Université de Caen Basse-Normandie

Analyse d'images thermiques

Master 2 Informatique
Réseau, Application Documentaire, Ingénierie et Sécurité

Année 2011-2012

Projet annuel de SITHONGSOUK Alex
Tutoré par Mr CHAHIR Youssef

Remerciements

je souhaite adresser mes sincères remerciements à Mr CHAHIR Youssef pour son soutien et ses conseils apportés tout au long de ce projet. Son expérience dans le domaine a permis de mener à bien ce projet. En effet, le domaine de l'imagerie informatique et surtout le traitement d'image reste quand même un domaine assez mal connu pour ma part. J'ai beaucoup appris durant ce projet très intéressant et enrichissant.

Table des matières

1 Introduction

L'imagerie informatique touche des domaines aussi divers et variés qu'intéressants mais complexes. La compréhension de la notion d'image en elle-même doit être à son paroxisme. Dans le cadre de la formation RADIS, nous avons eu accès à des cours en traitement d'images dispensés par Mr CHAHIR. J'ai par conséquent choisi en tant que projet annuel, un projet en imagerie.

Mon projet intitulé "Analyse d'images thermiques" implique la manipulation d'images thermiques. Ces dernières contiennent de l'information qu'il faudra extraire. Ceci est le but de ce projet. Au cours de celui-ci, j'ai dû m'organiser et mettre en commun mes compétences acquises durant mon parcours universitaire à l'UCBN afin de rendre la réalisation des objectifs la plus efficace possible.

Ce document présente les différents aspects et spécifications de l'analyse d'images et plus particulièrement des images dites thermiques. Il est construit de la manière suivante : tout d'abord situons le projet et son contexte. Par la suite, ce rapport décrit les outils utilisés. Puis, une vision détaillée et technique permet d'entrer dans le cœur du sujet, traitant des points techniques abordés et des résultats obtenus. Il énumère ensuite les difficultés rencontrées et, pour finir, fait un bilan général du projet.

2 Projet

2.1 Contexte

Ce projet permettrait le progrès dans différents domaines divers et variés. Par exemple, la médecine en fait partie afin de savoir localiser les douleurs chez les bébés (plus particulièrement les bébés prématurés) car ils ne peuvent pas parler. On peut aussi citer les détecteurs de mensonge ou bien la reconnaissance faciale. Les détecteurs de stress, fatigue et/ou fièvre sont aussi dans l'optique de ce projet.

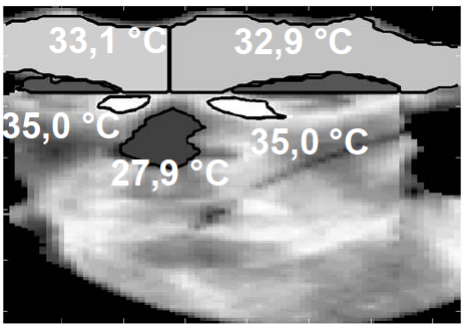

FIGURE 2.1 – Image thermique de rayon infrarouge d'un visage

FIGURE 2.2 – Image thermique de rayon infrarouge d'une personne

FIGURE 2.3 – Image thermique de rayon infrarouge d'un bébé

La sécurité peut aussi tirer profit de ce projet. La surveillance nocturne utilise le procédé de caméra thermique et ainsi l'analyse des images obtenues. L'armée fait aussi appel à des technologies basées sur des capteurs thermiques par exemple.

FIGURE 2.4 – Utilisation de caméra thermique pour détecter un comportement suspect en endroit sombres

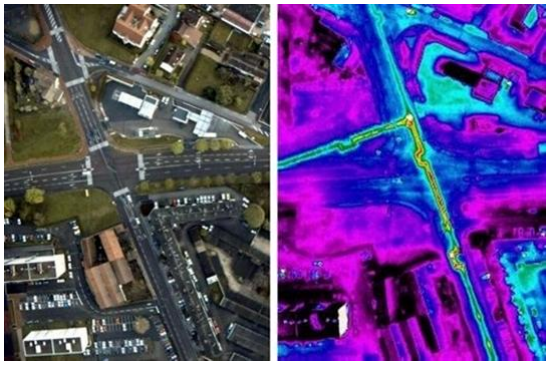

FIGURE 2.5 – Image thermique depuis un hélicoptère

En information géographique, des scientifiques en viennent souvent à traiter des images thermiques telles que des vues de la Terre depuis des satellites ou stations spatiales pour faire une analyse de l'évolution de la température de notre chère planète ou tout autre analyse géographique ou géologique.

On utilise aussi des procédés d'études bioinformatiques en utilisant les images thermiques comme certains examens médicaux.

Les procédés d'analyse d'images thermiques sont quasiment aussi vieux que l'informatique. Pourtant, ils restent très efficaces même si peu d'études en leur égard y sont consacrées.

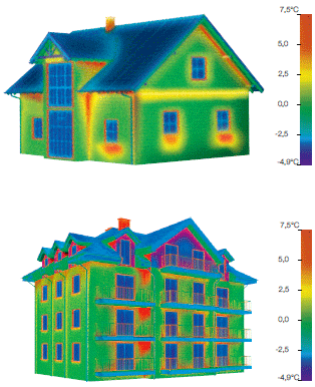

FIGURE 2.6 – Image thermique pour vérifier les ponts thermiques d'une maison

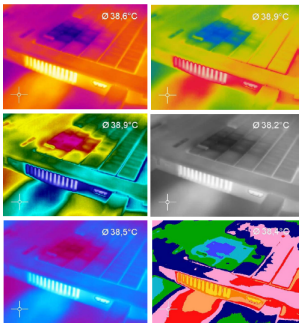

FIGURE 2.7 – Image thermique pour vérification en usine d'ordinateurs portables

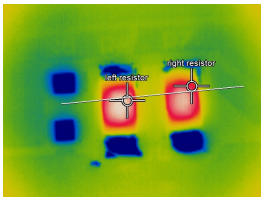

FIGURE 2.8 – Image thermique pour vérification en usine de résistances

2.2 Étapes du projet

On peut caractériser l'analyse d'images en six étapes importantes :
- L'obtention de l'image thermique.
- La segmentation de l'image.
- La labellisation des régions.
- Extraction des régions.
- Extraction des données.
- Présentation des données sous forme de documents.

