Norbert KOUANDOU
Matthias LARCHER
Yana POPOVA
Sabrina WONS

Chargés de TD:
Pascal DESBARATS
Anne VIALARD

Client:
Achille BRAQUELAIRE

April 7, 2006

Résumé

Les images numériques 2D issues d'échantillonage de données réelles présentent des défauts de crénelage. L'application d'un filtre d'agrandissement sur de telles images sans traitement distinct des régions rend les frontières indésirablement floues ou crenelées. Afin de préserver leur netteté durant le redimensionnement, on agrandit séparément les régions (objets) en gardant la meilleure approximation possible de la frontière réelle. Le défaut de flou apparaissant à l'intérieur des objets, n'est non seulement plus un problème, mais permet même d'obtenir un meilleur résultat puisqu'ils sont constitués de dégradés de couleurs. Nous considérons ici, le cas d'images préalablement segmentées avec des frontières bien nettes, accompagnées de leurs etiquettes.

La segmentation de l'image permet le suivi du contour discret des objets à traiter. La construction d'un chemin euclidien dirigé par la tangente sur ce contour approxime au mieux la frontière réelle, ce qui atténue d'une part l'effet d'escalier ou de flou engendré après agrandissement, et d'autre part permet le calcul du canal alpha utilisé dans l'antialiasing lors de l'incrustation de l'objet sur un fond.

Au final, cette application permet d'agrandir une image segmentée en limitant les désagréments inhérents à l'agrandissement sur les bords des régions et donc d'améliorer la qualité visuelle de l'image.

Mots-clés : agrandissement, chemin euclidien, aliasing (crénelage), canal alpha.

Abstract

Two-dimensional digital pictures sampled from real data all suffer from aliasing problems. Applying a resizing filter on such pictures in a global way (without defining regions) blurs the borders or make aliasing appear. In order to preserve clearness during resizing, we resize sparately the regions (objects) while keeping the best approximation of the real border. The blur effect generated inside the objects, is no more a problem, it's even producing a better result as they ar made of colour gradients. Here we consider the case of previously segmented pictures with clear borders, associated to a file containing their labels.

Picture segmentation allows the following of the discrete border of the objects. The construction of an euclidean path directed by the tangent on this edge approximates at best the real border, on one hand this attenuates the aliasing effect and makes the blur unnoticeable after resizing, and on the other hand this allows the computation of an alpha canal used to apply antialiasing when recomposing the object with a background.

To conclude, this application allows to increase the size of a segmented picture while reducing the inherent nuisance of enlargement on the regions borders, hence improving the visual image quality.

Key words : zoom, Euclidian path, aliasing, alpha channel, enlargement.

Sommaire

Table des figures

Chapitre 1

Introduction

1.1 Domaines d'application

Depuis son apparition dans les années 70, l'imagerie numérique est devenue un élément indispensable dans le domaine scientifique, mais aussi de la vie courante. On appelle "image numérique" toute image stockée sous forme binaire. Elle peut être acquise par un capteur CCD (Charged Coupled Device), ou créée numériquement. On appelle ces dernières "images de synthèse".

Les avantages des images numériques par rapport aux images dites "classiques" sont nombreux. Leur stockage et référencement est simple et rapide ! Il est bien plus aisé et rapide d'envoyer une image numérique grâce aux réseaux informatiques qu'une image "classique" par voie postale. Elles sont plus faciles à traiter (changer une ou plusieurs couleurs, diminuer ou agrandir la taille, etc.) et à analyser.

La plupart des images numériques sont représentées sous forme d'un ensemble de points appelés pixels, chacun étant caractérisé par sa couleur. Dans la plupart des cas ceux-ci sont placés sur une grille. C'est dans cette représentation que réside le problème majeur de la qualité des images : plus la grille contenant les pixels est fine, meilleure est la qualité de l'image et réciproquement, une grille moins fine représentera une image de qualité inférieure.

En agrandissant une image numérique, c'est en effet sa grille qui change de dimension, ce qui crée un manque d'informations à combler. Ceci conduit à une approximation de valeurs et les contours des objets dans l'image résultante deviennent flous. Ce qui est important, c'est de pouvoir agrandir une image tout en gardant ces derniers nets.

