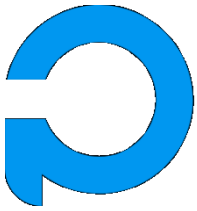


POLYTECH°
NANCY

Loria
Laboratoire lorrain de recherche
en informatique et ses applications

RAPPORT DE STAGE DE 4A

Segmentation d'images médicales en 2D et 3D par contours actifs

Polytech Nancy
LORIA – Laboratoire Lorrain de Recherche en Informatique et Applications

Clémence BIGEARD

Juin-Juillet 2019

Remerciements

Je remercie tout d'abord Jean-Yves Marion, directeur du Loria ainsi que l'ensemble du personnel pour leur accueil et pour m'avoir permis d'effectuer ce stage au sein de leurs locaux.

Je remercie également grandement Vincent Despré, professeur à Polytech Nancy, pour m'avoir aidée dans mes recherches de stage.

Merci à Fabien Pierre et Pierre-Frédéric Villard, mes maîtres de stage, pour leurs conseils et leur aide précieuse qu'ils ont apporté tout au long de ce stage.

Merci enfin à tous les membres du bureau C037 pour leur accueil et leur sympathie durant ces semaines de stage.

Table des matières

Introduction

Un contour actif, aussi appelé *snake*, est une structure dynamique utilisée en traitement d'image modélisé par une courbe déformable qui épouse la forme des objets. Il intervient notamment dans la segmentation d'image en permettant de séparer une forme du reste de l'image. Cette méthode qui peut être utile dans plusieurs domaines peut notamment être utilisée dans le milieu médical pour segmenter des organes sur des images médicales en 2D mais également en 3D.

L'objectif de ce stage est de développer sur Matlab des méthodes permettant de segmenter des images médicales en 2D ou 3D. Dans le cadre de projets précédents, des algorithmes ont déjà été implémentés pour réaliser ces segmentations. Il faut donc partir de cette base et la comprendre pour l'améliorer et mettre en place d'autres méthodes. Il faut également effectuer plusieurs tests pour appréhender l'influence des différents paramètres.

Nous verrons donc plusieurs méthodes qui ont été implémentées pour améliorer ces contours actifs d'abord en 2D puis en 3D

1. Etude de l'existant

Les simulateurs informatiques sont de plus en plus utilisés pour aider à la formation de chirurgiens ou de médecins. On peut ainsi simuler de nombreuses procédures sans avoir recours à un patient réel. Cependant ces simulateurs utilisent des anatomies humaines classiques, ils ne s'adaptent pas à la morphologie d'un patient en particulier. Des recherches se focalisent sur ce point pour créer un simulateur basé sur un patient en particulier [1, 2, 3]. L'un des problèmes actuels est de réussir à segmenter un organe en 3D. Différentes techniques existent et sont basées sur l'utilisation d'un atlas anatomique [4] ou sur du *machine learning* [5]. Elles reposent donc sur l'utilisation de bases de données conséquentes. Nous allons donc nous concentrer sur des techniques nécessitant moins de données.

On considère que l'image est définie par une grille régulière de pixels en 2D (respectivement des voxels en 3D). Le but de la segmentation d'image est le calcul de N ensemble de pixels tel que chaque set corresponde à une zone de l'image. Ces zones sont calculées en fonction des contours, des formes et des statistiques de leurs pixels.

On peut distinguer deux types de segmentation d'images. Le premier est basé sur la grille de pixels qui définit une image. Le second est basé sur la discrétisation de la limite des sets de pixels.

Le premier type a été largement utilisé, notamment avec des seuillages d'image [6]. L'une des méthodes les plus connues est la méthode d'Otsu, basée sur les histogrammes [7]. Le second type est utilisé pour la segmentation d'images par Michael Kass [8] dans une méthode appelée *snake* basée sur la minimisation d'une énergie. Ce concept a été étendu en 3D par Terzopoulos [9].

2. Contours actifs sur images 2D

2.1. Modèle de contours actif en 2D

Commençons par expliquer la résolution théorique en 2D. L'objectif du contour actif va être de trouver et délimiter une zone précise dans une image.

On part d'un contour initial fixé qui peut être dessiné par l'utilisateur. Ce contour va se rapprocher de l'image au fil des itérations. Pour assurer cette convergence, différentes forces sont mises en œuvre et l'idée principale consiste à minimiser une somme d'énergies.

Voici l'équation finale utilisée pour passer d'une itération à l'autre donc du contour V_{t-1} au contour V_t :

$$V_t = (\gamma A + I_d)^{-1}(V_{t-1} + \gamma(\nabla(\mathrm{P}(\mathrm{V}_{t-1})) + F_B))$$

Avec :

- γ le pas de temps de la discrétisation temporelle d'Euler.
- V_t le vecteur V au temps t.
- A la matrice de régularisation qui dépend de 2 autres paramètres α et β
- $\nabla(P(V_{t-1}))$ le gradient du potentiel de l'image qui correspond à la force gradient de l'image pour attirer le contour.
- F_B la force de ballon qui contracte le contour vers l'intérieur.

Nous allons revenir sur plusieurs de ces paramètres pour expliquer et comprendre leur influence.

2.2. Explication et Influence des paramètres :

Pour mieux comprendre l'influence des différents paramètres, une série de test a été appliquée à différentes images 2D. Les valeurs par défaut sont :

Alpha=1750 Beta=500 Gamma=0.0001 Fb=-50 Sigma=1

- Alpha α :
  Alpha est un paramètre de lissage du contour actif. S'il est trop faible, cela cause l'apparition de boucles sur le contour (appelées aussi des oreilles de Mickey). S'il est trop grand, on perd en précision et les contours de l'image ne sont plus respectés.

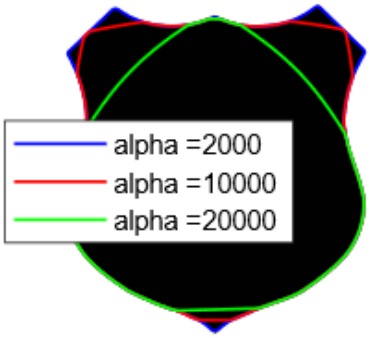


Figure 1 : Tests sur alpha

- Beta β :

  Beta (et alpha) sont des paramètres qui contrôlent la régularité du contour. En théorie beta favorise les faibles courbures et agit sur le lissage du contour. Pour savoir si Beta influe sur le résultat, nous avons mené les tests suivants avec des valeurs assez extrêmes pour les paramètres.

Alpha = 17.5	Alpha = 17.5
Beta = 4	Beta = 40000
Sigma = 1	Sigma = 1
Gamma = 0.0002	Gamma = 0.0002
BaloonCoeficient = -0.9	BaloonCoeficient = -0.9
MaxIteration = 8400	MaxIteration = 1250

Figure 2 : Résultat pour beta=4

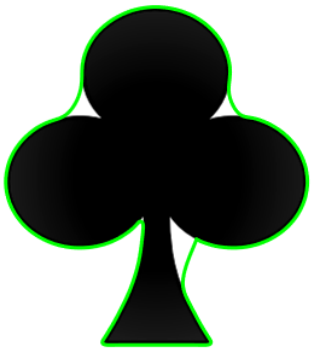


Figure 3 : Résultat pour beta=40000

Sur les résultats ci-dessus, on peut voir que le contour est plus lisse sur l'image de droite. Sur celle de gauche, pour un beta plus petit, les angles sont plus marqués. Ainsi Beta influe bien sur le résultat final en lissant les contours mais cela n'est visible que pour des jeux de paramètres particuliers.

- Sigma σ :

Ce paramètre correspond au flou gaussien. Le flou gaussien dépend de l'image. Il permet au contour de la forme à épouser à attirer le *snake* vers lui mais s'il est trop élevé, c'est un élément à l'intérieur de la forme qui risque d'attirer le *snake*. Sur la première image ci-dessous, plus σ est élevé et plus le contour actif est attiré vers l'intérieur de l'image, on le voit particulièrement sur la patte arrière du chat. Sur la seconde image, si sigma est plus élevé, le contour sera plus proche du chat sur la droite.