L'obtention de l'image thermique. L'obtention d'une image thermique est un procédé de capture d'un évènement physique thermique en un signal analogique qui sera ensuite traduit en signal numérique pour créer la fameuse image numérique thermique. Le projet ne couvrera pas ce point.

La segmentation de l'image La segmentation d'image est la division d'une image en zones homogènes afin de séparer les divers composants visibles et de les identifier. Il permet donc d'isoler les objets d'intérêt du fond de l'image. On a donc pour but de former un masque qui permettra d'éliminer le fond. On joue ainsi sur les différentes caractéristiques de l'image pour différencier le fond des objets. Des outils seront présentés dans la suite de ce rapport pour permettre un tel procédé sur une image.

La labellisation des régions Cette étape permet d'étiqueter les différentes régions obtenues par segmentation. Une région correspond à un objet d'intérêt extrait précédemment par la segmentation. La labellisation est très importante, elle permet de différencier un objet d'intérêt d'un autre par les différences entre les caractéristiques des objets. Des outils seront présentés plus tard dans ce rapport pour permettre un tel procédé sur une image.

Extraction des régions Une fois la labellisation terminée, il faut extraire les objets, les isoler pour pouvoir les traiter individuellement. La notion floue de différenciation d'objets montre qu'il est important de tester différents paramètres de la méthode d'extraction d'objets, sous peine d'avoir un résultat érroné. Des outils seront aussi présentés plus tard dans ce rapport pour permettre un tel procédé sur une image.

Extraction des données Cette étape permet d'extraire toute information importante et caractéristique d'une région détectée. Ces informations vont permettre de faire une analyse complète de l'image. C'est le but ultime de ce projet. Le plus important des objectifs de ce projet a donc été énoncé. Des outils seront aussi présentés plus tard dans ce rapport pour permettre un tel procédé sur une image.

Présentation des données sous forme de documents Pour faciliter l'analyse des données ainsi récoltées, il faut les organiser dans des documents pour faciliter la visualisation des données. Un graphe est quand même plus lisible qu'un fichier de données brutes.

Image thermique

21,7	21,2	20,9	29,3	34,1
21,7	22,0	30,1	34,2	34,0
21,6	21,8	30,3	34,2	34,1
21,3	21,4	31,1	34,3	35,9
20,9	21,2	21,4	31,4	37,2

FIGURE 2.9 – Organisation des données dans un tableau représentant une image thermique

Ainsi, toutes les étapes ont été décrites. Le projet s'intéresse plus particulièrement aux étapes de la segmentation jusqu'à la présentation des données sous forme de documents.

2.3 Différents types de segmentation

Comme dit précédemment, la segmentation d'image est une opération de traitement d'images qui a pour but de rassembler des pixels entre eux suivant des critères pré-définis. Les pixels sont ainsi regroupés en régions, qui constituent un pavage ou une partition de l'image. Il peut s'agir par exemple de séparer les objets du fond. Si le nombre de classes est égal à deux, elle est appelée aussi binarisation. On peut classer les différents types de segmentation comme ceci :

2.3.1 Segmentation par approche « région »

Les méthodes appartenant à cette famille manipulent directement des régions. Soit elles partent de quelques régions, qui sont amenées à croître par incorporation de pixels jusqu'à ce que toute l'image soit couverte, et on parle alors de méthodes par croissance de régions, soit ce sont des méthodes fondées sur la modélisation statistique conjointe de la régularité des régions et des niveaux de gris de chaque région. Elles peuvent également partir d'une première partition de l'image, qui est ensuite modifiée en divisant ou regroupant des régions, et on parle alors de méthodes de type décomposition/fusion (split and merge).

FIGURE 2.10 – Segmentation par régions

Les algorithmes par croissance de régions partent d'un premier ensemble de régions, qui peuvent être calculées automatiquement (par exemple, les minima de l'image), ou fournies par un utilisateur de manière interactive (clic de souris par l'utilisateur dans une interface graphique). Les régions grandissent ensuite par incorporation des pixels les plus similaires suivant un critère donné, tel que la différence entre le niveau de gris du pixel considéré et le niveau de gris moyen de la région. Les algorithmes de segmentation par ligne de partage des eaux, développés dans le cadre de la morphologie mathématique, appartiennent à cette catégorie.

