Segmentation d'images médicales

Olivier Boissard, Antoine Lavier

27 janvier 2016

Table des matières

Chapitre 1

Introduction

L'objectif de ce projet est de développer un logiciel de segmentation d'images médicales, afin de détecter les différents éléments présents sur un scanner. Les images à analyser sont au format DICOM, conçu dans le but de stocker à la fois l'image et les données du patient dans le même fichier, afin d'assurer la traçabilité du patient. Cette segmentation doit permettre de trouver tous les éléments apparaissant sur l'image pour aider au traitement du cancer du rein chez l'enfant. Or il n'existe pas de logiciel capable d'effectuer une segmentation sans intervention humaine, c'est à dire de logiciel capable de détecter et d'afficher une tumeur à partir d'un cliché sans avoir besoin d'un utilisateur. L'objectif de ce projet est donc de trouver une solution pour effectuer cette segmentation de façon automatique.

Chapitre 2

Contexte

Ce projet s'inscrit dans le cadre d'un projet de plus grande envergure : le projet SAIAD (Segmentation Automatique de reins tumoraux chez l'enfant par Intelligence Artificielle Distribuée). Il s'agit d'un projet mené conjointement par l'Université de Franche-Comté, le Centre Hospitalier Régional Universitaire, Covalia, et l'Ecole Polytechnique Fédérale de Lausanne.
Les tumeurs du rein sont les tumeurs abdominales les plus fréquentes chez l'enfant, et sont diagnostiquées entre 30 et 47 mois. Actuellement, ce diagnostic se base sur l'échographie pour détecter la tumeur et le scanner pour la localiser précisément.

Or il faut analyser en moyenne 250 images par patient. Le projet SAIAD a donc pour objectifs de :

— construire une base de connaissances permettant d'automatiser la segmentation,
— utiliser l'intelligence artificielle pour optimiser la segmentation,
— permettre la représentation en 3 dimensions des viscères segmentés.

Cette représentation en 3 dimensions pour alors servir de base pour les diagnostics pour les simplifier par rapport à une série de scanner. Or la transformation des scanners en images en 3 dimensions de façon automatique est actuellement impossible car il n'existe pas de logiciel capable de segmenter une image sans intervention humaine. La segmentation des images devra donc permettre de détecter tous éléments présents dans l'abdomen sur des coupes à des niveaux différents sans interaction avec l'utilisateur.

Chapitre 3

Etat de l'art de la segmentation

La segmentation d'image est une opération qui consiste à rassembler des pixels entre eux selon des critères prédéfinis. Ces groupes de pixels forment des régions.
La segmentation idéale n'existe pas. Une bonne technique de segmentation permet de simplifier une image, sans perdre de contenu. Les algorithmes les plus efficaces dépendent donc du type d'image à analyser.
La segmentation d'une image I correspond à sa partition en une ou plusieurs région R_i telles que :

- $\bigcup_{i=1}^{n} R_i = I$
- $\forall i, j.\ i \neq j \Rightarrow R_i \cap R_j = \emptyset$

Il existe plusieurs types de segmentation : la segmentation par seuillage, et la segmentation par approche région.

3.1 Segmentation par seuillage

La segmentation par seuillage permet de partitionner une image uniquement à partir de son histogramme, c'est à dire la représentation du nombre de pixels de l'image en fontion de leur niveau de gris. Le niveau de gris d'un pixel est une valeur comprise entre 0 et 255. 0 correspond à du noir, et 255 à du blanc.
La segmentation par seuillage peut être effectuée de différentes façons. Les plus courantes sont :

- la détection de vallée,
- la minimisation de variance : permet de regrouper les pixels en N classes. Le choix des seuils permet de trouver le nombre optimal de classes,

— le seuillage entropique : non applicable à ce projet car l'image résultante parait de moins bonne qualité que l'image originale,
— la méthode du pourcentage,
— la maximisation du contraste : cette méthode n'utilise pas l'histogramme mais la répartition des niveaux de gris sur l'image. Le but est de maximiser le contraste dans l'image résultante.

Ces techniques de segmentation sont simples à mettre en place, et s'exécutent rapidement. Cependant, elles sont inutilisables si la complexité de l'image est trop importante pour être résumée dans son histogramme.