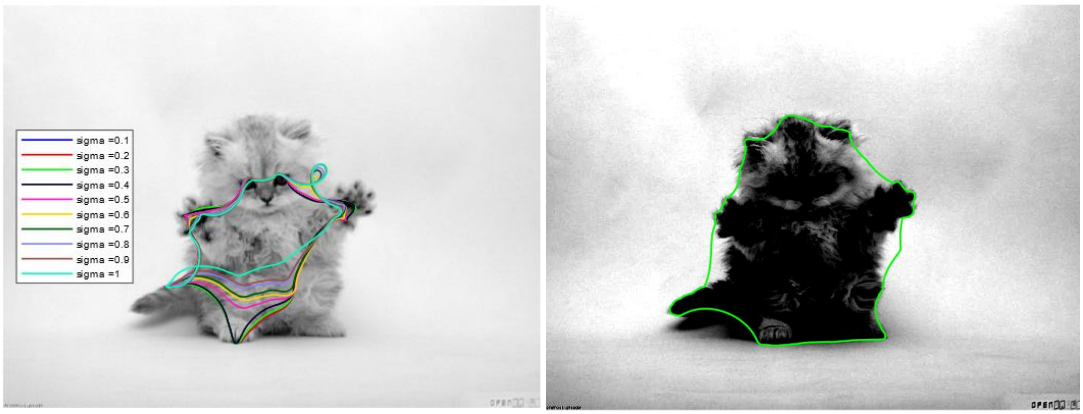


Figure 4 : Segmentation d'un chat

- Gamma :
  Gamma représente la vitesse de convergence. Trop faible, le programme risque d'être trop lent. Trop haute, la convergence n'est pas stable. Ci-dessous, la courbe n'est pas représentée lorsque la convergence n'est pas stable.

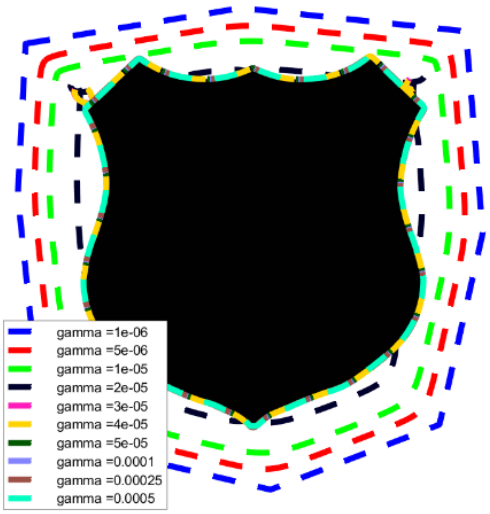


Figure 5 : Tests sur Gamma

- Fb :
  La force Ballon permet d'éviter que le contour actif ne se stabilise loin de l'image et applique des forces de convergences sur le contour. Il entre en action lorsque le gradient de l'image n'influe pas suffisamment sur le contour. Plus le coefficient kb est fort, plus la force de ballon est forte, le signe de kb détermine si la force de ballon est dirigée vers l'extérieur ou l'intérieur. Sur l'image ci-dessous, kb est négatif. Si kb est trop faible, le contour restera à l'extérieur de l'image. Si kb est trop grand, le contour migrera à l'intérieur de l'image.

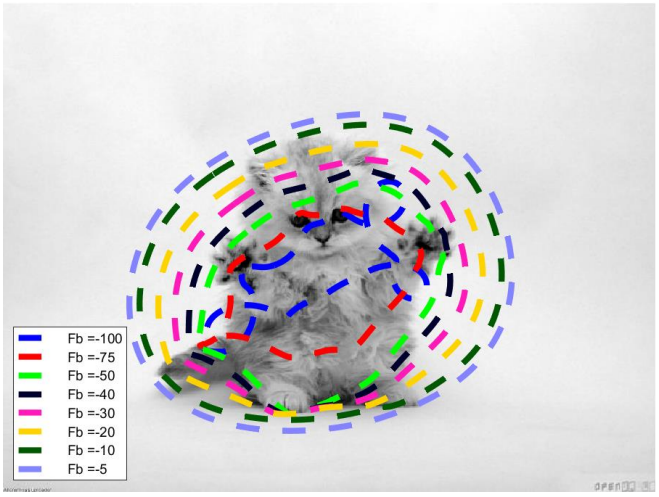


Figure 6 : Tests sur la force Ballon

- Itérations : on peut également modifier le nombre d'itérations. Plus il y a d'itérations, plus l'exécution du programme sera long. S'il n'y en a pas assez, le *snake* n'aura pas le temps de converger. Le nombre d'itération fait partie des critères d'arrêt. Il est également possible d'utiliser un critère d'arrêt basé sur la convergence.

2.3. Subdivision adaptative

Actuellement la subdivision du contour actif est assez simple. A chaque distance de k pixels, un point est créé, l'algorithme est ensuite appliqué pour que le contour converge vers l'image, le nombre de points restant constant. On souhaite mettre en place une subdivision adaptative, c'est-à-dire que des points seront rajoutés ou supprimés au fur et à mesure de l'évolution du contour actif.

En effet, lorsque la densité de points est trop haute, le risque d'apparitions d'oreilles de Mickey augmente. Lorsque celle-ci est trop basse, le *snake* a des difficultés pour segmenter précisément l'image. Une subdivision adaptative permettrait notamment au *snake* de s'aventurer plus loin lorsqu'il y a des concavités importantes.

Figure 7 : Problème actuel avec les concavités importantes

Pour rajouter ou retirer des points le long du contour, des *splines* ont été utilisées. Le but étant d'obtenir une représentation continue du contour.

Nous utilisons la formule d'interpolation suivante [10] :

$$\gamma(t) = a\begin{pmatrix} X_j \\ Y_j \end{pmatrix} + b\begin{pmatrix} X_{j+1} \\ Y_{j+1} \end{pmatrix} + c\begin{pmatrix} X"_{j+1} \\ Y"_{j+1} \end{pmatrix} + d\begin{pmatrix} X"_{j+1} \\ Y"_{j+1} \end{pmatrix}$$

Avec :

- (X_i,Y_i) une suite de points
- Une courbe paramétrée Υ : [0,1]→R^2 qui vérifie $\forall 0 \leq i \leq N, \gamma(t_i) = \begin{pmatrix} X_i \\ Y_i \end{pmatrix}$
- $a = \begin{pmatrix} A & 0 \\ 0 & A \end{pmatrix}$ $b = \begin{pmatrix} B & 0 \\ 0 & B \end{pmatrix}$ $c = \begin{pmatrix} C & 0 \\ 0 & C \end{pmatrix}$ $d = \begin{pmatrix} D & 0 \\ 0 & D \end{pmatrix}$
- $A = \frac{t_{j+1}-t}{t_{j+1}-t_j}$ $B = \frac{t-t_j}{t_{j+1}-t_j}$ $C = \frac{1}{6}(A^3 - A)(t_{j+1} - t_j)^2$ $D = \frac{1}{6}(B^3 - B)(t_{j+1} - t_j)^2$
- $\forall 0 \leq i \leq N, t_i = i * \frac{1}{N}$

On peut également obtenir la longueur par arc d'une *spline* par la formule :

$$L = \int_0^1 ||\gamma'(t)||dt$$

Au fil des itérations, la distance entre les points voisins sera calculée et si celle-ci dépasse un paramètre m donné, un point sera ajouté entre les points concernés. Si la distance est inférieure à m/3, le point est retiré. Les points ajoutés sont déterminés à l'aide de la *spline* cubique.