L'agrandissement d'images numériques trouve son application dans plusieurs domaines, qu'ils soient scientifiques ou de la vie courante. Pour donner quelques exemples, prenons d'abord le domaine de la médecine, plus particulièrement la mammographie où le traitement de l'image est uniquement basé sur l'agrandissement car la faible différence de densité des tissus du sein ne permet pas au système numérique de faire des auto-détections. Les radiologues sont obligés d'examiner très finement le cliché du sein avec une loupe pour faire un meilleur diagnostique.

Dans le domaine de la photographie, il existe maintenant un zoom numérique qui consiste à recadrer et agrandir l'image obtenue par le capteur optique grâce à une méthode numérique d'agrandissement. Actuellement le zoom numérique produit des clichés d'une qualité insuffisante par rapport à ceux pris en utilisant uniquement le zoom optique. Une autre application possible est l'agrandissement d'une photo apr
'es la prise de celle-ci. Actuellement les appareils photos numériques ont nettement moins de ressources, alors le cliché pris en utilisant le zoom optique peut être agrandi en utilisant un ordinateur, pour obtenir une meilleure qualité.

En géographie, des satellites sont actuellement utilisés pour prendre des photos de la surface terrestre. La communication entre les laboratoires et ces derniers étant peu rapide, les photographies sont envoyées en "petit" format et agrandies numériquement par la suite.

Dans le domaine de la cartographie, depuis 1993 l'IGN (Institut Géographique National [6]) utilise des avions spécialement équipés de caméras numériques pour la prise de vue aérienne dans le but d'élaboration de cartes. Cette flotte survole des terrains précis à une hauteur de 4800 m, afin de photographier tout le terrain. Les clichés pris peuvent par la suite être agrandis. Le passage à l'imagerie numérique a amélioré la dynamique, la disponibilité des images et la rapidité de leur traitement, en éliminant les phases de développement et de numérisation.

Ces exemples montrent que l'agrandissement des images est un domaine qui touche des secteurs variés sans cesse à la recherche de nouvelles méthodes plus performantes et d'améliorations.

1.2 Analyse de l'existant

L'agrandissement d'images numériques est une opération utile et nécessaire. Agrandir une image, permet d'améliorer le confort visuel de celui qui la regarde, d'augmenter sa résolution ou encore sa taille. En bref, l'agrandissement d'une image permet l'accès à ses détails.

Aussi, selon l'approche que l'on adopte par rapport à l'image à étudier (le contexte d'étude, le système d'acquisition, etc.), et des connaissances que l'on a sur celle-ci (en niveau de gris, couleur, 2D etc..), il existe plusieurs techniques permettant son agrandissement. Leur étude et leur recensement font l'objet de notre analyse de l'existant.

1.2.1 Représentation d'une image

Référence : [7]

Domaine spatial

Une image peut être représentée comme une variation d'intensité dans l'espace. On se réfère alors aux signaux dans le domaine spatial et l'image peut être définie parune fonction f(x, y) qui donne la valeur de l'intensité en fonction des coordonnées (x, y) d'un point donné.

Domaine spectral

Une image peut aussi être représentée dans le domaine fréquentiel. Celui-ci est très complémentaire du domaine spatial et présente une image comme un ensemble de sinusoïdes de fréquences différentes ayant chacune une amplitude et un décalage de phase particuliers. Le passage du domaine spatial au domaine spectral se fait par l'intermédiaire de la transformée de Fourier.