3.2 Segmentation par approche région

Dans le cas d'une approche région, l'image est partitionnée en fonction de régions homogènes (constituées de pixels similaires). Il existe différentes techniques :

— la croissance de régions (ascendante, bottom-up),
— la division-fusion (top-down)
— la fusion de régions

Tous ces algorithmes utilisent un critère d'homogénéité. Dans le cas d'un scanner en noir et blanc, il s'agit du niveau de gris des pixels. En effet, tous les tissus différents ont un niveau de gris différent.
Dans le cas de la croissance de région, on commence par choisir des pixels germes. L'algorithme analyse les pixels voisins aux germes, et le respect du critère d'homogénéité. Tant que les pixels connexes respectent ce critère, on les ajoute à la même région. Cette technique est simple à implémenter, mais elle est sensible aux anomalies des détails de l'image.
Pour l'algorithme de division-fusion, on vérifie le respect du critère d'homogénéité par l'image entière. Tant qu'il n'est pas respecté en recommançant sur chaque région obtenue, l'image est divisée. Les divisions successives de l'image peuvent conduire à une sur-segmentation. En effet, l'image est divisée en généralement en 4 sous régions, mais il est possible que seule une partie d'entre elles (par exemple 1 ou 2 régions) ne respectent pas le critère d'homogénéité. Il faut alors les fusionner. Cet algorithme est moins sensible au bruit que la croissance de régions, mais est plus complexe à implémenter. De plus, selon l'image d'origine et l'algorithme de fusion, les régions ne seront pas fidèles à l'image.
Enfin, la fusion de région est une approche ascendante après une sur-segmentation dans le cas d'une technique de division-fusion, ou sur l'image de départ. On analyse les pixels successivement, et on teste si les carrés de 2 * 2 pixels respectent le critère d'homogénéité. Si c'est le cas, on les fusionne en une région. Une fois toute l'image analysée, on recommence avec des carrés de 2

*2 régions, jusqu'à ce qu'il ne soit plus possible de fusionner des régions.

3.3 Application de la segmentation aux scanners du rein

3.3.1 Anatomie de l'abdomen et du rein

L'abdomen est la partie du corps située entre le thorax et le petit bassin. Il contient la plupart des organes digestifs. De plus, on y trouve également les reins, qui sont les organes cibles de ce projet. L'abdomen est composé de la cavité abdominale en avant, et du rétropéritoine en arrière. Les reins se situent dans ce rétropéritoine.
Le rein possède des fonctions hormonales, régule la tension et permet l'élimination des toxines avec le foie et les poumons. Il s'agit d'un organe vital. Les deux reins ne sont pas tout à fait à la même hauteur (le rein gauche étant plus haut). Ceci implique qu'une seule coupe transversale ne permet pas de voir la même région de chaque rein.
Une tumeur peut se développer sur les reins. Le traitement est alors le plus souvent chirurgical. En effet, la chimiothérapie et la radiothérapie ne sont pas actifs sur ce type de cancer. Toutefois, quand il y a des métastases, l'immunothérapie peut permettre de stimuler les défenses immunitaires pour détruire les cellules cancéreuses. Si ces thérapies sont inefficaces, on peut procéder à une tumorectomie, c'est à dire l'ablation de la tumeur en préservant le rein.
Un seul rein suffit pour vivre. C'est le cas de 5% des individus. Ainsi, le chrirurgien peut également procéder à une néphrectomie, c'est à dire l'ablation du rein entier, ainsi que de la tumeur.

3.3.2 Détection des tumeurs

Dans certains cas, le cancer du rein est découvert grâce à un dépistage précoce chez les personnes à risques (personnes possédant certains troubles génétiques héréditaires ou en dialyse rénale prolongée).
Mais la plupart du temps, il est découvert par hasard lors d'une échographie, ou d'un scanner abdominal. Dans tout les cas, il est nécessaire de pouvoir le localiser précisément. Pour cela, l'application développée au cours de ce projet utilisera plusieurs algorithmes de segmentation existants. En effet, un algorithme de minimisation de variance permettra de simplifier l'image, puis une segmentation par la technique de division-fusion servira à repérer les différentes régions apparaissant sur le scanner. Enfin une croissance de région pourra repérer précisement les contours de ces régions.

Chapitre 4

Architecture du projet

Lors de ce projet, la première étape du développement a été de de convertir une image au format DICOM en une image au format JPEG, plus simple à manipuler.
Ensuite, il a fallu mettre en place une segmentation par minimisation de variance. Cet algorithme permet de simplifier l'image en réduisant le nombre de niveaux de gris différents. Ceci permet d'avoir une image moins précise de l'intérieur de chaque région, mais d'augmenter la précision du contour des viscères représentés sur le scanner.
Après avoir simplifié l'image, le logiciel peut exécuter un algorithme de division-fusion, afin de détecter les différentes régions représentant les viscères. Cet algorithme permet de définir la position de pixels germes, qui permettront ensuite d'appliquer un algorithme de croissance de région.
Ce dernier algorithme permet d'agglomérer des pixels à l'intérieur d'une région, pour repérer précisément les limites des organes.
Ces différents calculs sont possibles grâce à l'utilisation de plusieurs outils permettant la compilation du code source et la manipulation d'images.
Enfin, une interface graphique permet l'utilisation de l'application de façon simple, avec la possibilité de paramétrer la précision de chacune des méthodes de segmentations appliquées en attendant que l'application soit intégrée au projet SAIAD.