En appliquant cette subdivision adaptative, nous obtenons ceci :

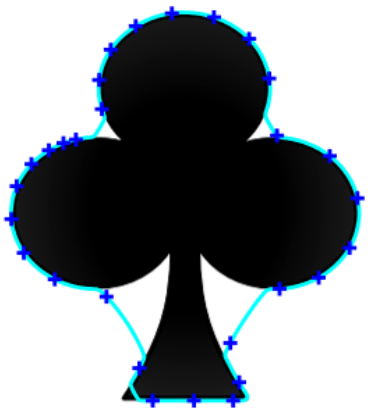


Figure 8 : Résultat sans subdivision adaptative

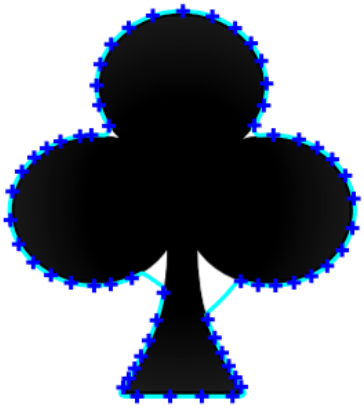


Figure 9 : Résultat avec subdivision adaptative

Pour une meilleure lisibilité ci-dessus, un marqueur a été placé tous les 20 points. On voit que la subdivision adaptative permet une répartition un peu plus homogène des points et permet également au *snake* d'aller un peu plus loin dans les concavités fortes.

La valeur de l'écart inter-point m doit se situer entre 1 et 3. Trop faible, des oreilles de Mickey apparaissent (le contour actif forme des boucles). Trop grande, on perd en précision. Ici m=2.

2.4. Moyennage de la force Ballon

Nous avons vu plus haut que la force ballon aidait le contour actif à converger. Cependant, il arrive parfois que le contour actif forme une ou plusieurs boucles (aussi appelées oreilles de Mickey). Sur l'image ci-dessous, on peut constater la présence de trois boucles.

Figure 10 : Contour initial

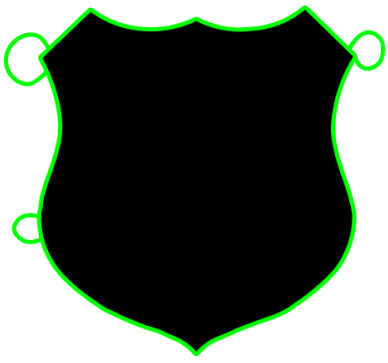


Figure 11 : Résultat obtenu avec des oreilles de Mickey

Ce phénomène est lié aux valeurs des paramètres initiaux mais aussi à la définition du contour initial. De plus, la force de ballon contribue au développement de cette boucle en la poussant à s'agrandir au lieu de se résorber.

La force ballon est définie normale au contour actif et prend le relai de la force gradient lorsque celle-ci n'est pas suffisante pour faire converger le contour. Lorsqu'une boucle apparaît, la normale au contour est dirigée vers l'extérieur ce qui pousse la boucle à s'agrandir.

De plus, les vecteurs de la force ballon étant définis normaux au contour, ils se croisent. Nous souhaitons atténuer ce phénomène de croisement grâce au moyennage des forces en conservant le rôle d'attraction des vecteurs.

Pour cela, nous utilisons une convolution circulaire. Chaque vecteur va alors être moyenné par ses voisins. Voici l'équation utilisée :

$$F_f = F * P_f = \Gamma^{-1}(\Gamma(F) * \Gamma(P_f))$$

Ce calcul étant mis en place, nous allons essayer de visualiser son efficacité.

Nous appliquons la force Ballon sur un contour initial et affichons les vecteurs de cette force. Dans le premier cas, nous avons appliqué la force ballon classique, dans le second, nous avons appliqué la force Ballon avec le moyennage des forces.

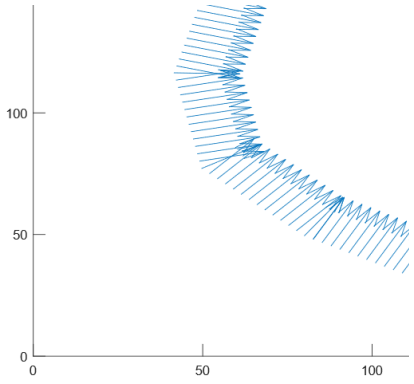


Figure 12 : Sans moyennage des forces

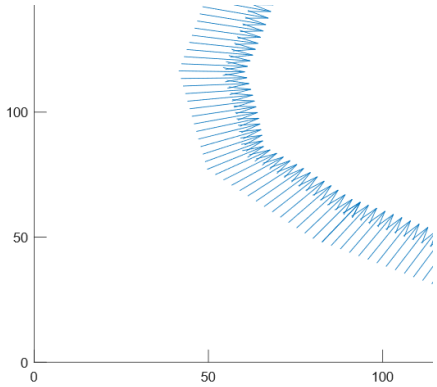


Figure 13 : Avec moyennage des forces

On peut noter que les vecteurs ne semblent plus se croiser lorsque le contour change de direction après application du moyennage.

La fonction qui réalise le moyennage de la force ballon possède deux paramètres :

- La force Ballon elle-même
- Sigma, le paramètre de la décroissance exponentielle

On peut donc jouer sur le paramètre sigma pour voir l'influence de ce paramètre. En faisant des tests dans les mêmes conditions initiales, on peut s'apercevoir que plus sigma est grand, plus les oreilles de Mickey ont tendance à disparaitre.

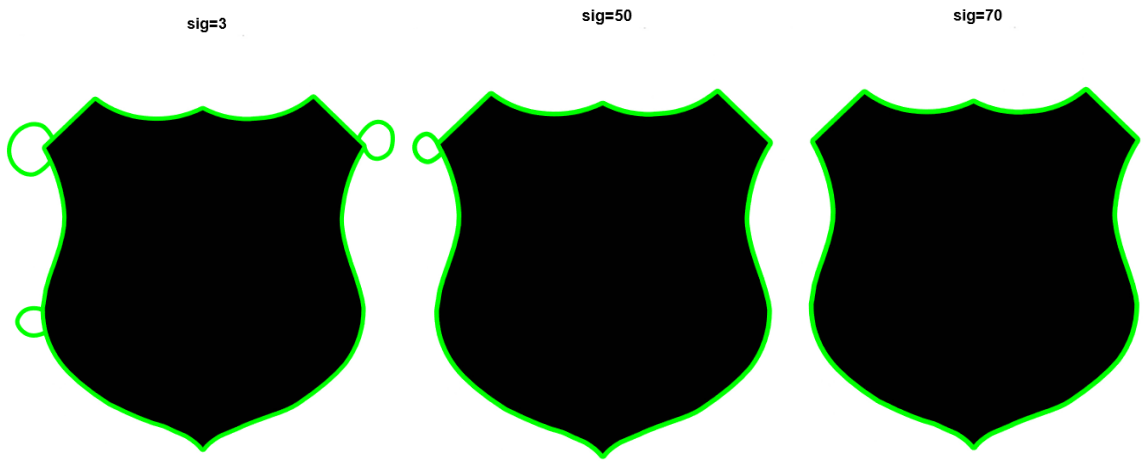


Figure 14 : Evolution des oreilles de Mickey en fonction de sigma

Le moyennage des forces de ballon permet donc de diminuer l'apparition des oreilles de Mickey si le paramètre sigma est bien fixé.