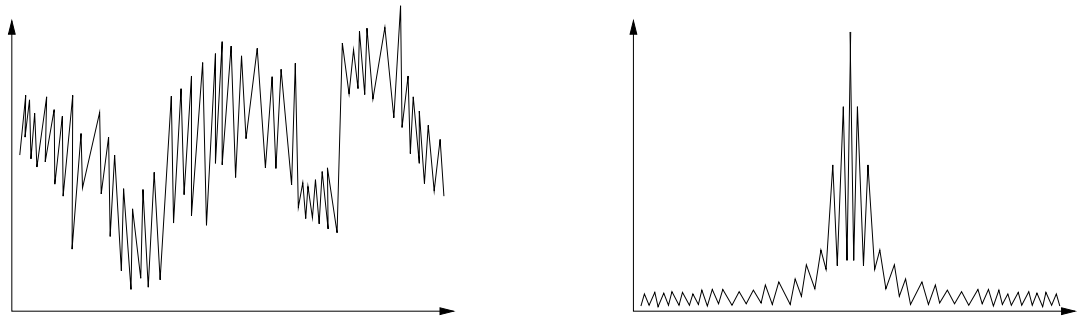

FIG. 1.1 – Signal dans le domaine spatial et son spectre à droite

Transformée de Fourier

La transformée de Fourier est utilisée pour transformer un signal du domaine spatial en un spectre du domaine fréquentiel. Inversement, on peut aussi ajouter les sinusoïdes entre elles afin de reconstituer une représentation spatiale de l'image, c'est la transformée inverse de Fourier (Figure : **??**).

On remarque que la transformée de l'image du babouin est plus claire que celle de Lena, ce qui traduit une présence plus importante de hautes fréquences.

Echantillonage

L'échantillonnage consiste à transformer un signal continu en signal discret, en capturant des valeurs à intervalle de temps régulier. C'est une étape nécessaire pour pouvoir enregistrer, analyser et traiter un signal par ordinateur, car celui-ci ne peut traiter que des nombres. Si la fréquence d'échantillonnage est très faible, les acquisitions seront très espacées et, de ce fait si le signal original comporte des détails entre deux positions de capture, ils ne seront pas enregistrés. C'est pour cela que la fréquence d'échantillonnage doit être bien choisie et suffisamment grande pour restituer correctement l'ensemble des informations transportées par le signal analogique. Le théorème de Shannon montre que toutes les fréquences du signal inférieures à la moitié de la fréquence d'échantillonnage sont correctement restituées.

1.2.2 Méthodes d'interpolation

Zero-padding

La méthode du zero-padding se résume par trois étapes illustrées par la figure 1. Dans la première, il s'agit de passer du domaine spatial au domaine fréquentiel avec la transformée

de Fourier. Dans la deuxième, le spectre est élargi avec des zéros. Et dans la troisième étape, on utilise la transformée de Fourier inverse pour revenir dans le domaine spatial.

Fonctions polynomiales

Il s'agit ici de méthodes d'interpolation obtenues à partir de fonction polynomiales de différents degrés. Chacune est une amélioration de la précédente prenant en compte diverses contraintes, notamment la symétrie, la normalisation, la continuité aux points de raccord, ou encore la préservation des échantillons originaux. L'interpolation se fait de manière générale, en réalisant une moyenne d'un ou de plusieurs voisins selon l'orientation des droites et de la position du pixel à interpoler.

Plus proche voisin

On la nomme ainsi car seul le point le plus proche intervient dans l'interpolation. Elle génère beaucoup d'erreurs notamment l'effet de pixellisation de l'image.

Linéaire

C'est une méthode d'interpolation simple qui consiste à prendre la moyenne de deux valeurs. Elle est rapide et aisée, mais manque de précision. Plus précisément, l'erreur est proportionnelle au carré de la distance entre les noeuds.

Bilinéaire

Ce type d'interpolation donne globalement de meilleurs résultats que l'interpolation linéaire, avec quelques calculs supplémentaires (Choix des deux points les plus proches de chaque côté du point étudié puis pondération par des coefficients inversement proportionnels à la distance et dont la somme vaut 1).