4.1 Outils utilisés

En parallèle aux recherches sur les différents algorithmes de segmentation, nous avons dû également faire des choix en matière de technologies à utiliser. Notre choix s'est tout de suite porté sur le langage de programmation C++ pour mener à bien ce projet. En effet, le code étant compilé, l'exécution du programme est plus rapide (d'autant plus que l'application de formules mathématiques sur des images représente des calculs assez longs).

4.1.1 CMake

CMake est un système de construction logiciel (build systems, en anglais). Comme le nom l'indique, un tel système a pour but de permettre, directement ou indirectement, la compilation d'un code source.
Son fonctionnement est le suivant : dans un premier temps, le développeur écrit dans un fichier la description du projet (incluant la liste des fichiers sources, les bibliothèques à lier au binaire final, etc...) de façon totalement indépendant de la configuration logicielle de la machine. Ce fichier s'appelle *"CMakeList.txt"*. Ensuite, les personnes souhaitant compiler le projet appellent le système de construction (CMake) en lui passant en paramètre le fichier *"CMakeList.txt"* écrit par le développeur. Le système analyse alors le fichier d'une part et tente de récupérer des informations sur la configuration logicielle de la machine l'exécutant d'autre part. Une fois ces opérations accomplies, il sera enfin en mesure de produire un script de compilation de bas niveau (ici, un fichier *"Makefile"*).
Nous avons donc choisi d'utiliser cet outil pour permettre à notre code d'être réutilisé par d'autres développeurs de la manière la plus simple possible en matière de configuration de la machine.

4.1.2 Imebra

Imebra Dicom SDK est une bibliothèque C++ Open Source permettant de manipuler des fichiers DICOM, qu'ils soient bruts ou compressés. Ses fonctions principales sont les suivantes :

- Parser et construire des fichiers DICOM,
- Compression et décompression d'images dans plusieurs formats (jpeg 8bits, jpeg 12bits, jpeg16bits, raw dicom, rle dicom),
- Support transparent de l'Unicode (utilisation avec des fichiers internationaux),
- etc...

Nous utilisons Imebra simplement pour extraire l'image contenue dans un fichier DICOM et l'enregistrer sous format JPEG pour pouvoir ensuite travailler dessus. En effet, nous n'avons pas jugé utile de passer beaucoup de temps sur la manipulation des fichiers DICOM au détriment du résultat de nos algorithmes. Nous utilisons juste un programme fourni en tant qu'exemple dans la librairie DICOM permettant de convertir un fichier DICOM en image JPEG. Bien sûr, à terme cette libraire est intéressante pour afficher dans le logiciel, en plus de l'image, les informations concernant le patient, l'historique des manipulations, etc...

4.1.3 OpenCV

OpenCV est une bibliothèque graphique libre, initialement développée par Intel et spécialisée dans le traitement d'images (fixes ou en temps réel).

Celle-ci met à disposition un grand nombre de fonctionnalités très diversifiées permettant de créer des programmes partant de données brutes pour aller jusqu'à la création d'interface graphiques basiques. Parmi ces fonctionnalités, on peut trouver lecture/écriture et affichage d'une image, le calcul de l'histogramme des niveaux de gris ou couleurs, lissage, filtrage, morphologies mathématiques,...
Nous avons donc choisi d'utiliser cet outil pour manipuler les images et y appliquer nos algorithmes.

4.1.4 Qt

Qt est une bibliothèque multiplateforme permettant de créer des GUI (des interfaces graphiques). En fait, il s'agit plus d'un framework mettant à notre disposition un ensemble d'outils pour développer. Cela dit, nous utilisons Qt uniquement pour la création d'une interface graphique basique simplifiant l'utilisation de nos différents algorithmes et l'affichage des résultats.