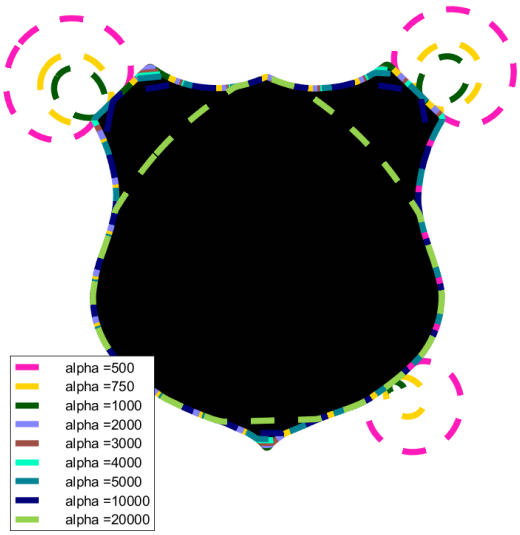


Figure 15 : Sans le moyennage de la force Ballon

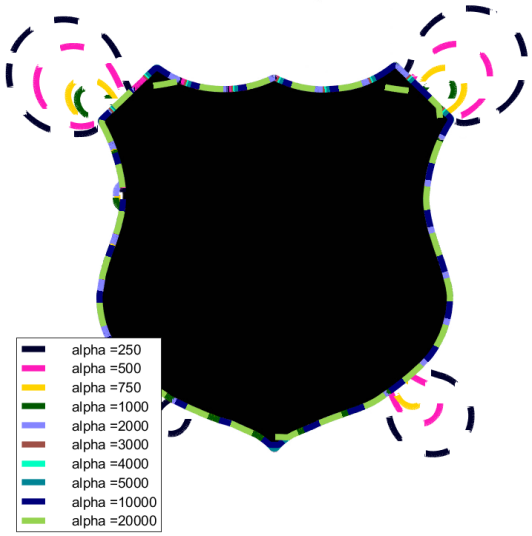


Figure 16 : Avec le moyennage de la force Ballon

Sur l'image de gauche différentes valeurs de alpha ont été testées sans le moyennage de la force Ballon, à droite, les mêmes tests ont été réalisés avec le moyennage et sigma=70. Si on compare les tracés pour les mêmes valeurs, par exemple le tracé rose pour alpha=500, on peut voir que le moyennage a diminué la taille des oreilles de mickey. On a également un résultat plus précis pour alpha=20000.

2.5. Modification du contraste

Pour certaines images, il serait intéressant de modifier la luminosité et le contraste de l'image pour avoir de meilleurs résultats sinon le contour a des difficultés à détecter la différence de gradient. Pour éviter une modification manuelle du contraste des images, des fonctions déjà présentes dans Matlab ont été utilisées.

image=imadjust(image,stretchlim(image,0.1));

- La fonction imadjust sature les valeurs de certains pixels. Par défaut, ce sont 1% des pixels aux valeurs les plus basses et 1% des pixels aux valeurs les plus hautes qui seront saturés.
- La fonction stretchlim permet de choisir la proportion de pixels choisis pour la saturation, ici nous avons opté pour 10%.

Cet ajustement du contraste permet d'obtenir de meilleurs résultats sur des images où le contour se fond parfois avec le fond, comme l'image du chat ou du vase.

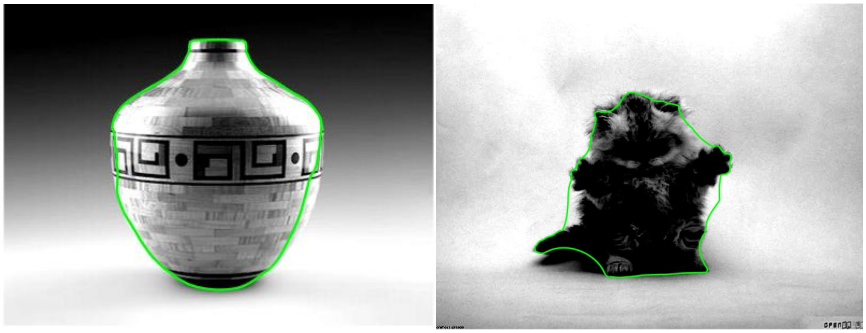


Figure 17 : Résultats après ajustement du contraste

2.6. Application sur des images médicales

L'objectif à long terme est d'utiliser ces algorithmes sur des images médicales, en particulier pour isoler les poumons ou le diaphragme, il faut donc le tester sur des images médicales.

La subdivision adaptative, moyennage des forces de ballon et la modification de contraste ont été utilisés.

Selon le contour initial dessiné, soit l'algorithme va trop loin dans les itérations et le contour converge et quitte le contour de l'image, soit au contraire l'algorithme ne va pas assez loin dans les itérations et n'épouse pas suffisamment les cavités de la forme.

Figure 18 : L'image médicale initiale

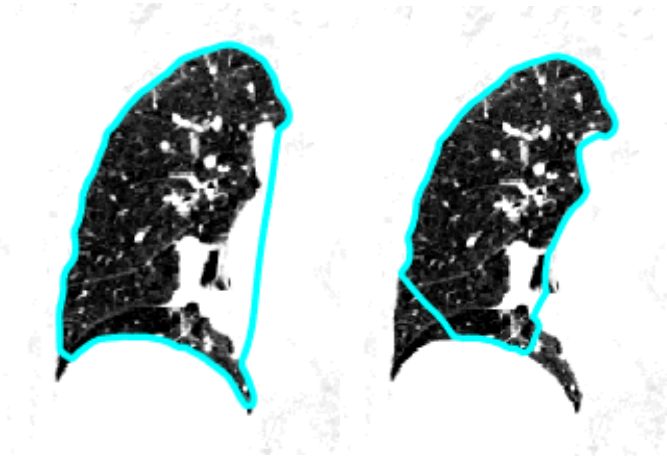


Figure 19 : Problèmes rencontrés

Sur l'image de gauche, le contour n'a pas bien épousé la forme du poumon sur le côté droit.

Il faut donc revoir les conditions d'arrêt de l'algorithme, notamment la valeur epsilon qui calcule la différence du *snake* entre les deux dernières itérations.

Sur l'image de droite, en utilisant les mêmes paramètres et le même contour initial, on diminue uniquement la valeur epsilon (cela permet à l'algorithme d'effectuer plus d'itérations), l'algorithme épouse bien la forme sur la droite du poumon mais le bas du contour finit par quitter sa position et commence à converger vers le centre de l'image. Au fil des itérations le contour aurait complétement quitté l'image et aurait convergé vers un point.

Différentes causes :

- Les conditions d'arrêt sont à revoir : notamment la variable epsilon, si epsilon est trop faible, l'algorithme s'arrête. On peut diminuer la valeur limite d'epsilon ou faire une moyenne d'epsilon sur quelques itérations.
- La force de Ballon : elle doit passer à 0 lorsque le contour est bien sur l'image mais si elle reste à 1, le contour peut quitter l'image et converger. On peut également essayer de modifier les paramètres de la force de Ballon.

Résultats :

- Mise en place du moyennage de epsilon qui se fait sur 10 itérations. Cependant, cela n'est pas suffisant, il a donc également fallu diminuer légèrement sa valeur dans les conditions d'arrêt. Ce qui évite à l'algorithme de s'arrêter trop tôt.
- Pour le problème de la convergence trop forte, la force Ballon ne semble pas en cause. En mettant la force Ballon à zéro et en lançant le programme, le problème est présent, il a juste besoin de plus d'itérations pour se manifester puisque la convergence du contour est plus lente.
- La subdivision adaptative retire des points lorsque ceux-ci sont trop proches. C'est que qui semble causer le problème, des points sont retirés et le contour n'ayant plus d'accroche se replie sur lui-même. Il faut donc régler la distance à partir de laquelle des points sont supprimés. Après quelques tests, la distance minimale a été diminuée.
    - Lorsque la distance inter-point est supérieure à une variable m, on ajoute un point.
    - Lorsque cette distance est inférieure à m/10, on retire un point.

Figure 20 : Résultat obtenu après les modifications

2.7. Tests avec une autre façon de subdiviser

Comme nous l'avons vu précédemment, l'algorithme ajoute ou supprime un point entre deux voisins lorsque la distance inter-voisins est trop grande. Ainsi l'ajout ou la suppression de point était localisé sur une zone précise du contour. On souhaite comparer avec une autre façon de subdiviser : lorsqu'une distance inter-voisins est trop grande, on recalcule avec des *splines* cubiques le contour complet en augmentant le nombre total de points. Ainsi tous les points sont modifiés et équitablement répartis sur le contour. On peut qualifier cette subdivision de générale.