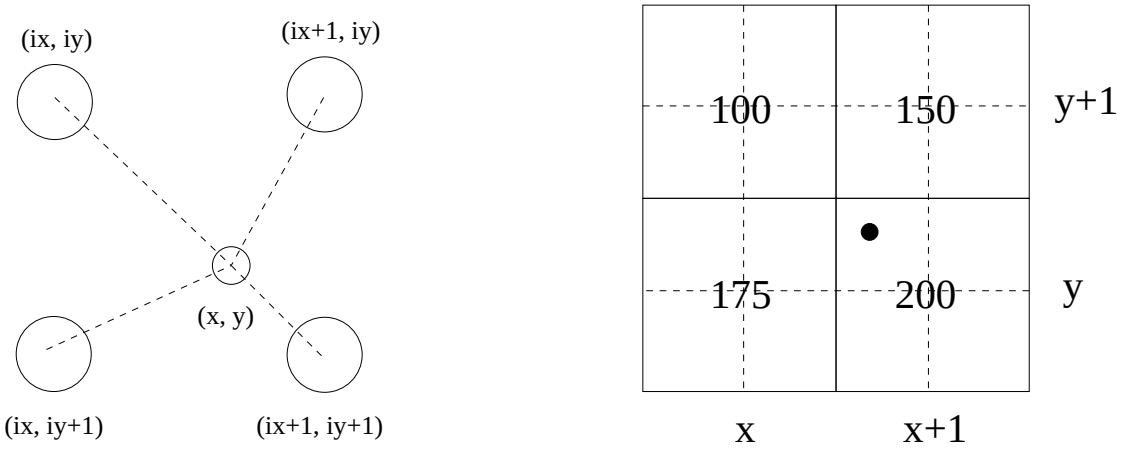

Fig. 1.2 – Mise en oeuvre de l'interpolation, Interpolation bilinéaire : 181, Interpolation au plus proche voisin : 200

On retrouve l' interpolation bilinéaire tout comme la précédente, assez souvent dans les applications en temps réel. Cependant, le rendu se traduit encore par un flou visuel assez important.

Quadratique

L'interpolation ici est effectuée entre les trois points les plus proches, en utilisant trois polynômes de degré deux. Par rapport aux méthodes précédentes, l'image interpolée paraît moins floue. Cependant, l'effet de pixellisation est encore visible.

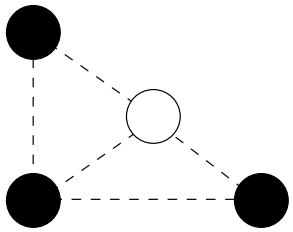

FIG. 1.3 – Interpolation quadratique

Cubique

L'interpolation cubique est le processus à travers lequel on trouve un point en utilisant une moyenne pesée de huit valeurs des points voisins.

Zoom fractal

Cette méthode est fondamentalement différente des précédentes puisqu'elle ne tient pas com pte des notions de contours et de régions, mais plutôt de la recherche des similarités structurelles de l'image et l'hypothèse de leur invariance par changement de résolution. L'ensemble de ces relations constitue le code fractal qui est une représentation approximative de cette image. L'utilisation du code fractal pour l'agrandissement d'images est récente, originale et s'appuie sur son indépendance vis-à-vis de la résolution de l'image initiale d'une part, mais d'autre part permet de respecter la nature discontinue des images naturelles et d'introduire des détails supplémentaires grâce à l'autosimilarité. Toutefois, un de ses principaux défauts est d'introduire de fausses discontinuités.

1.2.3 Filtres

Boite

Le filtre boîte doit son nom à sa fonction de transfert représentée par la formule :

Filtrer consiste à multiplier la fonction de transfert par le spectre que l'on désire filtrer. Dans le domaine spatial, il faut procéder à la convolution de la transformée de Fourier par la fonction de transfert, appelée réponse impulsionnelle h(t) du filtre. Cette dernière, pour le filtre boîte correspond à la fonction sinus cardinal sinc(x)=sin(pi*x)/pi*x. Cependant, elle a un inconvénient majeur, elle supprime de trop nombreux détails, d'où l'aspect flou de l'image.

Bartlett

Contrairement au filtre boîte, le filtre de BARTLETT conserve plus hautes fréquences et de ce fait plus de détails dans l'image, cependant certaines basses fréquences utiles sont supprimées. Il défini par une fonction de transfert :

B-spline

La transformée B-spline fournit une représentation continue d'un signal discret. Plus simplement, il s'agit d'une courbe lisse qui traverse deux points spécifiques ou plus. L'interpolation par les fonctions splines sont souvent préférées aux interpolations linéaires car le taux d'erreur y est moindre.1.4