4.2 Segmentation par minimisation de variance

La segmentation par minimisation de variance est une tâche qui consiste à regrouper un ensemble de données (ici des pixels) de tels manières que les données d'un même groupe (appelé cluster) sont plus proches (au sens d'un critère de similarité choisi) les unes des autres que de celles des autres clusters. Cette technique s'applique dans de nombreux domaines : apprentissage automatique, reconnaissance de formes, traitement du signal, recherche d'informations, et bien sûr, traitement d'image. L'idée est donc de créer ces clusters de façon automatique. Pour celà, on utilise une heuristique de partitionnement connu sous le nom de méthode des k-moyennes.
Nous avons donc utilisé l'algorithme K-means pour la classification automatique des pixels $(x_1, ..., x_n)$. K-means minimise le critère d'erreur (distorsion) suivant par rapport aux centres des classes $\Psi = (\mu_1, ...\mu_K)$ et les classes $z = (z_1, ..., z_n) : (z, \Psi)$ grâce à l'équation :

$$\jmath(\mu_1, ..., \mu_K, z) = \sum_{k=1}^{K} \sum_{i=1}^{n} ||x_i - \mu_k||^2 \tag{4.1}$$

qui correspond à la distance euclidienne totale entre chaque données x_i et le centre μ_{zi} dont elle est la plus proche au sens de la distance Euclidienne :

$$||x_i - \mu_k||^2 = d(x_i, \mu_k) = \sqrt{\sum_{j=1}^{d} (x_{ij} - \mu_{kj})^2} \tag{4.2}$$

Dans l'expression du critère $\jmath$, z_{ik} est une variable binaire qui vaut 1 si la classe du $i$ème exemple x_i est k et 0 sinon.

4.2.1 Méthode des k-moyennes

L'algorithme K-means est composé des trois étapes suivantes :

— **Initialisation** : On initialise les centres des classes $(\mu_1^{(0)}, ..., \mu_K^{(0)})$ pour donner le pas de départ de l'algorithme (en choisissant des valeurs de gris au hasard en fonction du nombre de clusters par exemple). Il s'agit donc de démarrer à l'itérationn $t = 0$ avec des valeurs initiales pour les paramètres du modèle $(\mu_1^{(0)}, ..., \mu_K^{(0)})$,

— **Etape d'affectation (classification)** : Chaque donnée est assignée à la classe du centre dont elle est la plus proche : $\forall i = 1, ..., n$

$$z_{ik}^{(t)} = \begin{cases} 1 \text{ si } k = arg \min_{z \in \{1,...,k\}} ||x_i - \mu_z||^2 \\ 0 \text{ sinon} \end{cases} \tag{4.3}$$

— **Etape de recalage des centres** : le centre μ de chaque classe k est recalculé comme étant la moyenne arithmétique de toutes les données appartenant à cette classe (sutie à l'étape d'affectation précédente) : $\forall k = 1, ..., K$

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{n} z_{ik}^{(t)} x_i}{\sum_{i=1}^{n} z_{ik}^{(t)}}, \tag{4.4}$$

t étant l'itération courante.

La convergence peut être considérée comme atteinte si la valeur relative au niveau de la distorsion $\jmath(1)$ devient inférieure à un seuil petit préfixé ou si un nombre maximum d'itérations préfixé a été atteint. En attendant, on peut répéter ces deux dernière étapes.

Un fois que l'algorithme a convergé, on a donc une estimation des classes $(z_1, ..., z_n)$ et des centres $(\mu_1, ..., \mu_K)$. On connait donc l'appartenance à z_i (la classe) de chaque donnée x_i à partir de z_{ik}.

4.2.2 Algorithme

Input : Données $(x_1, .., x_n)$, nombre de clusters K
Itération : $t \leftarrow 0$
/* Initialisation : */
Initialiser les centres des classes : $\Psi^{(0)} = (\mu_1^{(0)}, ..., \mu_K^{(0)})$
/* */
Nombre d'itérations max : $maxIter \leftarrow 1000$
Fixer un seuil petit pour le test de convergence : $\epsilon > 0$; e.g., $\epsilon = 10^{-6}$
Distorsion : $\jmath^{(t)} \leftarrow +\infty$

/* K-means */

while $((converge \neq 0)$ **and** $(t \leq maxIter))$ **do**

/* Etape affectation */

for $i \leftarrow 1$ **to** n **do**
for $k \leftarrow 1$ **to** K **do**
Calculer $z_{ik}^{(t)}$ en utilisant l'équation (4.3)
end for
end for

/* Etape recalage (recalcul des centres) */

for $k \leftarrow 1$ **to** K **do**
Calculer $\mu_k^{(t+1)}$ en utilisant l'équation (4.4)
for $j \leftarrow 1$ **to** d **do**
Calculer $\mu_{kj}^{(t+1)}$
end for
end for

$t \leftarrow t + 1$ (itération suivante)