FIGURE 2.11 – Segmentation de partage des eaux par itération

FIGURE 2.12 – Segmentation de partage des eaux finale

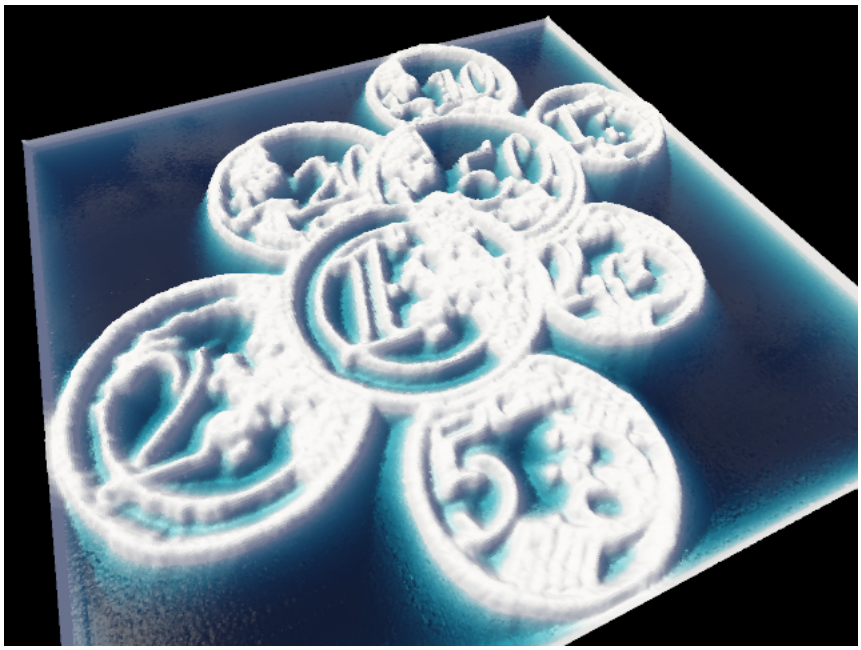

FIGURE 2.13 – Partage des eaux synthétisé pour mieux comprendre le principe

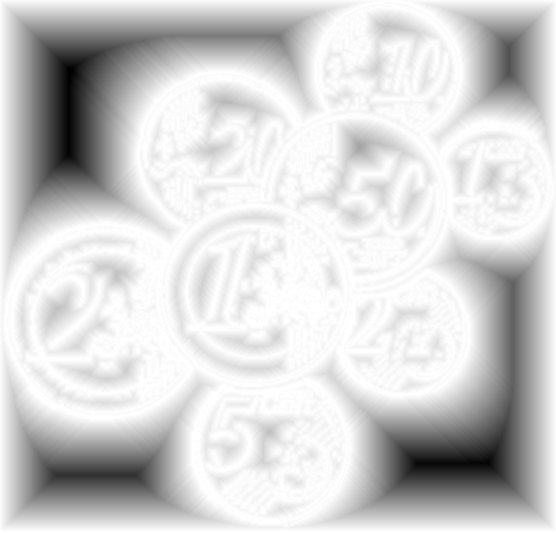

FIGURE 2.14 – Partage des eaux avec le principe de bassins

Les algorithmes fondés sur une modélisation statistique conjointe des régions et des niveaux de gris, notamment ceux s'appuyant sur les Champs de Markov Cachés, reposent sur la minimisation d'une fonction de vraisemblance (ou énergie). Cette fonction prend simultanément en compte la vraisemblance de l'appartenance du pixel à une région considérant son niveau de gris, et les régions auxquelles appartiennent les pixels voisins. Cette fonction effectue un compromis entre la fidélité à l'image initiale et la régularité des régions segmentées.

Les algorithmes de type décomposition/fusion exploitent les caractéristiques propres de chaque région (surface, intensité lumineuse, colorimétrie, texture, etc.). On cherche des couples de régions candidates à une fusion et on les note en fonction de l'impact que cette fusion aurait sur l'apparence générale de l'image. On fusionne alors les couples de régions les mieux notés, et on réitère jusqu'à ce que les caractéristiques de l'image remplissent une condition prédéfinie : nombre de régions, luminosité, contraste ou texture générale donnée, ou alors jusqu'à ce que les meilleures notes attribuées aux couples de régions n'atteignent plus un certain seuil (dans ce dernier cas, on parle d'un algorithme avec minimisation de fonctionnelle).

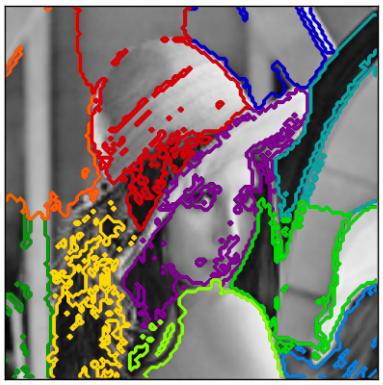

FIGURE 2.15 – Segmentation de split and merge de Lena

2.3.2 Segmentation par approche « frontière »

Cette approche cherche à exploiter le fait qu'il existe une transition détectable entre deux régions connexes.

Les méthodes les plus anciennes utilisent des opérateurs de traitement d'images, tels que le filtre de Canny ou celui de Sobel, pour mettre en évidence les pixels qui semblent appartenir à un contour. La construction d'une partition est alors souvent difficile.

On peut aussi faire intervenir des modèles déformables à l'aide de courbes paramétriques (courbe de Bézier, spline...) ou de polygones (par exemple algorithme à bulle).

Pour initier le processus, on recherche des points remarquables de l'image, tels que des points à l'intersection de trois segments au moins. De tels points sont appelés des graines (ou "seeds").

L'intérêt principal des méthodes de segmentation selon l'approche frontières est de minimiser le nombre d'opérations nécessaires en cas d'itération du processus sur des séries d'images peu différentes les unes des autres (cas des images vidéo notamment). En effet, une fois que les contours des régions ont été trouvés dans la première image, l'application du modèle déformable à l'image suivante est plus efficace que de tout recalculer, si la différence entre les images est peu importante.

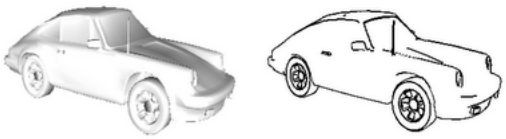

FIGURE 2.16 – Segmentation par contours

2.3.3 Segmentation par classification ou seuillage

On part ici d'un rapport qu'entretient chaque pixel individuellement avec des informations calculées sur toute l'image, comme par exemple la moyenne des niveaux de gris de l'ensemble des pixels, ou la médiane, permettant de construire n classes d'intensité. Lorsque les classes sont déterminées par le choix d'un seuil, on parle de seuillage. Les pixels appartenant à une même classe et étant connexes forment des régions de l'image.

