Mona Mokart

TP IMA 203 : modèles déformables

PARTIE 1 : Paramètres

Méthode 1 - contours

- Energie interne : régularisation

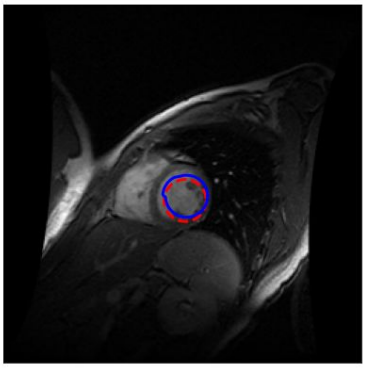
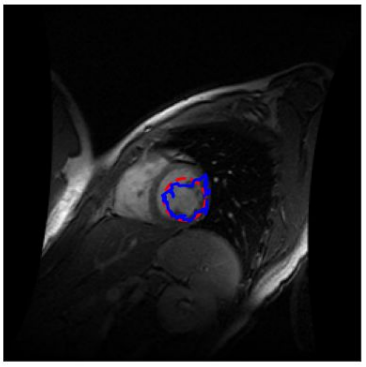
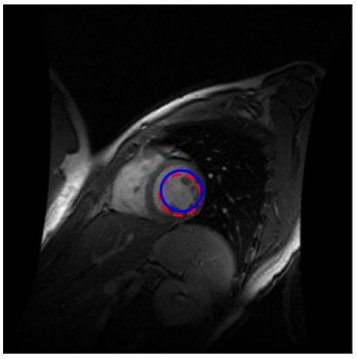
$$E_{interne} = \alpha(s)\left(\frac{dv}{ds}\right)^2 + \beta(s)\left(\frac{d^2v}{ds^2}\right)^2$$

contrôle de la tension et de la courbure ou élasticité (si s = abscisse curviligne, tangente $T = \frac{dv}{ds}$, $||T|| = 1$, et $\frac{dT}{ds} = kN$, k courbure).

Alpha et beta sont des paramètres de régularisation. Ici alpha régule la tension et bêta correspond à la courbure.

Alpha :
Au début on a ces paramètres pour les contours fermés:
snake = active_contour(gaussian(im, 0.01), init, alpha=0.5, beta=30, w_edge=20, gamma=0.001)

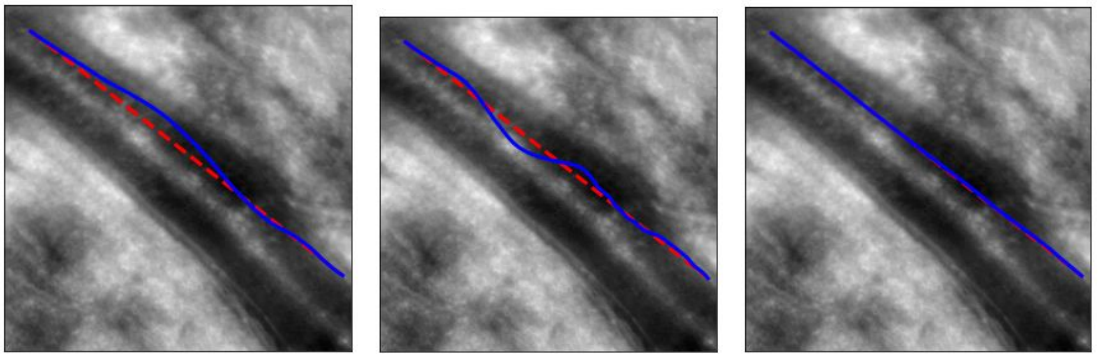
En augmentant alpha on régularise plus la tension donc il y aura moins de tension dv/ds qui correspond à l'ampleur de la variation du contour le long de s. Donc avec alpha élevé on va avoir moins de variation de contour.
Testons:

alpha = 0.5 *alpha=0* *alpha =5*

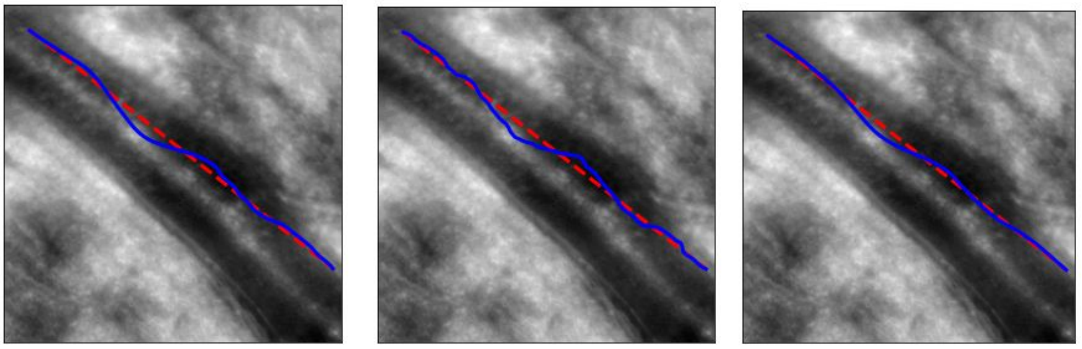
Cela confirme donc ce qu'on dit. Avec alpha=0 on ne régularise plus la tension et on remarque que les contours varient beaucoup le long de s. Avec alpha = 5 on régularise bien, donc on n'observe pas de variation spatiale du contour le long de s

Pour les contours ouverts on remarque la même chose :

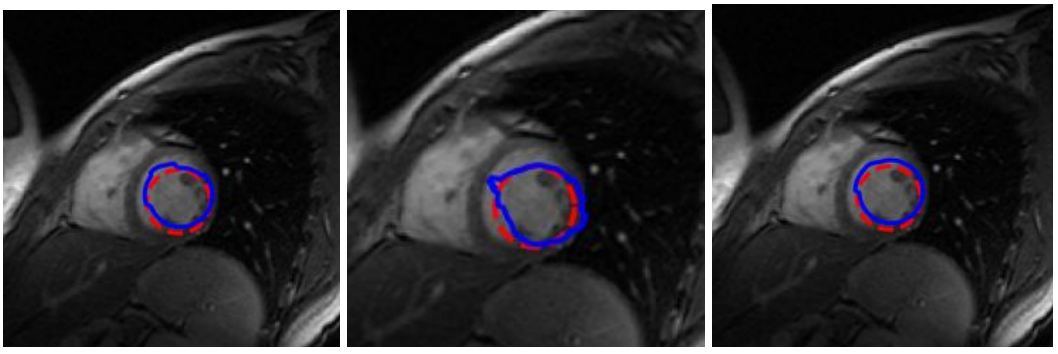


alpha = 0.5 *alpha=0* *alpha =5*

Beta :
Beta est censé réguler la courbure/élasticité c'est-à-dire la variation de la variation du contour. Plus beta est grand, plus il va réguler les courbures.



beta = 2 *beta=0* *beta=10*

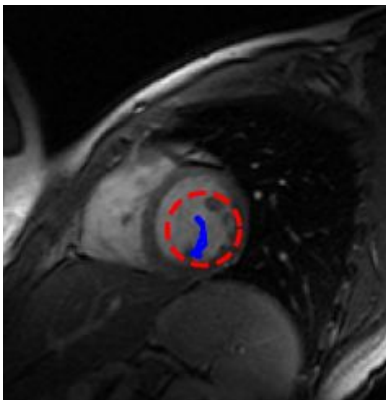
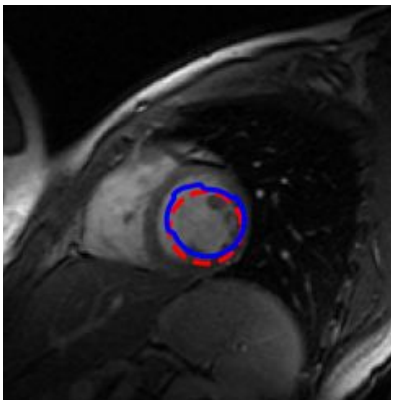


beta =5 *beta=0* *beta=10*

On voit bien qu'en augmentant beta on empêche les courbures mais on n'enlève pas les variations de contours (ce qui correspond à alpha). En cela garde la variation contour mais elle se fait de manière plus lisse avec un beta important.