Figure 21 : Résultat avec la subdivision locale

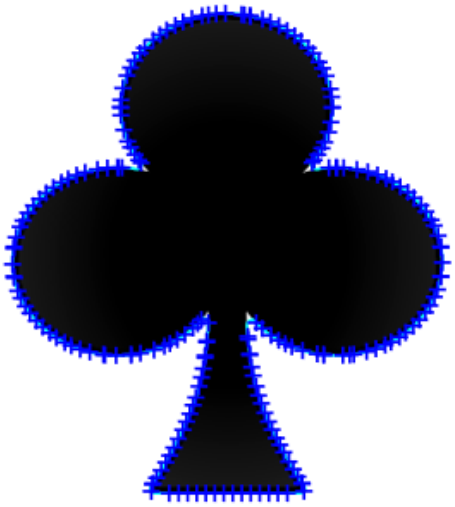


Figure 22 : Résultat avec la subdivision générale

Ci-dessus, un marqueur a été placé tous les 20 points pour une meilleure lisibilité.

On peut voir que la seconde méthode est un petit peu plus précise car elle crée plus de nouveaux points que l'autre méthode, elle a cependant besoin de plus d'itérations avant de s'arrêter. Il faut aussi préciser que ces deux résultats ont été obtenus en utilisant un contour initial très proche du trèfle. Il faudrait donc tester avec des contours plus éloignés et d'autres images pour pouvoir comparer plusieurs résultats.

3. Contours actifs sur images 3D

3.1. Théorie en 3D

Le paramétrage du problème en 3D est similaire à celui en 2D. La principale difficulté est de passer de deux voisins par points en 2D à trois voisins ou plus par point en 3D. Pour cela une matrice d'adjacence est utilisée pour définir la matrice de régulation A.

L'équation utilisée au final reste la même :

$$V_t = (\gamma A + I_d)^{-1}(V_{t-1} + \gamma(\nabla(\mathrm{P}(\mathrm{V_{t-1}})) + F_B))$$

3.2. Tests 3D sur des formes simples

On retrouve les mêmes paramètres en 3D qu'en 2D. Le contour actif en 3D est modélisé par une sphère dont on peut modifier le rayon et le placement dans l'espace. Elle se resserre ou s'agrandit autour de la forme à détourer au fil des itérations.

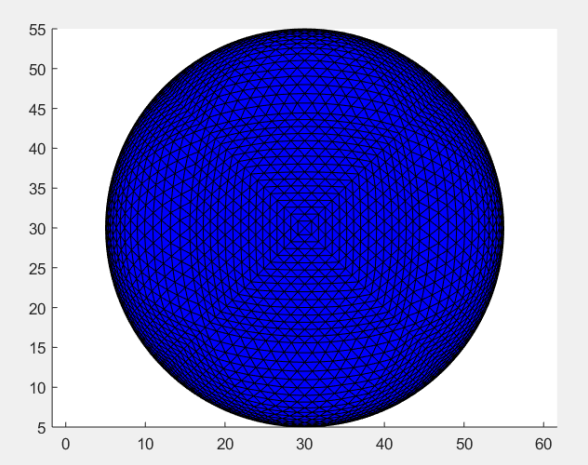


Figure 23 : Sphère initiale

On procède à différent tests sur des formes simples en modifiant ces paramètres. En faisant quelques essais, il a été possible de déterminer des jeux de paramètres fonctionnant correctement pour détourer un cube et un ovale noir.

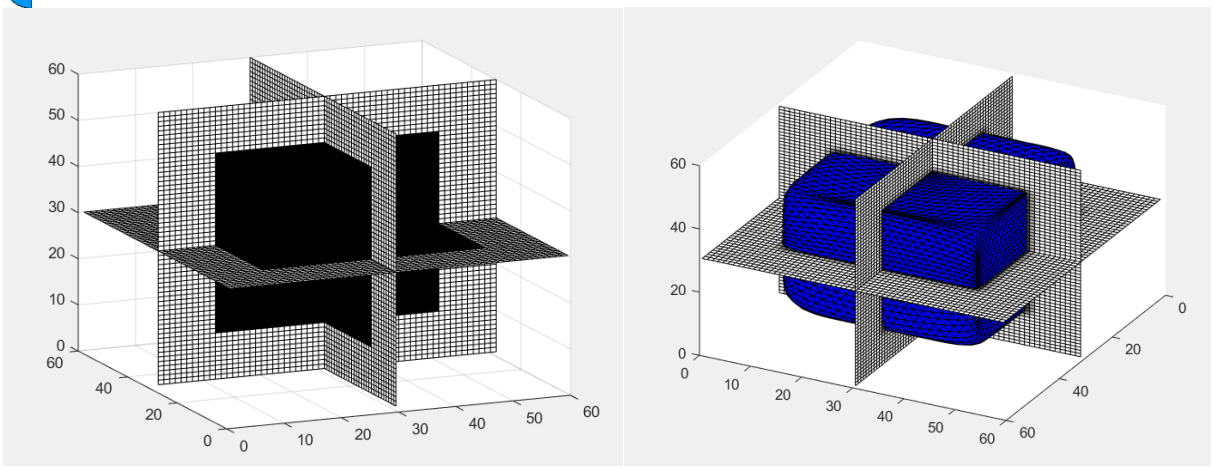


Figure 24 : Cube noir avant et après détourage

Paramètres utilisés :

alpha=150 ;
beta=500 ;
gamma=0.0001;
nmax=300; %nombre d'itérations
kb=75; %coefficient de la force ballon
sigma=1; %flou gaussien

Utilisons maintenant un ovale noir comme image. En utilisant sur cet ovale le jeu de paramètres qui fonctionne avec le cube, la sphère converge mal par endroit, il faut donc adapter les paramètres à la forme de l'image.

alpha=800 ;
beta=500 ;
gamma=0.0002;
nmax=300; %nombre d'itérations
kb=75; %coefficient de la force ballon
sigma=1; %flou gaussien

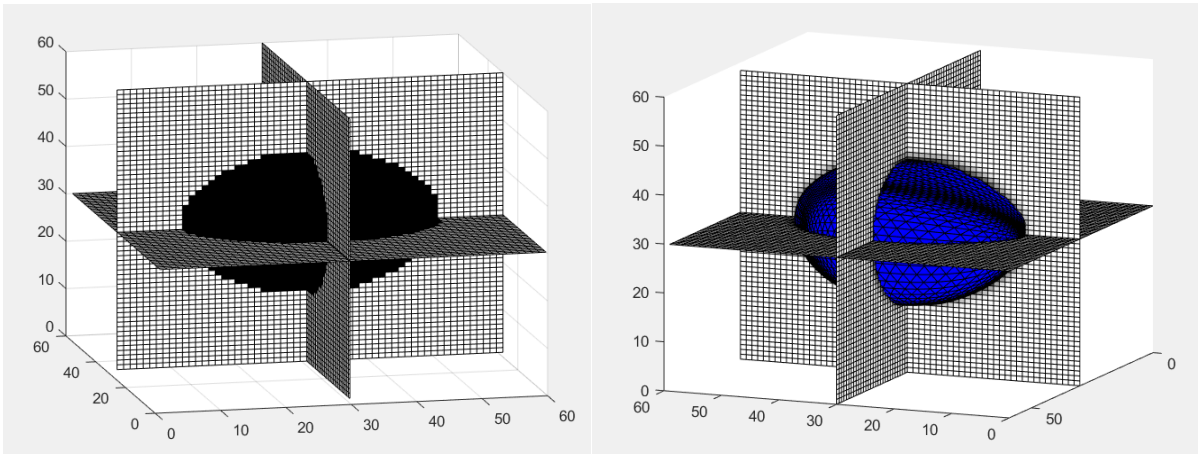


Figure 25 : Ovale noir avant et après détourage

3.3. Tests 3D sur des poumons

Les images 3D sont codées à l'aide de matrices. Pour tester le contour 3D sur des images médicales 3D, nous avons une matrice nommée « Lungs » qui contient les données d'une image de poumons en 3D.