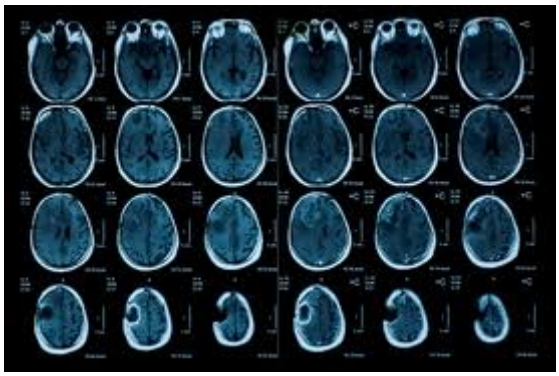

FIGURE 2.17 – Organisation des données dans un tableau représentant une image thermique

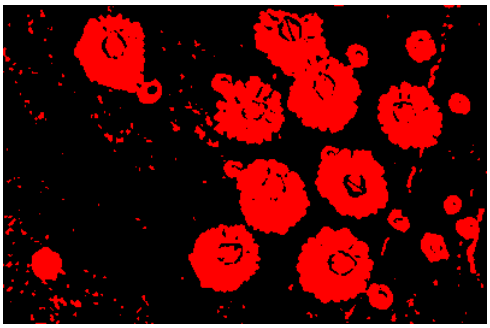

FIGURE 2.18 – Seuillage par entropie

2.4 Explications du choix des quad-tree

L'utilisation de quad-tree requiert un procédé de segmentation par approche « région ». L'algorithme de split and merge est donc la segmentation utilisée, associée aux quad-tree. Cette méthode permet un traitement plus rapide, dû au fait que l'on peut paralléliser tout cela. De plus, l'extraction de donnée se fait en même temps que la segmentation. Il en est de même de la labellisation et l'extraction d'objets. Ainsi, pour ce projet, je vais utiliser l'approche de split and merge associée aux quad-tree.

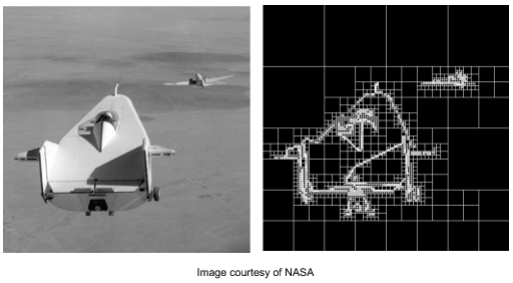

FIGURE 2.19 – Division par quad-tree

Ayant décrit les étapes de mon projet, je vais maintenant m'attarder sur la description des outils utilisés au cours de celui-ci.

3 Outils utilisés

3.1 C++

FIGURE 3.1 – Logo C++

Description Le C++ est un langage permettant la programmation sous de multiples paradigmes comme la programmation procédurale, la programmation orientée objet et la programmation générique. Le langage C++ n'appartient à personne et par conséquent n'importe qui peut l'utiliser sans besoin d'une autorisation ou obligation de payer pour avoir le droit d'utilisation.

La bibliothèque standard du C++ est en grande partie un sur-ensemble des fonctions disponibles dans la bibliothèque standard du C. Elle englobe la Standard Template Library (STL) qui met à la disposition du programmeur des outils puissants comme les collections (conteneurs) et les itérateurs.

Pandore C++ est le premier langage utilisé lors de ce projet. En effet, il est de loin le plus rapide de tous les langages de programmation pour tout ce qui est traitement d'images. De plus, on peut avoir accès à la bibliothèque de pandore développée par le GREYC de l'UCBN. Pandore est une bibliothèque standardisée d'opérateurs de traitement d'images. La version actuelle regroupe des opérateurs traitant d'images 1D, 2D et 3D, en niveaux de gris, en couleurs et multispectrales. Elle se compose d'une collection d'opérateurs exécutables et d'un environnement de programmation en C++. Cet outil permet donc de manipuler et de représenter des images en C++.

FIGURE 3.2 – Logo du GREYC qui partage pandore

En C++, les pointeurs et références sont très intéressantes et surtout très pratiques quand on veut faire une structure de type quad-tree. Cependant, le manque d'expérience a fait que j'ai du changer de langage de programmation car contrairement à Java, il n'existe pas de Garbage-Collector. Ainsi, toutes les allocations et désallocations de mémoire sont à faire par l'utilisateur. Sans habitude, cela peut devenir un véritable calvaire.

3.2 Java

FIGURE 3.3 – Logo Java

Description Le langage Java est un langage de programmation informatique orienté objet. La particularité principale de Java est que les logiciels écrits dans ce langage sont très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. C'est la plate-forme qui garantit la portabilité des applications développées en Java.

Le langage reprend en grande partie la syntaxe du langage C++, très utilisé par les informaticiens. Néanmoins, Java a été épuré des concepts les plus subtils du C++ et à la fois les plus déroutants, tels que les pointeurs et références, et l'héritage multiple remplacé par l'implémentation des interfaces. Les concepteurs ont privilégié l'approche orientée objet de sorte qu'en Java, tout est objet à l'exception des types primitifs (nombres entiers, nombres à virgule flottante, etc.)

Java2D et Image I/O L'API Image I/O pour lire et écrire des images dans des formats comme JPEG et PNG existe dans la librairie standard depuis 2002 dans Java. Quant à Java2D, elle a été intégrée à Java depuis 2008, en même temps que le passage en licence GPL. Ces deux API permettent comme pandore de manipuler des images. Certes, on peut faire beaucoup moins de chose qu'avec pandore, cela reste tout de même suffisant.

Lors de ce projet, j'utilise Java tout simplement parce que durant une grande partie de ma scolarité, Java a été le langage que j'ai étudié. J'ai donc plus de connaissance en Java qu'en C++. Avec un Garbage-Collector, c'est tout de suite plus facile. Cependant, je n'utilise pas Java tout seul, j'utilise aussi un éditeur de type IDE nommé NetBeans.