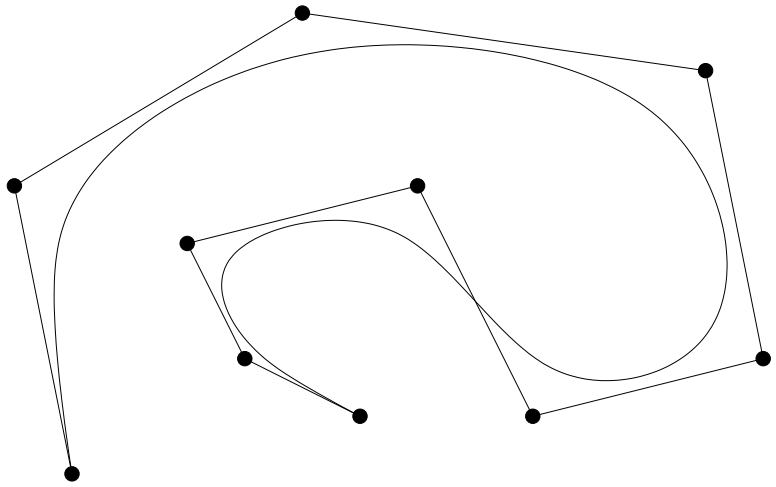

FIG. 1.4 – Approximation par une fonction spline de segments discrets.

La transformée B-spline possède de bonnes propriétés de lissage, et la modification d'un point du polygone n'entraine qu'une modification locale de la courbe obtenue.

Approximation par une fonction spline de segments discrets.

Median

Une interpolation médiane est une interpolation non linéaire produisant comme résultat le médian des valeurs traitées. L'idée est simple : prenons un ensemble de pixels au voisinage d'un pixel donné et rangeons les niveaux de gris de cet ensemble de pixels en ordre croissant, puis choisissons le niveau de gris qui arrive en position médiane. Ce niveau de gris sera celui du pixel donné.

Exemple : Si la liste des niveaux de gris d'un pixel est : [64, 64, 64, 64, 255, 255, 64, 64, 255].La liste dans l'ordre croissant est alors : [64, 64, 64, 64, 64, 64, 255, 255, 255] et la valeur du pixel devient donc la 5 ième valeur de la liste soit 64.

L'avantage du filtre médian réside dans la simplicité de sa mise en oeuvre et dans le temps de calcul relativement faible. Il préserve mieux la netteté que les filtres linéaires,

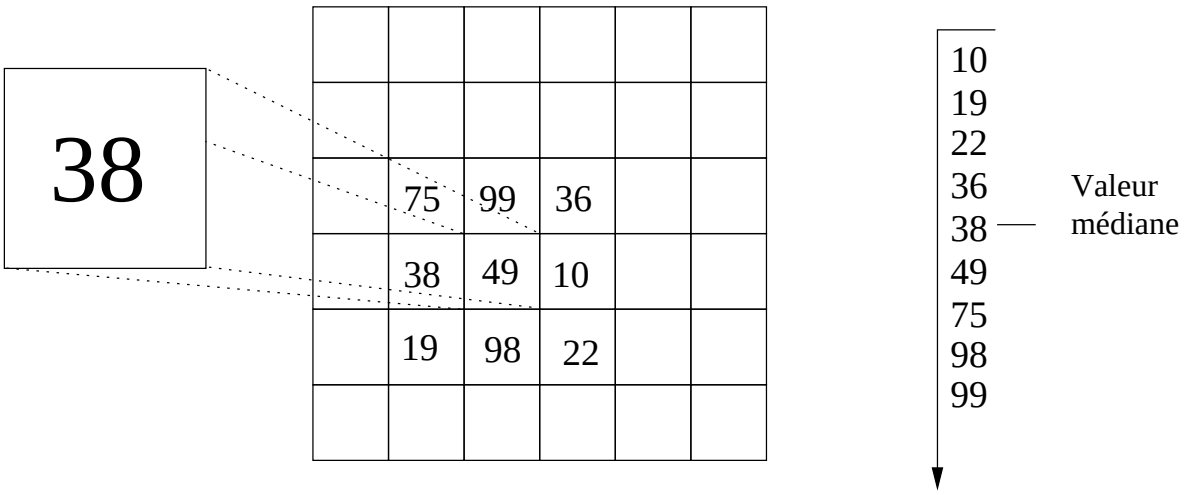

FIG. 1.5 – Filtre median

mais a tendance à déformer les structures présentes dans l'image puisque celles-ci ne sont pas analysées.