Il faut effectuer quelques réglages sur cette matrice pour pouvoir s'en servir.

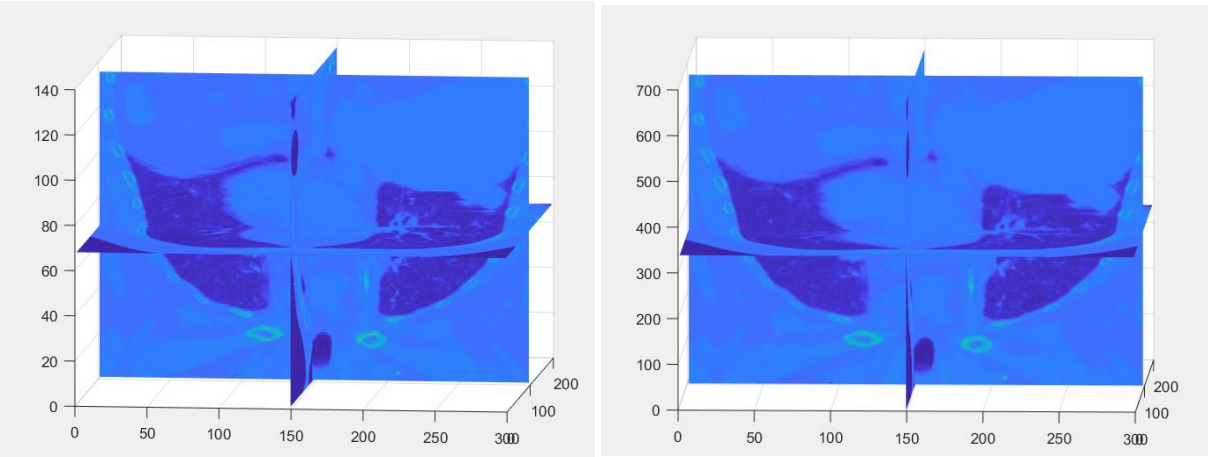
Tout d'abord le programme a été codé pour travailler avec des images dont les pixels sont codés sur 8 bits (de 0 à 255). Un bit à 0 est noir et un bit à 255 est blanc. Les valeurs intermédiaires représentent les différents niveaux de gris. Or la matrice Lungs n'est pas codée sur 8 bits mais sur 12.

On peut repasser les valeurs sur 8 bits en divisant les valeurs des pixels par 16.

```
img8 = uint8(img12 / 16);
```

On peut alors afficher l'image avec la fonction slice. Une interpolation a également été faite sur l'axe Z. En effet pour x et y, la matrice dispose d'un point par millimètre. Cependant pour l'axe z, il n'y a qu'un point tous les 5 mm.

On peut voir ci-dessous, l'image avant et après l'interpolation, on peut voir le changement sur l'axe Z.



Pour les premiers tests, l'image 3D sera binarisée afin d'avoir un bon contraste, lorsque l'on obtiendra un bon contour sur l'image binarisée on pourra rajouter les niveaux de gris.

3.4. Subdivision adaptative en 3D

On souhaite mettre en place une subdivision adaptative. La sphère qui sert de contour actif est découpée en triangles. Le but est de pouvoir créer de nouveaux triangles pour gagner en précision.

Ici notre sphère de base est petite et doit s'agrandir pour rejoindre les bords d'un cube. Le but est donc de calculer l'aire de chaque triangle et de subdiviser si le triangle est trop grand.

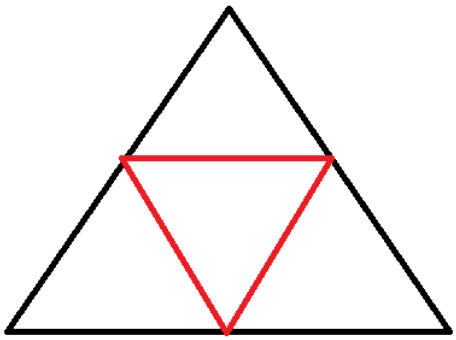


Figure 26 : Subdivision d'un triangle

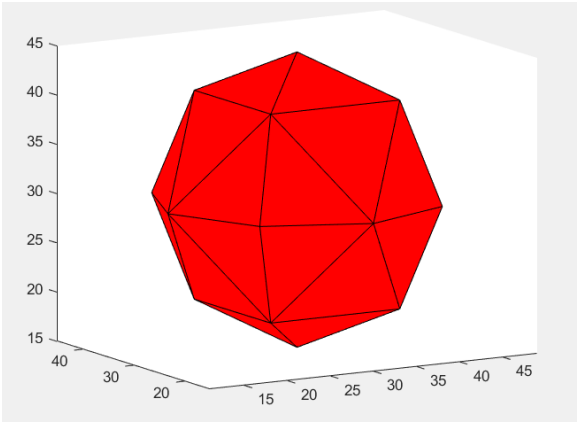


Figure 27 : Sphère de départ

Ci-dessus, le triangle noir a été subdivisé en ajoutant un triangle à l'intérieur de sorte que les sommets du nouveau triangle soient au milieu des côtés du triangle de base. En appliquant cela à l'algorithme sur une « sphère » assez basique, on observe ci-dessous que la subdivision se déroule bien mais qu'au fil des itérations, les nouveaux triangles se détachent du reste de la sphère.

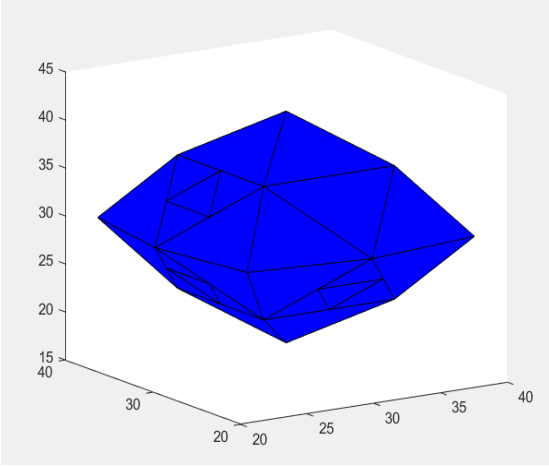


Figure 28 : Juste après une subdivision

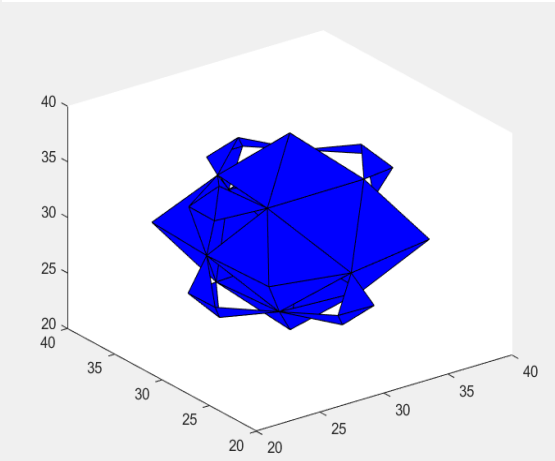


Figure 29 : Après une itération supplémentaire

A chaque itération, des forces s'appliquent sur la sphère pour la faire converger ou diverger, or comme les sommets des nouveaux triangles ne sont pas communs à des sommets voisins, ils se détachent de la sphère principale. Il faut trouver une solution pour éviter ce phénomène. Il faudrait éviter que les nouveaux sommets se retrouvent au milieu d'une arête.

Pour cela, on peut changer la façon de subdiviser :

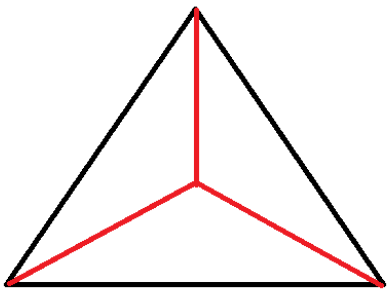


Figure 30 : Autre forme de subdivision

Ci-dessous, on a subdivisé au niveau du centre de gravité du triangle. Les sommets sont en contact uniquement avec d'autres sommets. Un triangle est divisé en 3.