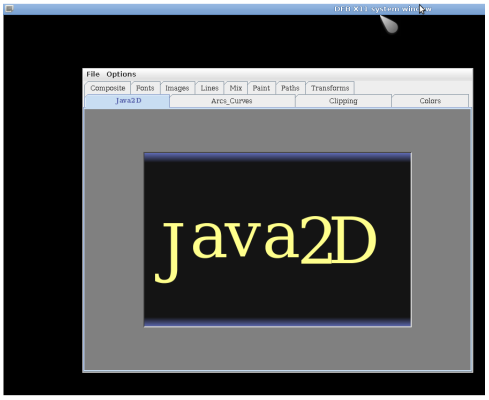

FIGURE 3.4 – Java2D

3.3 NetBeans

FIGURE 3.5 – Logo NetBeans

Description NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDK est requis pour les développements en Java.

Netbeans supporte les principaux systèmes de gestion de versions13 : CVS, Subversion, Mercurial, ClearCase. Il intègre par ailleurs des outils collaboratifs.

L'IDE Netbeans s'enrichit à l'aide de plugins.

Maintenant que tous les outils ont été décrits, regardons le côté technique un peu plus en détail.

4 Détails techniques

4.1 Représentation d'une image

Durant cette partie du rapport, quand on mentionne la notion d'image, on fait alors référence à une image en couleur dans l'espace colorimétrique RVB (ou RGB). La couleur est définie par un triplet de valeurs qui sont les composantes Rouge, Vert et Bleu. Avec des méthodes bien spécifiques, on est capable d'extraire n'importe quelle information de l'image, que ce soit en Java ou bien en C++.

4.1.1 En C++

L'image se représente sous la forme d'un objet qui est défini dans pandore.Il peut être de plusieurs types, composé d'un tableau de flottants, d'un tableau de long non signés, ou encore un tableau de caractères non signés. Ces types sont respectivement Imc2dsf, Imc2dul ou bien Imc2duc.

4.1.2 En Java

L'image se représente sous la forme d'un objet qui est défini dans la bibliothèque standard de Java. On fait appelle à la classe de Java2D qui est BufferedImage, avec cela, on peut obtenir la grille de valeurs qui représente la grille de pixels. En Java, cet objet grille est de type WritableRaster.

4.2 Algorithmes de décomposition / fusion

Lors de mon projet, on a choisi d'utiliser la méthode de segmentation basée sur les quad-tree et sur le split and merge. Par la suite, je vais donc vous détailler les algorithmes du split, puis du merge en pseudo-code.

Tout d'abord, voici le pseudo-code du split :

```
Split

// Au début, le quad-tree est la racine
Entrées: Quad-tree Q, image_en_entrée I
Résultat : aucun (il ne fait que remplir l'arbre donné)

hauteur = hauteur de I
largeur = largeur de I

Im1 = nouvelle image de dimension hauteur/2 x largeur/2
Im2 = nouvelle image de dimension hauteur/2 x largeur/2
Im3 = nouvelle image de dimension hauteur/2 x largeur/2
Im4 = nouvelle image de dimension hauteur/2 x largeur/2


Pour x de 0 à largeur/2 Faire
Pour y de 0 à hauteur/2 Faire
Pour chaque composante R, G ou B Faire
ajouter le point (x, y) à Im1
ajouter le point(x+largeur, y) à Im2
ajouter le point (x, y+hauteur) à Im3
ajouter le point (x+largeur, y+hauteur) à Im4
calculer les min, max, moyenne et médiane
FinPour
FinPour
FinPour
Ajouter toutes les caractéristiques calculées pendant la boucle \\
\\ de parcours dans l'arbre
Ajouter les coordonnées x et y du coin supérieur gauche du quad courant
Avec le chainCode, je peux savoir quel fils exactement je suis du père.
Je mets donc à jour les coordonnées du noeud courant.
Test d'homogénéité.
Si c'est vrai
on ne fait rien.
Sinon
on créé les 4 fils du noeud courant
```

```
On leur associe le noeud courant en tant que père
On lance en récursif pour chaque fils le split de lui même avec pour \\
\\ paramètre l'arbre qui est question
FinSi

FinSplit


Merge

Entrées: image_en_entrée I, image_en_sortie S, Quad-tree Q
Résultat: image_en_sortie S

Créer la liste « [K] » des blocs associés aux feuilles du QuadTree
Pour chaque bloc « B » dans la liste « [K] »
Créer une nouvelle région « [R] »
Retirer « B » de « [K] » et l'ajouter dans « [R] »
Calculer la valeur/couleur moyenne de « [R] »
Créer la liste « [N] » des blocs voisins du bloc « B »
Pour chaque bloc « Bn » dans la liste « [N] »
Si (« Bn » est dans « [K] ») ET (« Bn » et « [R] » sont similaires) Alors
Retirer « Bn » de « [K] » et l'ajouter dans « [R] »
Ajouter les blocs voisins de « Bn » dans la liste « [N] »
Recalculer la valeur/couleur moyenne de « [R] »
Fin Si
Fin Pour
Fin Pour

FinMerge
```

Le split va découper jusqu'à l'unité qui est le pixel s'il y a besoin. Tant que le quad courant n'est pas homogène, on découpe en quatre.

Le merge quant à lui va fusionner les quad voisins de la région courante.