w_edge : "*Contrôle l'attraction vers les bords*" d'après la documentation

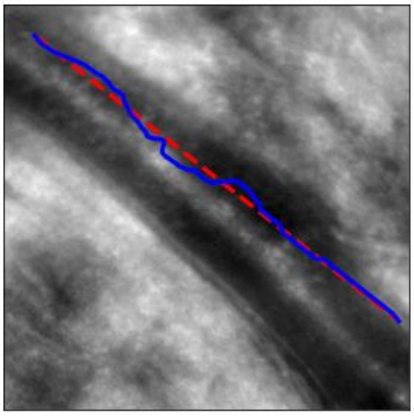
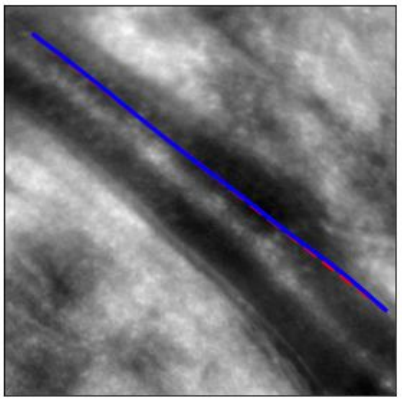
Donc avec wedge grand cela devrait tendre vers des contours,des zones de forts gradient et pour une valeur négative ça devrait tendre vers des zones constantes.
Pour des contours fermés:

w_edge=1 *w_edge=20*

On voit ici que pour w_edge=20 cela détecte les variations de nuances de gris et se place au niveau des taches foncées sur une zone plus claire (correspond à un gradient élevé). Pour w_edge=1 il n'y a pas spécialement d'attraction vers les zones de fort gradient.

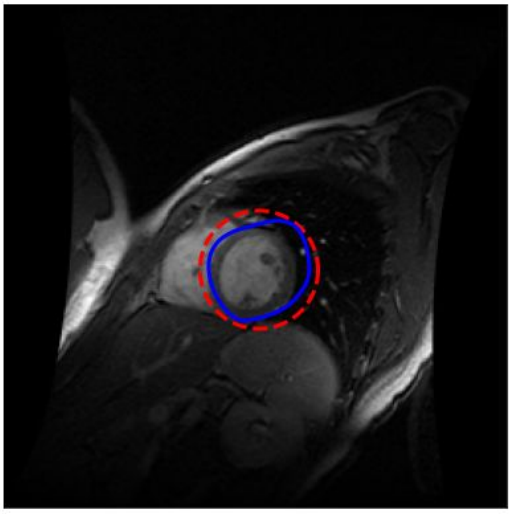
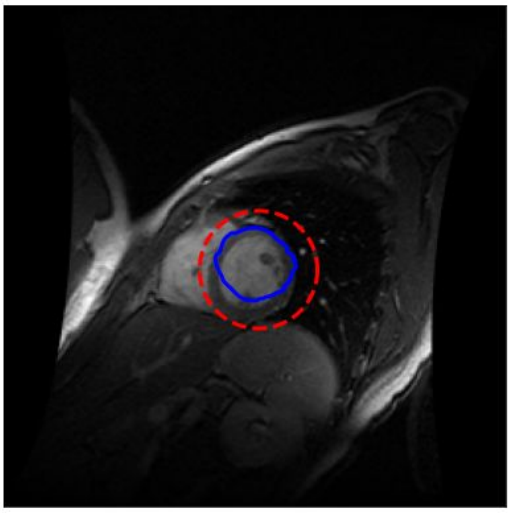
Pour les contours ouverts

w_edge=100 *w_edge = 1*

Ici on voit que pour w_edge très grand que la courbe finale n'est pas lisse du tout, étant attiré par les zones de très fort gradient elle va prendre la forme des tout petits détails, alors que pour w_edge petit la courbe ne va pas détecter ces détails précis et va rester très proche de la forme initiale.