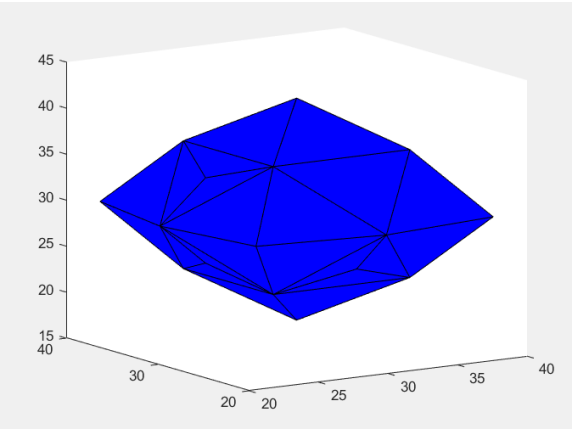


Figure 31 : Juste après une subdivision

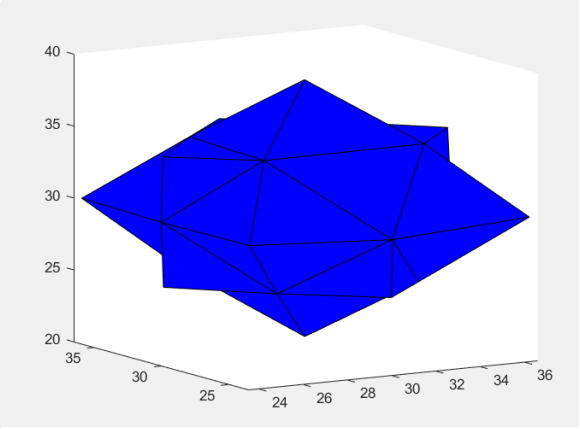


Figure 32 : Après une itération supplémentaire

La subdivision se déroule bien avec ce procédé, seulement au fil des subdivisions, les triangles de base sont toujours aussi grands et les nouveaux triangles sont de plus en plus allongés.

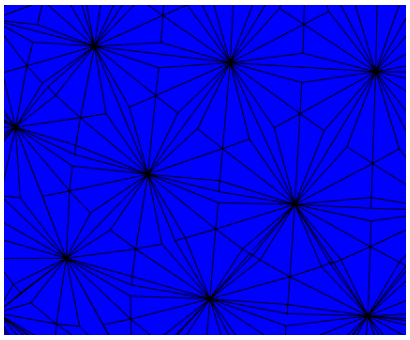
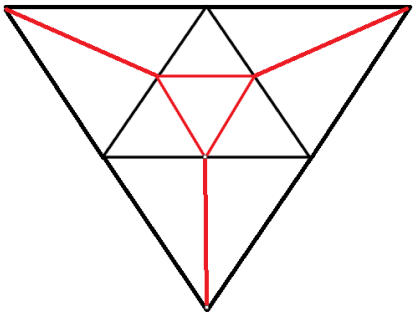


Figure 33 : Après plusieurs subdivisions

Avec l'autre forme de subdivision, les nouveaux triangles n'étaient pas allongés même après plusieurs subdivisions. On souhaite donc reprendre cette méthode. Pour l'utiliser et éviter le phénomène de décollement d'autres solutions existent :

- Faire la subdivision sur tous les triangles de la sphère en même temps, ainsi les nouveaux sommets ne seront pas au milieu d'une arête mais en contact avec d'autres sommets.
- Subdiviser en deux les 3 triangles voisins.



Nous avons essayé de faire la subdivision sur tous les triangles.

Il faut prendre le temps de bien régler les paramètres pour que cette subdivision donne un bon contour final, notamment la variable airemax qui correspond à l'aire à partir de laquelle il est nécessaire de subdiviser.

```
alpha=150;
beta=50;
gamma=0.0001;
nmax=300;
kb=-1000;
sigma=1;
lambda=50;
gradlim=1e3;
kg=0.01*30;
airemax=0.5;
```

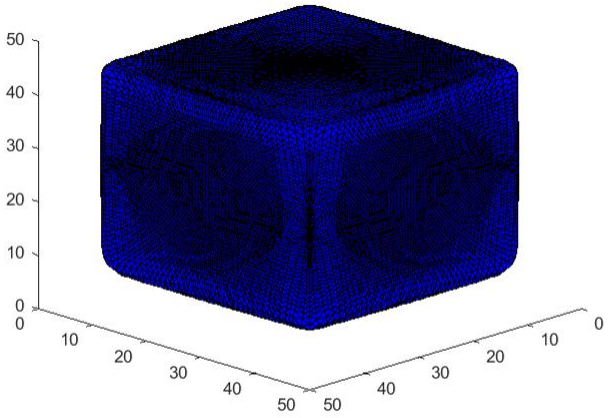


Figure 34 : Contour actif sur le Cube avec la subdivision adaptative

On peut ensuite essayer de trouver un jeu de paramètres fonctionnant correctement avec les poumons. Après plusieurs essais, la résolution des poumons semble poser un problème. En effet le cube était codé sur une matrice 60x60x60 et les poumons sont codés sur une matrice 201x299x136. La différence de résolution est donc importante. Il est difficile de trouver un bon jeu de paramètres, d'autant plus que les temps d'exécution du programme sont bien plus longs avec les poumons.

On modifie la résolution des poumons pour avoir la même que celle du cube. Pour cela on utilise les fonctions linspace, meshgrid et interp3 de Matlab.

Cela nous permet d'avoir de bons résultats avec les poumons. On a réutilisé les mêmes valeurs de paramètres utilisés pour le cube. Voici le résultat ci-dessous, l'un des deux poumons a plutôt bien été détouré.

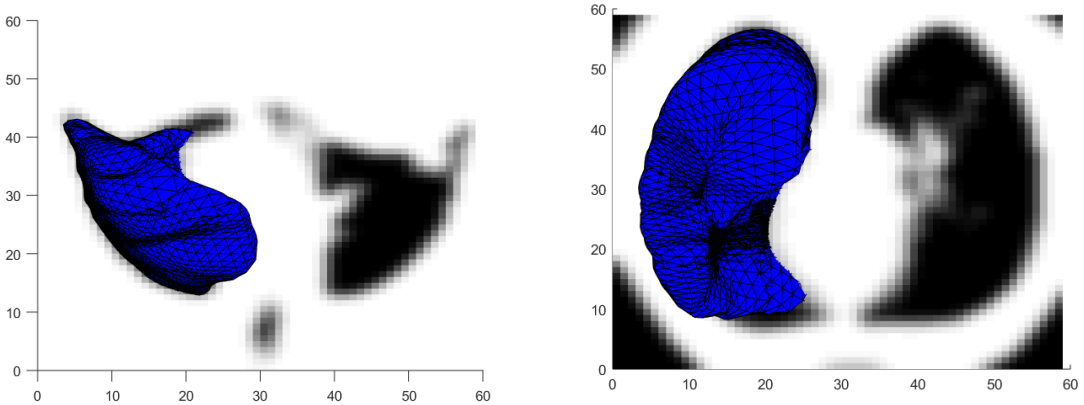


Figure 35 : Résultats obtenus pour les poumons sur deux plans différents

3.5. Tests sur Alpha

Nous avons pu déterminer un jeu de paramètres fonctionnant correctement sur le cube. Nous partons d'une petite sphère initiale qui s'agrandit au fil des itérations pour détourer les limites du cube.

```
alpha=150;
beta=50;
gamma=0.0001;
nmax=300;
kb=-1000;
sigma=1;
lambda=50;
gradlim=1e3;
kg=0.01*30;
airemax=0.5;
```

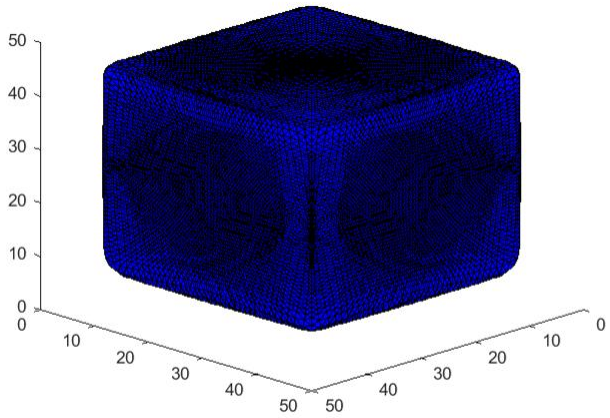


Figure 36 : Résultat pour alpha=150

Nous avons conservé ces paramètres et changé uniquement alpha ou voir son influence. Alpha agit sur la convergence de la sphère, comme on souhaite que notre sphère s'agrandisse, alpha ne doit donc pas être trop grand. La force Ballon pousse la sphère à s'agrandir et contre cette force de convergence.