Wiener

L'utilisation d'un filtre de Wiener nécessite un apprentissage préalable pour évaluer ses poids. Celui-ci consiste, à partir d'une image originale et de sa version réduite, à optimiser les poids du filtre d'interpolation afi n de minimiser l'erreur entre l'image originale et l'image interpolée d'après la version réduite. Il s'agit en fait d'un problème classique de minimisation par les moindres carrés. La qualité de l'interpolation avec un filtre de Wiener va dépendre du couple d'images utilisé pour calculer ses poids. En général, il est préférable de prendre comme couple d'images l'image originale que l'on veut agrandir et sa version ré duite. L'interpolation avec le filtre de Wiener donne un résultat correct lorsque l'image à interpoler a des caractéristiques très simples. Ce type de filtre produit des résultats sensiblement meilleurs que le filtre médian. Cependant, en raison de la phase d'apprentissage, il nécessite beaucoup plus de calculs que ce dernier.

Mitchell

Le client a suggéré l'utilisation de ce fi ltre car il est un bon compromis entre l'effet de lissage d'un B-spline et l'effet d'accentuation obtenu par un filtre boîte. D'autre part, les autres filtres ont un coût de calcul équivalent et celui-ci convient particulièrement à nos attentes.

1.2.4 Détection de contours

Références : [3, 5, 1]

Nous ne sommes pas directement concernés par les méthodes de détection de contour à proprement parler. Nous allons utiliser comme données d'entrée une image à étiquettes, ce qui signifie que ses régions seront déjà définies et identifiables. Le travail consistera

alors à suivre les bords de ces régions afin de construire leur contour. Nous n'aurons donc pas à rechercher les régions en utilisant un des algorithmes cités précédemment, malgré cela il est intéressant de connaître leur existence dans l'éventualité d'une modification des données d'entrée. La détection des contours associée à une segementation de l'image représente l'extension la plus intéressante qui pourrait être apportée à notre programme. Ainsi la contrainte de l'étiquetage manuel des régions ne serait plus.

Le but de la détection de contours est de repérer dans une image, les changements brusques de couleur et ainsi pouvoir la séparer en régions. Un contour peut être la limite entre deux zones de couleur différente, ou encore une ligne traversant une zone de couleur uniforme. Dans ce cas, le contour se situe de chaque côté de la ligne.

L'approche la plus simple, pour la détection s'effectue en parcourant les lignes de pixels horizontales et verticales séparément. On lit donc les valeurs des pixels se suivant sur chaque ligne en surveillant les changements importants. On a alors deux images séparées, l'une représentant les contours horizontaux et l'autre les contours verticaux. En superposant les deux, on obtient une image contenant tous les contours.

Présentons maintenant les différentes méthodes de détection de contours existantes : Un des premiers algorithmes de détection fut celui appelé Roberts' Cross. Il est assez simple et utilise deux noyaux de convolution 2x2. Cette approche est toujours utilisée car sa faible complexité rend l'exécution très rapide, cependant elle est très limitée car trop sensible au bruit sur les images, cela étant dû à la trop petite taille des noyaux.

1	0
0	-1

Gx

0	1
-1	0

Gy

L'algorithme de détection des contours de Nalwa, fonctionne par remplissage de surface. Il n'utilise pas le calcul de la dérivée, mais des fonctions tangentes hyperboliques et quadratiques sont utilisées pour correspondre à l'intensité de l'image dans un noyau de 5x5 pixels. La fonction avec la plus faible erreur est retenue. Un seuillage est utilisé pour limiter l'apparition de contours inutiles.

L'algorithme de détection d'Iverson tente d'améliorer les performances des algorithmes linéaires, en ajoutant des vérifications logiques de l'existence de contours et ce dans l'optique de diminuer le nombre de "faux" contours détectés.

L'algorithme de Bergholm a pour but encore une fois de repérer uniquement les contours significatifs, ceux-ci sont représentés sur l'espace d'échelle (convolution avec un noyau de Gauss). La technique consiste à analyser l'image en basse résolution puis projeter les contours détectés sur l'image haute résolution afin de les localiser précisément. Pour cela l'image analysée est d'abord modifiée à l'aide d'un filtre de "blur".

L'algorithme de Rothwell est assez similaire à celui de Canny, décrit ci-dessous mais n'utilise pas de méthode de seuillage à hystérésis. En effet le seuillage est ici dynamique et a une seule valeur.

Le détecteur par passage par zero (zero-crossing) est basé sur un filtre Log (Laplacian of Gaussian). Le Laplacien d'une image fait apparaître les changements d'intensité rapides. Il est calculé en utilisant trois noyaux de convolution 3x3 :

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

-1	2	-1
2	-4	2
-1	2	-1

Une fois calculé, on note quand il croise l'axe de abscisses car cela signifie qu'un contour a été détecté (principe de la seconde dérivée).

