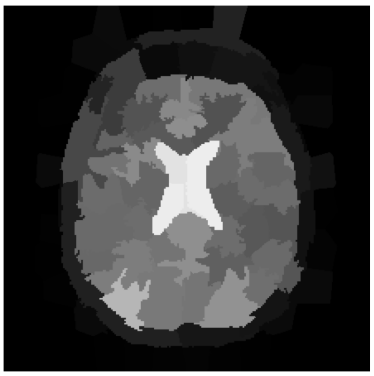

FIGURE 71 – SLIC Method

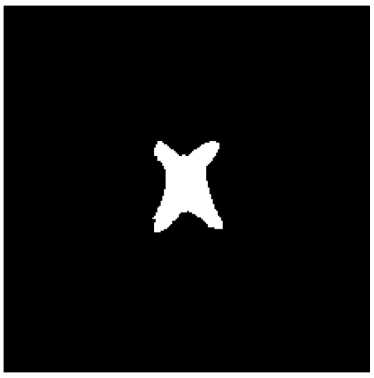

FIGURE 72 – Segmentation

3.3 Felzenszwalb

This algorithm also uses a machine-learning algorithm called minimum-spanning tree clustering under the hood. Felzenszwaib doesn't tell us the exact number of clusters that the image will be partitioned into. It's going to run and generate as many clusters as it thinks is appropriate for that given scale or zoom factor on the image.
Using the mark_boundaries function we can also highlight different segmentation, depending on the value we associated with k during the Felzenszwalb algorithm, the bigger it is the more clusters there will be.

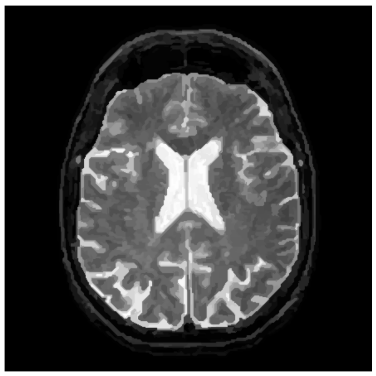

FIGURE 73 – Felzenszwalb result with default k

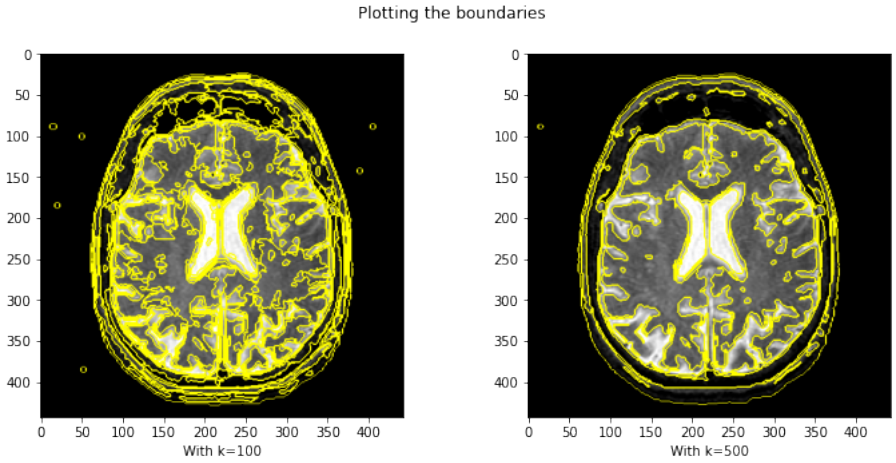

FIGURE 74 – Mark Boundaries on Felzenszwalb