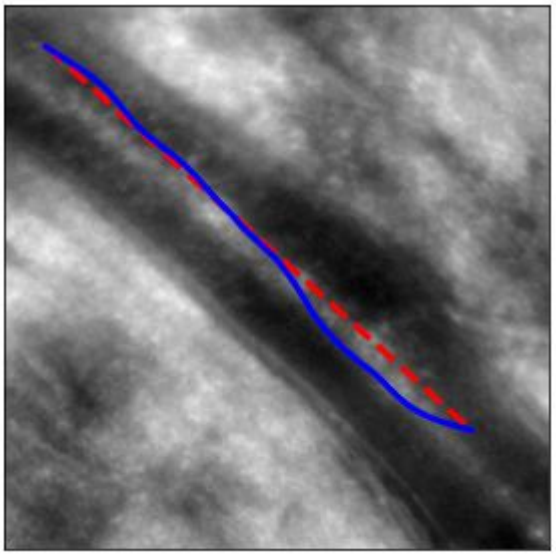
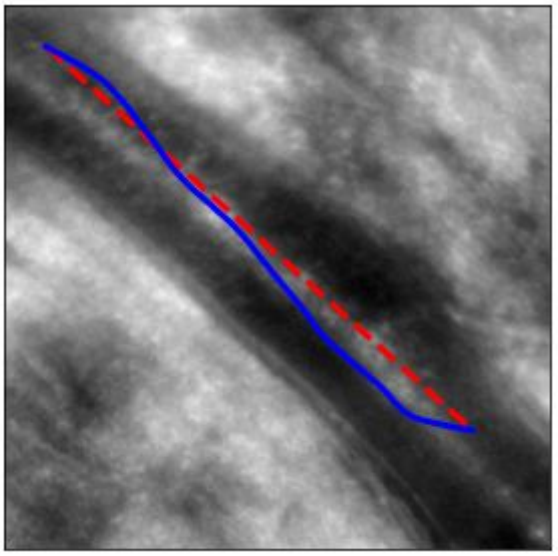
Gamma
Gamma correspond au pas. Ainsi si il est trop grand on risque de ne pas converger vers le minimum de l'énergie. Si il est trop faible le calcul va être long

gamma = 0.8 *gamma = 0.001*

On voit bien que pour gamma grand(0.8) le contour final ne change pas trop du contour initial.

w_line : “Contrôle l'attraction pour la luminosité”

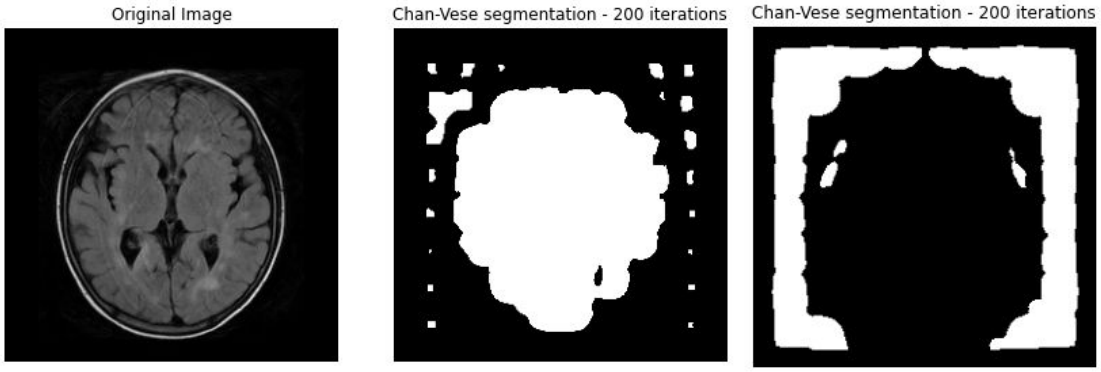
w_line=0 *w_line=5*

On remarque bien que le contour se décale vers la ligne lumineuse quand w_line=5 mais de façon légère.

Méthode 2 - régions

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)

mu:

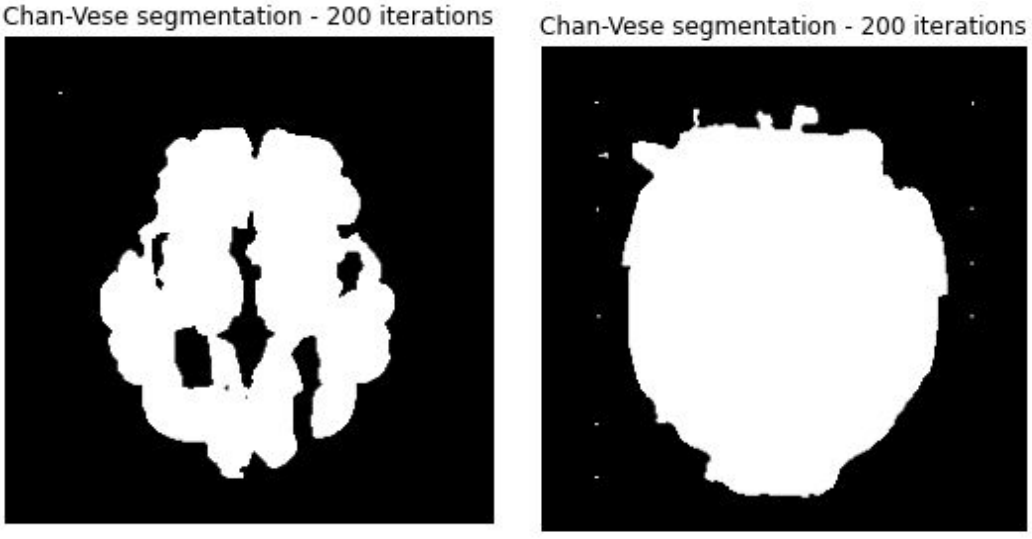


mu =0.5 mu = 0.1

La valeur de mu joue sur la sélection de la zone = 1. Théoriquement plus mu est petit plus l'algorithme va sélectionner des petits objets
Ici en pratique plus elle est grande, plus elle va se concentrer sur le centre, et plus la valeur de mu est petite plus elle va se concentrer sur des zones du fond. Peut-être que les petits objets correspondent aux zones noires dans le cerveau et donc en faisant la segmentation ça sélectionne également le fond qui a une couleur proche des ces petits objets

lambda 1 et lambda 2:

Ces deux paramètres sont liés et servent de rapport pour la surface sélectionnée pour la valeur 1 et la valeur 0. Si lambda1 < lambda2 la région de la classe 1 sera plus large que celle de la classe à et inversement.
Les lambdas régularisent l'étendue de leur classe correspondante.

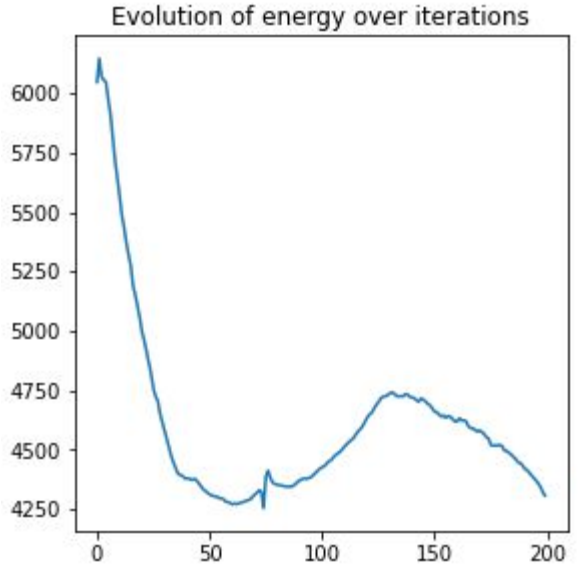


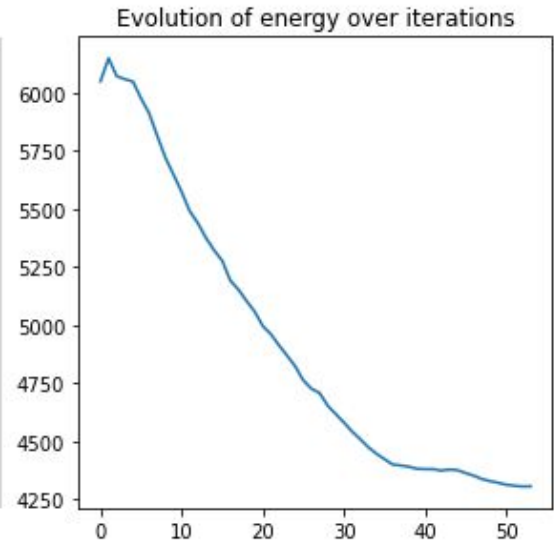
lambda1=5, lambda2 = 1 lambda1=3, lambda2 = 1

Ainsi en diminuant le rapport lambda1/lambda2 on augmente la proportion zone1/zone2

tol et max_iter:

Ce sont deux conditions d'arrêt. Tol est un epsilon sur la différence de l'image entre les deux dernières itérations. Donc si on impose tol petit on cherche un résultat qui converge et proche d'une énergie minimale. max_iter correspond au maximum d'itérations si on atteint pas la valeur de tol. Il permet à la fois d'empêcher que l'algorithme tourne trop longtemps mais aussi de jouer sur la convergence car en augmentant le maximum d'itération on se rapproche d'un minimum d'énergie.