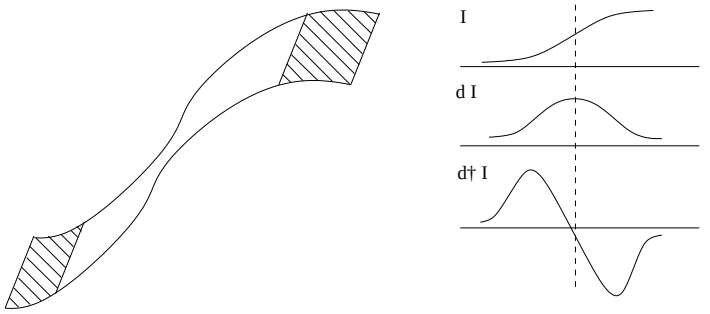

Image : `http://homepages.inf.ed.ac.uk/rbf/HIPR2/featops.htm`

Finalement, l'algorithme le plus utilisé est celui de Canny. Il se divise en plusieurs étapes, tout d'abord un filtre gaussien d'adoucissement est utilisé pour réduire le bruit présent sur l'image originale. Ensuite un gradient retournant l'intensité des contours est appliqué, en utilisant l'opérateur de Sobel, constitué de deux noyaux de convolution 3x3 (un horizontal, un vertical).

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Puis la direction des contours est calculée, on obtient donc pour chaque pixel, un gradient d'intensité et la direction du contour. La détection des contours se fait en utilisant une méthode de seuillage à hystérésis, un maximum et un minimum sont spécifiés, on utilise la valeur maximale pour détecter les contours nets, puis ils sont tracés tant que la valeur minimale n'est pas atteinte (cela permet de dessiner un contour qui s'estompe progressivement).

1.2.5 Lissage des contours

Référence principale : [8]