Pour un alpha plus petit (mais toujours positif), le résultat reste similaire à l'image ci-dessus. Le contour épouse bien la forme du cube.

En revanche pour un alpha=1000, cela augmente les forces de convergence de la sphère et celle-ci n'arrive plus à remplir les coins du cube.

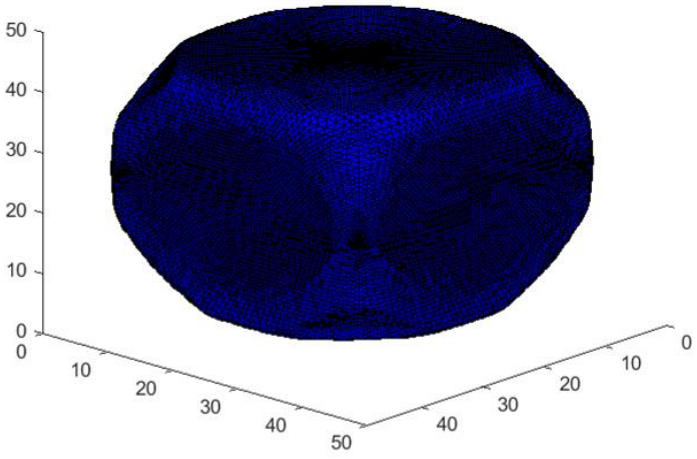


Figure 37 : Résultat pour alpha=1000

Pour un alpha encore plus grand, les forces de convergence deviennent plus fortes que la force Ballon et la sphère se rétrécit et converge vers un point.

3.6. Tests sur Beta

Beta lisse la surface de la sphère, pour pouvoir le mettre en évidence, les paramètres suivants ont été utilisés :

alpha=0.1 ;
beta=0.00001 ou 1000;
gamma=0.0001;
nmax=200 ;
kb=-1000 ;
sigma=1 ;
airemax=0.5;

Les images ci-dessous sont un zoom d'une partie du cube.

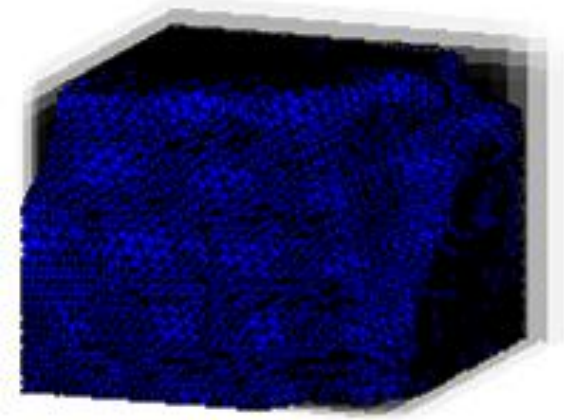


Figure 38 : Beta=0.00001

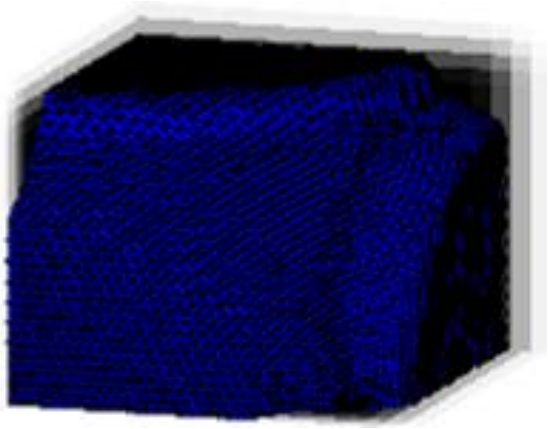


Figure 39 : Beta=1000

On peut voir que pour un Beta faible, la surface est plus irrégulière et bosselée, elle devient plus lisse quand Beta s'agrandit. Beta a donc bien une influence sur le lissage de la surface.

Conclusion

Ce stage a été sous plusieurs aspects riche d'enseignement. Le but était d'améliorer et tester des algorithmes déjà existants pour effectuer un contour actif.

Après avoir étudié l'influence des différents paramètres en 2D, nous avons pu améliorer la précision des contours en mettant en place une subdivision adaptative. De plus, en moyennant les forces de ballon, on a diminué l'apparition d'oreilles de Mickey. On a ensuite pu expérimenter tout cela sur différentes images dont des images médicales.

La subdivision du contour a également permis d'améliorer les résultats sur des images 3D et il a été intéressant de voir l'impact de la forme de subdivision sur le résultat final.

Cela m'a permis d'améliorer mes connaissances, notamment sur l'utilisation du logiciel Matlab, le traitement d'image en général et sur la façon dont est calculé un contour actif. De plus, ce stage en laboratoire m'a permis de découvrir un peu plus le monde de la recherche en Informatique et ses différents domaines d'applications.

Références

[1] Pierre-Frédéric Villard, Franck P. Vidal, Llyr Ap Cenydd, Richard Holbrey, S. Pisharody, Sheena Johnson, Andy Bulpitt, Nigel W. John, Fernando Bello, and Derek A. Gould. Interventional radiology virtual simulator for liver biopsy. International Journal of Computer Assisted Radiology and Surgery, pages 1{13, 2013.

[2] Erwan Kerrien, Ahmed Yureidini, Jeremie Dequidt, Christian Duriez, René Anxionnat, and Stéphane Cotin. Blood vessel modeling for interactive simulation of interventional neuroradiology procedures. Medical image analysis, 35:685{698, 2017.

[3] Andre Mastmeyer, Matthias Wilms, and Heinz Handels. Population-based respiratory 4d motion atlas construction and its application for VR simulations of liver punctures. 12 2017.

[4] Christophe Maggia, Thomas Mistral, Senan Doyle, Florence Forbes, Alexandre Krainik, Damien Galanaud, Emmanuelle Schmitt, Stéphane Kremer, Irène Troprès, Emmanuel L. Barbier, Jean-François Payen, and Michel Dojat. Traumatic Brain Lesion Quantification based on Mean Diffusivity Changes. In Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries, BrainLes (MICCAI), ed Crimi Aea (Springer International Publishing AG). 2018.

[5] R. Trullo, C. Petitjean, S. Ruan, B. Dubray, D. Nie, and D. Shen. Segmentation of organs at risk in thoracic ct images using a sharpmask architecture and conditional random fields. In 2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017), pages 1003{1006, April 2017.

[6] Mehmet Sezgin and Bülent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic imaging, 13(1):146{166, 2004.

[7] Nobuyuki Otsu. A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics, 9(1):62{66, 1979.

[8] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. International Journal of Computer Vision, 1(4):321{331, 1988.

[9] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Constraints on deformable models: Recovering 3d shape and nonrigid motion. Artificial intelligence, 36(1):91{123, 1988.

[10] William H Press, Saul Arno Teukolsky, William T Vetterling, and Brian P Flannery. Numerical recipes in c++: the art of scientific computing. 2002.