Calculer la distorsion $\jmath^{(q+1)}$ en utilisant l'équation (4.1)
if $((\frac{|\jmath^{(t+1)} - \jmath^{(t)}|}{|\jmath^{(t)}|} \leq \epsilon) || (t \geq maxIter))$ **then**
$converge \leftarrow 1$
end if
end while

Result : les classes $(z_1, ..., z_n)$ et les centres des classes $(\mu_1, .., \mu_K)$

4.3 Segmentation par division-fusion

L'algorithme de segmentation par division-fusion (algorithme Split and Merge) permet la détection de régions peu précises d'une image. Pour cela, on commence par analyser l'image, et la comparer à un critère d'homogénéité. Dans le cas d'un scanner en noir et blanc, ce critère correspond au niveau de gris des régions. En effet, une région sera homogène lorsque tous les pixels qu'elle contient auront le même niveau de gris. Ensuite, on divise l'image en sous-régions jusqu'à ce que toutes les régions respèctent le critère d'homogénéité.
Cette division ne peut produire que des régions rectangulaires, et surtout plus petites que les régions qui existent réellement sur l'image d'origine. On obtient donc un nombre de régions beaucoup plus grand que ce que l'on recherche. Cet effet s'appelle la sur-segmentation. Il faut donc ensuite appliquer un algorithme de fusion, pour réduire la quantité de régions en fusionnant les régions voisines qui respectent toutes le même critère d'homogénéité.

4.3.1 Division

L'algorithme de division est relativement simple à mettre en place. Si une région ne respecte pas le critère d'homogénéité, on la divise en plusieurs sous-régions. Généralement, on utilise 4, 6, 8... sous-régions. Dans ce projet, chaque région est divisée en 4. Cet algorithme utilise un graphe pour les sous-régions et un critère d'homogénéité. Le graphe peut être un quad-tree, chaque noeud correpondant à une région, et chaque fils correspondant à une sous-région. La seconde solution est d'utiliser un graphe d'adjacence des régions. Ceci permettant une gestion plus simple des régions voisines pour l'algorithme de fusion, c'est la solution qui sera implémentée.
Pour le déroulement de l'algorithme, on considère que l'image d'origine correspond à une région. Ensuite, on utilise une fonction récursive. Si la région d'origine ne respecte pas le critère d'homogénéité, on créé 4 sous-régions, et on applique l'algorithme à chaque sous-région, jusqu'à ce que toutes les régions respectent le critère d'homogénéité.

AJOUTER IMAGE EXEMPLE AVEC SUR SEGMENTATION

Afin de simplifier l'algorithme, la gestion des voisins de chaque région se fait après la division de l'ensemble de l'image.
Après cette étape, on obtient une sur-segmentation. En effet, une région est divisée en 4 même si seule une petite partie de la région ne respecte pas le critère d'homogénéité. Il faut donc compenser en fusionnant les régions voisines quand c'est possible.

4.3.2 Fusion

L'algorithme de fusion s'effectue juste après l'algorithme de division, afin de compenser la sur-segmentation.
Pour cela, on parcourt le graphe, et on compare chaque région à ses voisines. Si la région en cours d'analyse respecte le même critère d'homogénéité qu'une des régions voisines, on créé une région plus grande.

AJOUTER IMAGE EXEMPLE

Cet algorithme de fusion peut créer uniquement des régions carrées ou rectangulaires. Bien que ceci puisse donner une idée de la position des tumeurs sur un scanner, ces régions restent inutilisables par les chirurgiens. Ainsi, cet algorithme ne sert qu'à calculer les coordonnées des points de départ de l'algorithme de croissance de région. En effet, ces points correspondent au centre des régions générées par la technique de division-fusion.

4.3.3 Algorithme

L'algorithme de division fonctionne de façon récursive. Si la région passée en paramètre n'est pas homogène, on la divise, et on rapelle la fonction sur chaque sous-région.

Algorithm 1 split(Region region)

créer la liste des sous-régions potentielles $sr1$, $sr2$, $sr3$, $sr4$
if $isHomogeneous(sr1, sr2, sr3, sr4)$ **then**
  retirer la région du graphe
  ajouter les 4 sous-régions dans le graphe
  $split(sr1)$
  $split(sr2)$
  $split(sr3)$
  $split(sr4)$
end if

L'algorithme de fusion en revanche parcourt le graphe de façon itérative, et fusionne les régions homogènes.