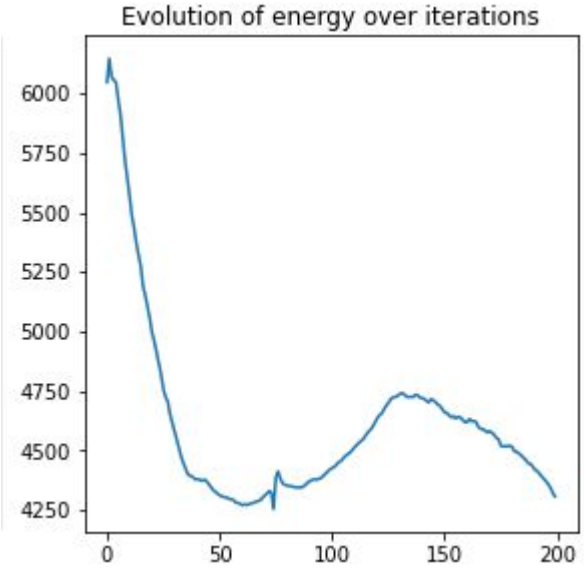


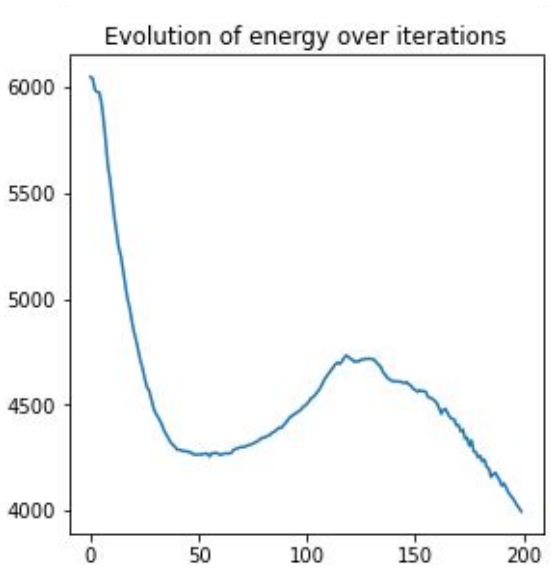
tol=1e-3, max_iter=200 tol=1e-2, max_iter=200

On voit bien qu'avec tol=1e-2 on s'arrête avant le maximum d'itération. Cependant on a une énergie qui n'est pas minimale et donc un résultat non satisfaisant:

dt : d'après la documentation : "A multiplication factor applied at calculations for each step, serves to accelerate the algorithm"

Il s’agit donc d’un “step size” qui lorsqu’il est trop petit ralentit la convergence et quand il est trop grand risque de ne pas faire converger correctement.





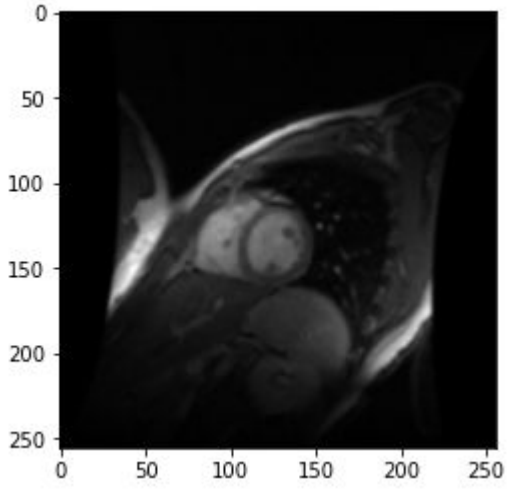
dt =0.5 dt = 1

On voit ici qu’en augmentant dt, on converge plus vite : pour un même nombre d'itération on obtient une énergie plus faible.

PARTIE 2 : Segmentation

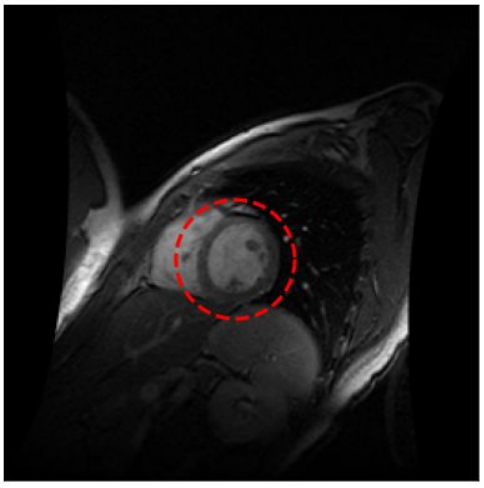
J’ai testé la détection de contour du ventricule sur l’image “coeurIRM.bmp”

- Premièrement j’ai changé la taille du noyau gaussien du filtre appliqué à l’image. J’ai choisi une taille de 1.