FIGURE 4.1 – Décompositions successives des blocs

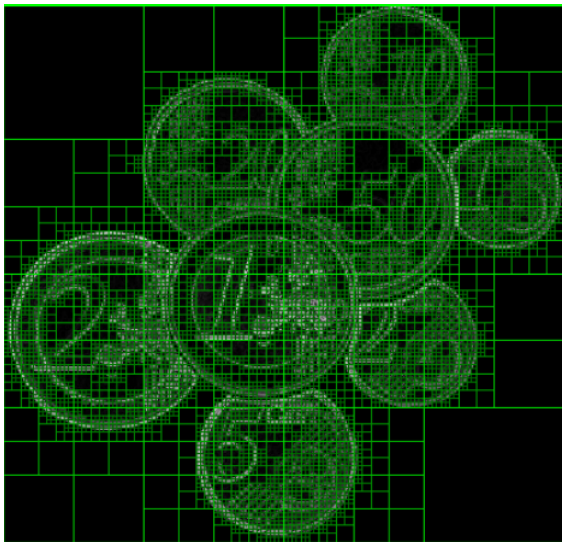

FIGURE 4.2 – Décomposition finale

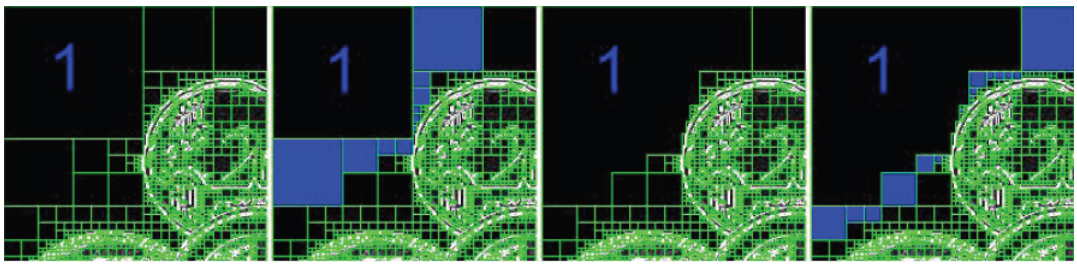

FIGURE 4.3 – Agrégation itérative des blocs similaires au bloc 1

4.3 Utilisation de quad-tree

On utilise les quad-tree car on peut modéliser la profondeur aisément. Chaque itération du split équivaut à un niveau de la profondeur du quad-tree.

De plus, à chaque itération du split, on ne prend que l'arbre qui va être modifié. Les noeuds de niveau supérieur ne seront pas modifiés. Du coup, on peut multithreader le split sans utiliser de mutex sur le quadtree, vu que chaque split récursif est indépendant des autres. Cela implique que le codage du multithreading est facilité.

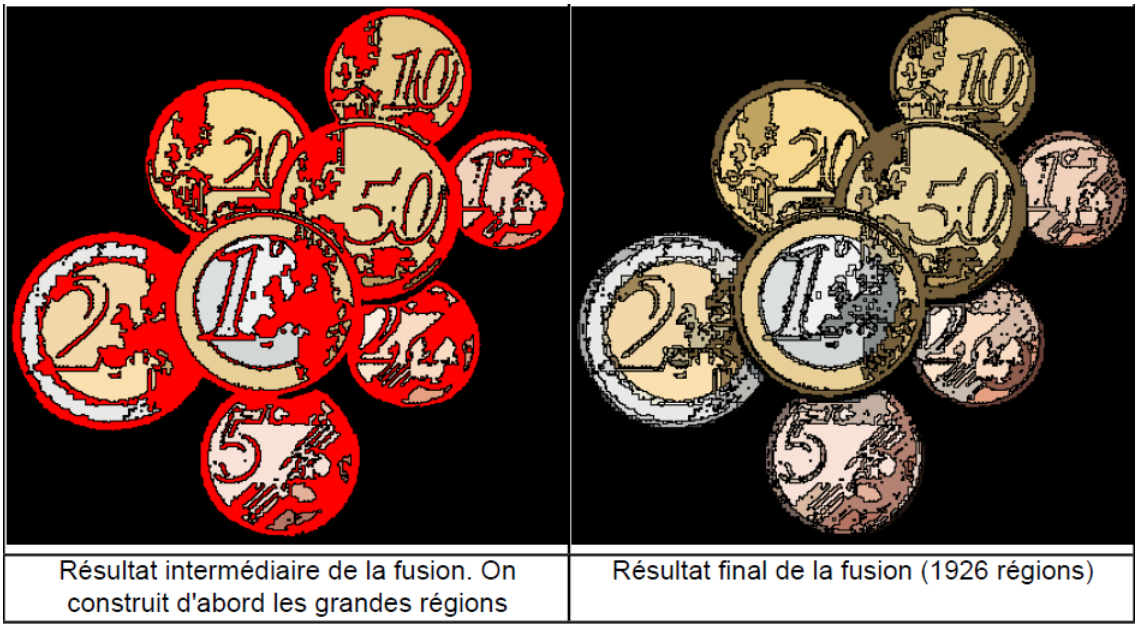

FIGURE 4.4 – Fusion finale

Ainsi, l'utilisation du quad-tree est tout à fait bénéfique pour ce projet. En effet, on récolte les informations de l'image en même temps que le split. On a donc l'extraction des données en même temps que le split and merge.

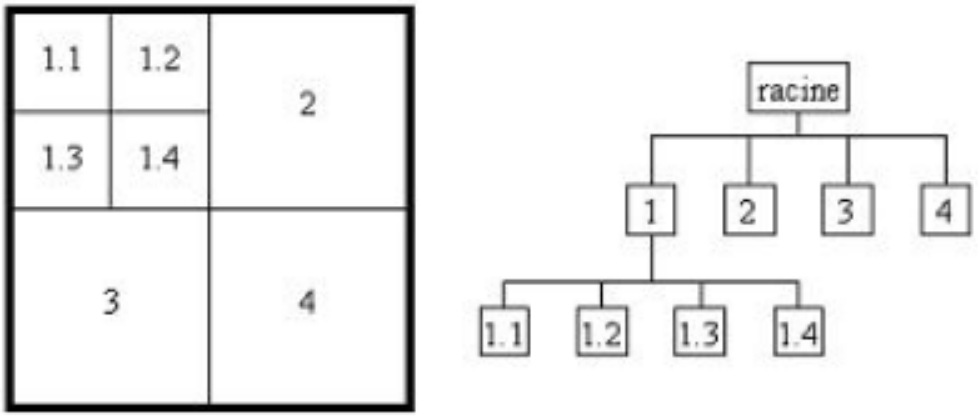

FIGURE 4.5 – Représentation en quad-tree

4.4 Recherche de voisins

4.4.1 Méthode géométrique avec listing des feuilles

La méthode suivante est la méthode utilisée actuellement, elle est aussi la méthode la moins compliquée à implémenter ainsi qu'à comprendre. On va résonner logiquement sur

les coordonnées géométriques. En effet, pour un quad donné, on peut à partir d'une liste de feuilles du quad-tree repérer les voisins. Pour chaque voisin, on regarde ses coordonnées x et y. On les compare aux x et y du quad donné. Les tests sont les suivants :