Algorithm 2 merge()

```
for all region from graph do
  for all neighbor from region.getNeighbors() do
    toMerge
    if region.getGrey() - neighbor.getGrey() < DIFGREYMERGE
    then
      if region.getCoordiante().getX() ==
      neighbor.getCoordiante().getX() then
        if region.getWidth() == neighbor.getWidth() then
          toMerge = true
          Calculer les coordonnées et la taille de la nouvelle région
        end if
      else if region.getCoordiante().gety() ==
      neighbor.getCoordiante().gety() then
        if region.getHeight() == neighbor.getHeight() then
          toMerge = true
          Calculer les coordonnées et la taille de la nouvelle région
        end if
      end if
      if toMerge then
        retirer les 2 régions du graphe
        ajouter la nouvelle région
        corriger les listes de voisins du graphe
      end if
    end if
  end for
end for
```

4.4 Segmentation par croissance de région

L'algorithme de croissance de région utilise des points de départ, appelés pixels germes. Ces points correspondent au centre des régions calculées précédemment. A partir de ces germes, on peut étendre les régions, en fonction d'un critère d'homogénéité. Il s'agit à nouveau du niveau de gris de chaque région.

4.4.1 Croissance de région

Pour cet algorithme, on parcourt la liste des pixels germe. Pour chaque point de départ, on créé la liste des pixels voisins. Puis tant qu'il reste des voisins qui n'appartiennent pas encore à une région et qui respectent le critère d'homogénéité, on les ajoute à la région. A chaque fois qu'un des voisins est ajouté à la région, on ajoute ses propres voisins à la liste des pixels

à analyser. De plus, pour améliorer la vitesse d'exécution de l'algorithme, on ignore les régions apparaissant en noir sur l'image. En effet, ces régions peuvent correspondre à 2 types d'éléments différents. Tout d'abord, la zone noire la plus grande correspond au tour de l'image, avec la coupe du patient au centre. Cette zone est délimitée par les régions consituant la peau et les muscles du patient sur le scanner. Le fait de ne pas analyser les régions en noir permet donc d'éviter l'analyse de la moitié de l'image environ, sans risque de perte d'informations. Ensuite, il existe des zones apparaissant en noir à l'intérieur du corps du patient. Ces régions correspondent aux zones vides (l'intérieur des artères, de l'estomac...)
DEMANDER AU CHIRURGIEN POUR ZONES NOIRES DANS LE CORPS SI C'EST BIEN CA
L'intérieur de ces régions ne peuvent pas être segmentés car il possède une couleur uniforme. Mais à nouveau, les contours seront représentés par les contours des régions qui entourent cette zone. Le fait de ne pas analyser les parties noires de l'image permet donc de réduire de moitié la quantité de données à analyser sans risquer de perdre des informations.

AJOUTER IMAGE EXEMPLE

4.4.2 Algorithme

L'algorithme de croissance de région utilise un tableau d'objets *"Pixel-Region"*. Ce tableau d'objet permet de garder facilement en mémoire l'id de la région à laquelle appartient un pixel. Si un pixel n'est pas encore affecté à une région, ce paramètre vaut -1.

Algorithm 3 regionsGrowing()

for all *region* from $m\_regions$ **do**
  $vector < PixelRegion > neighbors = createListNeighbors(pixel, matPixels)$;
  for all *neighbor* from *neighbors* **do**
    if $(matPixels[coordinate.getY() * image.cols + coordinate.getX()].getIdRegion() == -1$ **then**
      if $isHomogeneous(region, neighbor)$ **then**
        modifier l'id du pixel dans la matrice pour le passer de -1 à l'id de la région
        ajouter neighbor dans la liste de pixels de la région
        ajouter les pixels voisins de *neighbor* dans la liste *neighbors*
      end if
    end if
  end for
end for

4.5 Interface graphique

METTRE DES IMPRESSIONS D'ECRAN ICI POUR POUVOIR LES COMMENTER

Chapitre 5

Comparaison avec d'autres techniques de segmentation

Il exister d'autres façon de segmenter une image. Pas seulement dans les technologies et algorithmes employés, mais également dans la manière de faire. En effet, une segmentation totalement automatique n'existant pas encore, il est nécessaire, en fonction du besoin, de faire quelques compromis.

5.1 Segmentation manuelle

La segmentation manuelle d'une image reste la technique la plus efficace car sa seule limite est la compétence de la personne qui effectue les manipulations. La personne compétente va pouvoir parcourir l'image et retenir les parties qui sont importantes. Par exemple dans le cadre de notre projet, le médecin peut aller directement sélectionner le rein sur l'image du scanner, et le logiciel va mettre en surbrillance les contours du rein de manière certaine. En cas de présence de tumeur, celle-ci est assez facilement décelable de part la présence certaine du rein à tel endroit. Le principal problème que pose ce type de segmentation est le temps qu'elle prend ainsi que le fait qu'elle réquisitionne une personne le temps du traitement. Dans le cadre d'un scanner, il y tellement d'image pour un patient que cette méthode n'est absolument pas rentable niveau temps.