Ça permet de lisser les nuisances, c'est-à-dire les petits détails qui peuvent nuire à la forme des contours. Et à l'œil nu ici on peut toujours voir le contour qui correspond au ventricule donc l’algorithme devrait aussi pouvoir.

- Ensuite j’ai changé la zone initiale, je l’ai élargie pour qu’on puisse au sein de cette zone détecter correctement le ventricule.



*s = np.linspace(0, 2*np.pi, 100)*
*r = 137 + 32*np.sin(s)*
*c = 125 + 32*np.cos(s)*
init = np.array([r, c]).T

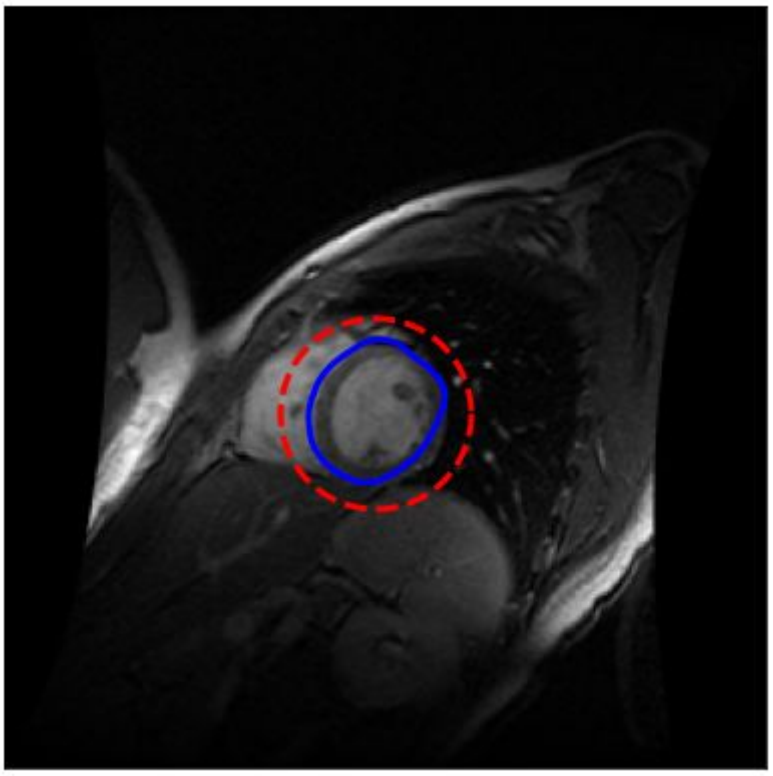
- Finalement : les paramètres de active_contour:
    - Pour laisser libre à la forme du contour finale j’ai régularisé très peu la tension (T=dv/ds), ainsi j’ai imposé alpha= 0.015

- Pour avoir un contour propre et lisse j'ai beaucoup régularisé la courbure (dT/ds), ainsi j'ai imposé beta = 20

- Pour que ça détecte les contours, donc les zones fortes de gradient sans non plus détecter les détails trop précis j'ai imposé w_edge=1.7 (w_edge=1 ça ne detecte pas du tout le contour du ventricule et donc donnait un résultat incohérent, wedge>2 cela détecte trop les contours en détail ce qui donnait un résultat pas très propre)

- Pour gamma, j'ai laissé gamma =0.001 car en l'augmentant on se rapprochait trop de la forme initiale même si cela donnait un résultat correct. Je ne voulais pas que la détection du contour soit faussée par le fait que la forme initiale ressemble au contour qu'on doit trouver, je voulais que ça converge vraiment.

Résultat final :



Le résultat est correct mais la partie droite pourrait être mieux, cela ne suit pas exactement le tour du ventricule. Je pense que c'est dû au fait que la couleur dans cette zone ressemble à celles des pixels proches du ventricule mais ne faisant pas partie du ventricule (zone foncée à droite du ventricule). Il faudrait peut-être faire une égalisation d'histogramme pour pouvoir bien distinguer les zones de niveau de gris proche, ou bien faire une segmentation par k-means toujours dans le même but.