Si le voisin est celui du Nord, alors :

```
         X <= vX <= X+L
vY+vH = Y  et        ou
    X <= vX+vL <= X+L
```

Si le voisin est celui du Sud, alors :

```
         X <= vX <= X+L
Y+H = vY  et        ou
    X <= vX+vL <= X+L
```

Si le voisin est celui de l'Est, alors :

```
         Y <= vY <= Y+H
X+L = vX  et        ou
    Y <= vY+vH <= Y+H
```

Si le voisin est celui de l'Ouest, alors :

```
         Y <= vY <= Y+H
vX+vL = X  et        ou
    Y <= vY+vH <= Y+H
```

où vX est la coordonnée X du voisin, vY la coordonnée Y du voisin, vH la hauteur du voisin et vL la largeur du voisin.

Si on résume tous ces tests, supposons que c'est un voisin vertical alors leurs coordonnées en Y se confondent en une ligne. Il ne reste plus qu'à tester si au moins l'un des sommets du voisin à sa coordonnée vX qui est comprise entre les deux sommets du quad courant qui sont en parallèle avec les sommets du voisin.

Il ne reste plus qu'à faire la même chose pour les voisins horizontaux sauf qu'on échange les tests des X et des Y et vice-versa.

4.4.2 Graphe d'adjacence

Rien de plus facile à utiliser car tous les liens de voisinages sont inscrits dans le graphe d'adjacence. En effet tous les arcs représentent le lien de voisinage d'un quad à un autre.

Le plus dur est de produire ce graphe. Apparemment, on peut le construire assez facilement par l'utilisation de chain Code ou bien de la clé de Peano, que je vais décrire prochainement.

4.4.3 Chain Code et clé de Peano

Rien de plus facile que d'implémenter les chain Code car à chaque fois que l'on va "split" et ajouter les quatres fils, on va ajouter à ce fils un chain Code qui lui est propre et qui est composé du chain Code de son père en y ajoutant au bout le numéro qui lui correspond entre 1 et 4.

En effet, on démarre de la racine que l'on note 0 pour ne pas la prendre en compte. Ensuite, c'est soit 1, 2, 3 ou 4 en fonction de sa position dans le quad parent.

Voici un schéma qui montre comment donner ce dernier chiffre au chain Code tout en suivant l'ordre de Peano qui suit un "Z".

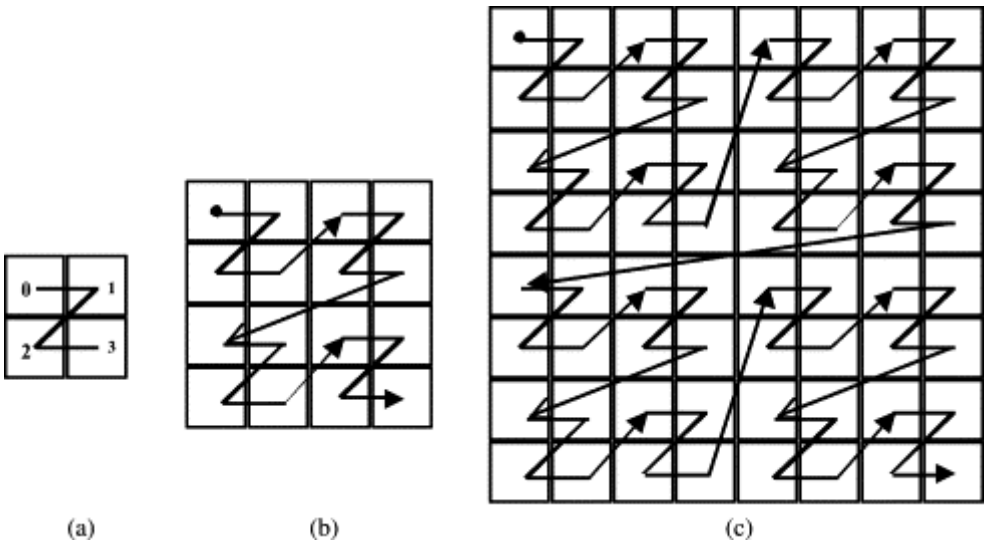

FIGURE 4.6 – a) ordre de Peano, b) et c) sont le parcours de Peano

Avec des algorithmes spécifiques, on peut retrouver les voisins sans parcours de l'arbre. Ainsi, on atteint un voisin en temps constant. La clé de Peano se construit de la manière suivante : à partir d'une position en x et y, on va prendre leur représentation en chaîne de bits. Ensuite, on commence par prendre le bit de poids faible de y, puis celui

de x, ensuite, on décale à gauche et on prend le bit en cours, d'abord de y puis de x, ainsi de suite...

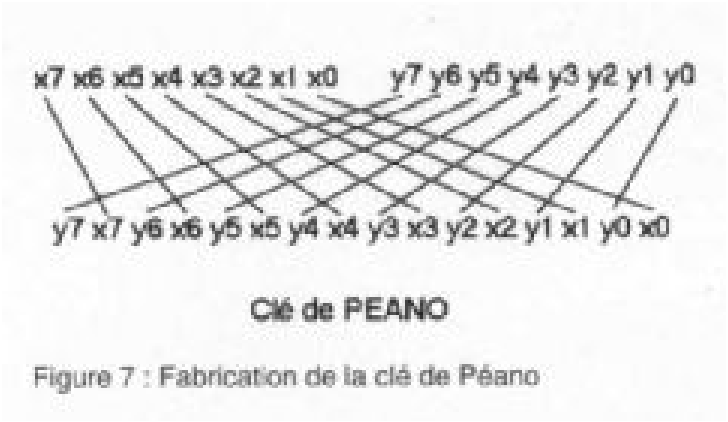

FIGURE 4.7 – Création de la clé de Peano

4.5 Les critères d'homogénéité

La mesure de la similarité de deux pixels est le point clé des techniques de segmentation. Il convient donc de choisir avec soin la méthode de calcul de la similarité.