5.2 Segmentation avec le logiciel OsiriX

Certains logiciels existent avec pour vocation de pallier au problème vu ci-dessus. Parmis ceux-là se trouve OsiriX. OsiriX est un logiciel de traitement d'image spécialisés dans les fichiers DICOM produits par des équipements médicaux. Il a été créer pour permettre à l'utilisateur de naviguer à travers des vues 2D, 3D et même 4D (plusieurs vues 3D avec un avancement dans le temps). Parmis les rendus 3D, on peut avoir un semblant de réalisme

avec la reconstitution des tissus, simulation des volumes.
Ce logiciel aussi puissant et pratique soit-il ne fait pas de segmentation à proprement dit, seulement du traitement d'images (même si celui-ci est extrêmement puissant et permet énormément de choses). En effet, les processus de segmentation doivent être assistés par l'homme. C'est une personne qui doit sélectionner telle ou telle zone, peut arbitrer sur l'élément choisi, placer un viseur et plusieurs outils de mesure sur l'image pour s'y repérer,... Le logiciel peut donc, dans son étape de segmentation, mettre en valeur les organes désirés. Mais n'est pas spécialement capable d'arbitrer, et c'est justement le problème avec l'automatisation.

METTRE UNE CAPTURE D'ECRAN D'OSIRIX

Chapitre 6

Problèmes rencontrés et limites de l'outil

6.1 Bug croissance région (nom a changer mais j'ai pas mieux pour le moment

Les premières version de l'algorithme de croissance de région présentaient toutes la même erreur dans le résultat. Les régions détectées semblaient être étirées en largeur, mais ressemblaient toutefois à la partie gauche du scanner (donc la droite du patient). Après de nombreuses recherches, et plusieurs versions de l'algorithme, il s'est avéré que l'erreur venait en fait de la façon d'ouvrir l'image. En effet, l'image était ouverte en temps qu'image couleur. La méthode *"getPixelGreyLevel(int x, int y)"* retournait alors le niveau de bleu de l'image. Le niveau de bleu était proche du niveau de gris, et les différences n'étaient pas visibles en affichant cette valeur. Le fait que l'image obtenue ressemblait aux résultats attendus donnait l'impression que l'erreur venait bien de l'algorithme et de la gestion des accès au tableau, au lieu de la technique d'ouverture de l'image.

6.2 Fusion des régions

Actuellement, l'algorithme de fusion permet de compenser la sur-segmentation en regroupant des régions homogènes. Le but de cette étape étant de déterminer l'emplacement des pixels germes pour la croissance de régions, il est nécessaire de pouvoir calculer les coordonnées du centre des régions. Pour cela, la fusion ne se fait que si elle donne une nouvelle région rectangulaire. Une amélioration possible serait de se passer de cette contrainte, et de générer des régions de différentes formes.
Dans ce cas, il devient nécessaire de trouver un algorithme permettant de trouver les coordonnées du centre d'une région ayant une forme quelconque.

De plus, en fonction de la forme de la région, son centre risque de se retrouver à l'extérieur de cette région, par exemple dans le cas de la région correspondant à la vertèbre. Une solution possible serait d'inverser l'algorithme de partitionnement de Voronoi, ou d'utiliser la triangulation de Delaunay.

6.3 Parallèlisation

L'exécution des 3 algorithmes prend un certain temps. Une autre amélioration possible serait de paralléliser une partie des traitements effectués. En effet ils doivent s'exécuter l'un après l'autre de façon séquentielle car chaque algorithme utilise les résultats du précédent, mais le temps de calcul de chacun d'entre eux pourait être réduit en utilisant plusieurs threads.

Chapitre 7

Conclusion

Le travail effectué pendant ce projet nous a permis d'assimiler de nouvelles technologies, que ce soit dans les domaines de l'informatique, des mathématiques ou du médical. Mais pas seulement, en effet nous avons maintenant des connaissances minimes en anatomie humaine, en éthique et surtout une grande fierté d'avoir pu participer à un projet extrêmement concret qui plus est dans le domaine de la médecine.
Automatiser complètement un diagnostique médicale ne requiert pas seulement de très bons algorithmes de segmentation d'image. En effet, une étape d'arbitrage est cruciale pour déterminer la position des différents organes et éventuelles tumeurs. Cet arbitrage peut se présenter sous forme d'un système multi-agents de superposition d'image, d'une intelligence artificielle,... Notre travail était de préparer le terrain à ce type de système en effectuant les traitements nécessaires à la mise en valeur des différents éléments se trouvant dans un cliché issu d'un scanner. Nous avons ainsi pu nous rendre compte des enjeux que présente ce projet et de sa place importante dans un projet de plus grande envergure d'automatisation du diagnostic.

Chapitre 8

Résumé

Ce projet a pour but de réaliser un logiciel de segmentation d'images médicales provenant d'un scanner. L'objectif est de pouvoir diagnostiquer un cancer du rein chez l'enfant au moyen d'outils de traitement d'image. En plus d'un état de l'art, nous avons choisi les algorithmes, les bibliothèques logicielles et les frameworks à utiliser. Ce travail s'intègre dans un projet de plus grande envergure visant à automatiser ce diagnostique en ajoutant de l'intelligence artificielle distribuée.
Mots clefs : segmentation, DICOM, Qt, C++, OpenCV, rein, scanner, division-fusion, k-means, croissance de région

This project aims to realize a segmentation software of medical images provided from a CT-Scan. The goal is searching for children's kidney tumor thanks to images processing tools. In addition to a state-of-theart, we chose algorithms, software libraries and frameworks to use. This work is part of a larger project concerning on automate this diagnosis by adding distributed artificial intelligence.
Keywords : segmentation, DICOM, Qt, C++, OpenCV, kidney, CT-Scan, split and merge, k-means, region growing

Chapitre 9

Lexique

— **Abdomen** : région du corps humain située sous le diaphragme.
— **Bruit** : parasites nuisant à la qualité d'une image.
— **Chimiothérapie** : traitement par des substances chimiques.
— **CMake** : système de construction logiciel, permet la compilation d'un code source.
— **Critère d'homogénéité** : critère arbitraire permettant de classifier des éléments. Ici, le niveau de gris permet de regrouper les pixels.
— **Delaunay** : méthode de segmentation.
— **DICOM** : norme standard pour la gestion informatique des données issues de l'imagerie médicale.
— **Histogramme** : représente la distribution des intensités (ou des couleurs) de l'image.
— **Imebra** : bibliothèque logicielle permettant de manipuler les fichiers DICOM.
— **Immunothérapie** : emploi préventif de sérums spécifiques.
— **JPEG** : norme qui définit le format d'enregistrement et l'algorithme de décodage pour une représentation numérique compressée d'une image fixe.
— **Métastases** : dissémination à distance de cellules infectées, cancéreuses, déjà présentes dans l'organisme.
— **OpenCV** : bibliothèque logicielle utilisée pour le traitement des images fixes ou animées.
— **Osirix** : logiciel de traitement d'images DICOM.
— **Parallèlisme** : action de faire exécuter plusieurs processus en parallèle pour optimiser le temps d'exécution d'une application.
— **Partition** : ensemble, classe homogène.
— **Partition de Voronoi** : découpage du plan (pavage) en cellules à partir d'un ensemble discret de points appelés "germes".
— **Pixel germe** : pixel servant de référence à un calcul ultérieur.
— **Qt** : framework utilisé pour la mise en place d'interface graphique.

— **radiothérapie** : traitement thérapeutique à base de radiations.
— **Région homogène** : étendue de données possédant des caractéristiques particulières et similaires qui en font une unité distincte.
— **Rétropéritoine** : relatif à la cavité du péritoine.
— **SAIAD** : Segmentation Automatique de reins tumoraux chez l'enfant par Intelligence Artificielle Distribuée.
— **Scanner** : appareil de radiodiagnostic.
— **Segmentation** : opération de traitement d'images qui a pour but de rassembler des pixels entre eux suivant des critères pré-définis.
— **Viscère** : nom générique des organes internes contenus dans le ventre, la tête, le thorax.
— **Voronoi** : mathématicien connu pour son diagramme de Voronoi qui permet de diviser une surface en polygones convexes.

Chapitre 10

Bibliographie

— http ://www.ac-grenoble.fr/disciplines/sti-biotechnologies/pages/Imagerie_Medicale_ST2S/I
— https ://openclassrooms.com/courses/modifier-une-image-pixel-par-pixel
— https ://imebra.com
— http ://urfist.enc.sorbonne.fr/anciensite/image_numerique/segmentation.html
— http ://info-radiologie.ch/cancer-rein-scanner.php
— http ://www.ncbi.nlm.nih.gov/pmc/articles/PMC3319195