L'approche la plus simple consiste à définir comme similaires des pixels qui sont visuellement proches : Pour une image en niveau de gris, il faut calculer la différence entre les valeurs. Si cette différence est inférieure à un certain seuil, les pixels seront jugés similaires. Cette méthode est équivalente au calcul de la norme du gradient de l'image en chaque point et à seuiller le résultat obtenu. (Distance Euclidienne)

Pour une image RGB, il faut calculer la différence entre les couleurs. Pour cela il est préférable de convertir les valeurs RGB des pixels en valeurs TSL. Le " T " représente la teinte de la couleur (palette du rouge au bleu), le " S " représente la saturation (du pastel à l'intense) et le " L " représente la luminance (du clair au foncé).

La comparaison des teintes seules n'est pas suffisante. En effet si la saturation est faible, les teintes n'ont plus beaucoup d'influence sur la couleur : les couleurs se rapprochent du blanc-gris-noir et la différence se fait alors par la luminance. Il faut donc une formule qui effectue une comparaison sur les teintes lorsque la saturation est forte, et sur les luminances lorsque la saturation est faible. Par exemple la formule de Carron utilise une sigmoïde. Une autre distance est celle utilisant la variance.

$$\boxed{\text{distance(color0,color1)} = \text{alpha} * \Delta(h0, h1) + (1 - alpha) * \Delta(l0, l1)}$$

$$\text{color0} = \{\ \text{h0},\ \text{s0},\ \text{l0}\ \}$$
$$\text{color1} = \{\ \text{h1},\ \text{s1},\ \text{l1}\ \}$$

$$a(s) = \frac{\pi/2 + ArcTan(rate * (s - offset))}{\pi}$$

$$alpha = \sqrt{a(s0) * a(s1)}$$
$$\Delta(h0, h1) = min(|h1 - h0|, 1 - |h1 - h0|)$$
$$\Delta(l0, l1) = |l1 - l0|$$

FIGURE 4.8 – Formule de Carron

```
Homogénéite( R ) = Variance( pixels de R ) = Moyenne(   distance( pixels de R , Moyenne( pixels de
 R )  ) ^2 )
```

FIGURE 4.9 – Distance utilisant la variance

Maintenant, jetons un petit coup d'oeil aux difficultés rencontrées.

5 Difficultés rencontrées

5.1 C++

Le C++ a été la plus grosses difficulté rencontrée. N'ayant pas l'expérience nécessaire en C++, je me voyais mal continuer à coder. Les notions de pointeurs et références sont difficiles à prendre en main. La notion la plus difficile à appréhender reste l'absence de Garbage-Collector. Bien sûr, gérer la mémoire est quelque chose de nouveau.J'espère régler ce problème pour un futur projet en C++.

5.2 Complexité de recherche de voisins

La notion mathématique de recherche de voisins est très dur à cibler. Il est impossible à notre niveau de connaissance de rédiger un algorithme qui demande du temps pour le comprendre et surtout de la maîtrise pour l'implémenter sans erreur. N'ayant pas réussi à trouver l'algorithme permettant de construire le graphe d'adjacence, je suis parti sur la piste de la clé de Peano sans succès. Je me suis donc dirigé vers la solution géométrique avec listing des feuilles du quad-tree.

5.3 Compréhension de la clé de peano

L'utilisation de la clé de Peano reste obscure pour ma part. Je sais qu'il sert à trouver un noeud voisin à partir d'un algorithme car le chemin de Peano suit toujours un ordre donné. Ainsi, on peut prédire le voisinage.Malheureusement, je n'arrive pas à trouver l'algorithme ou bien le lien logique entre le résultat de la clé de Peano et les voisins de ce quad.

5.4 Taille de l'image de rendu des segmentations

Dernière difficulté rencontrée durant ce projet, j'ai un facteur de redimensionnement de 0.5 qui traîne quelque part dans mon code. Seulement, après avoir reparcouru maintes et maintes fois mon code, je ne suis toujours pas en mesure de trouver l'erreur.

Après avoir exprimé les difficultés rencontrées, il est l'heure de faire le bilan général de ce projet.

6 Bilan général

6.1 Ce qui a été réalisé

6.1.1 Segmentation en quadtree

La segmentation en quadtree fonctionne convenablement. J'ai un quad-tree en retour qui est bien rempli. Le split est plutôt rapide car il est multithreadé. Le merge renvoie bien une image de même dimension.

6.1.2 Extraction des données

Comme dit ci-dessus, mon quad-tree est bien rempli. Il peut donc retourner de l'information. On a donc réussi à extraire des données en même temps que la segmentation de l'image.

6.2 Ce qui reste à faire

6.2.1 Correction de l'image de rendu des segmentations

Le soucis avec le merge, c'est qu'il renvoie une image où le résultat n'est que dans le cadran supérieur gauche. Il y a donc un problème de facteur 0.5 à trouver pour régler le problème.

6.2.2 Produire des graphes pour chaque région

Ce problème est la conséquence directe de celui cité précédemment. En effet, j'ai préféré m'attarder sur la résolution du problème de la redimension du résultat plutôt que de continuer à remplir mes objectifs pour ce projet, cet objectif étant mineur.

7 Conclusion

Pour conclure, ce projet m'a permis d'expérimenter le C++ ainsi que le domaine du traitement d'images en conditions réelles. Bien que mon programme soit perfectible, j'ai trouvé mon projet très enrichissant, découvrant beaucoup de choses. Cela m'a conforté dans ma perspective de carrière qui est de travailler dans le domaine de l'imagerie numérique et plus spécialement dans l'imagerie médicale. Il m'a aussi montré que j'avais encore beaucoup à apprendre à ce sujet et que cela ne sera pas simple.