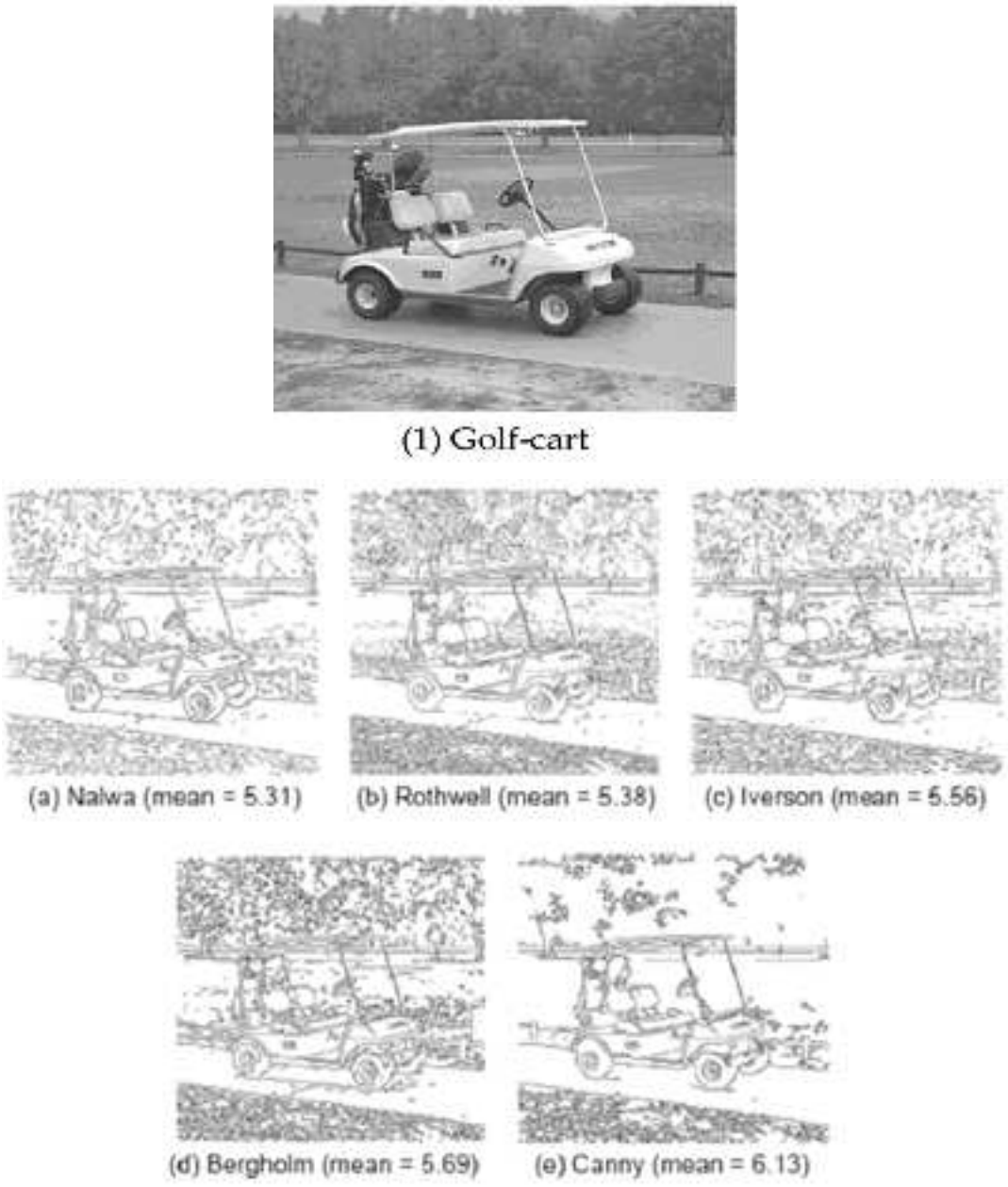

FIG. 1.6 – Exemple de détection des contours

Afin d'atténuer les effets de crénelage dûs au suréchantillonage de données réelles présents sur le contour discret d'un objet, le lissage est un traitement qui, à partir d'une liste de points (le contour discret) fournit une représentation continue de l'objet. Dans le but d'obtenir une meilleure approximation possible de la frontière réelle, ce nouveau contour devra satisfaire les deux caractéristiques suivantes :

- La construction de ce contour doit être indépendante du point de départ et du sens de parcours.
- L'algorithme doit être sans perte d'informations c'est-à-dire réversible, à partir du nouveau contour, on doit pouvoir récupérer le contour initial en le rediscrétisant.

Il existe plusieurs méthodes permettant ce lissage.

La vectorisation

C'est un procédé qui recherche à partir du contour discret de l'objet traité une représentation polygonale (suite de segments de droite). Ce nombre de segments doit être minimal tout en gardant une bonne approximation des caractéristiques de la frontière.

Méthode de la corde L'algorithme, proposé par V. Ramer consiste à substituer l'ensemble des points par un ensemble de segments garantissant la distance minimale entre ceux-ci et les points. "On approxime l'ensemble des points par un segment. Si la distance entre le segment et le point le plus éloigné est supérieur à un seuil donné, ce point devient un point de rupture. Le processus recommence en considérant les deux ensembles de points ainsi constitués".

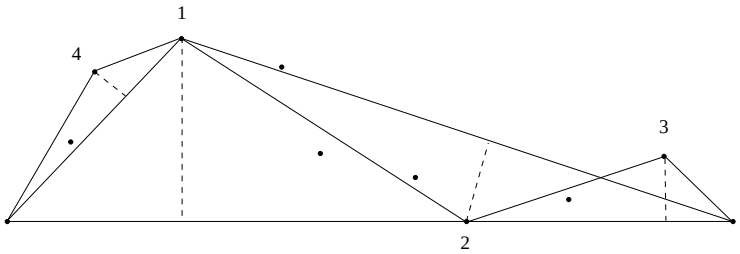

FIG. 1.7 – Méthode de la corde

Image :
`http://depinfo.u-bourgogne.fr/Maitrise-info/trait_images/analyse1.pdf`

Vectorisation basée sur la reconnaissance de droites discrètes L'algorithme de Debled consiste à reconnaître les droites discrète[1]s. A partir d'un point initial, tant que le segment est un segment de droite discrète on parcours le chemin en rajoutant le point visité. Lorsque ce point ne permet plus d'étendre le segment. On recommence le processus à partir du dernier point. La vectorisation sera donc basé sur les extrémités de tous les segments de droite trouvés.

[1]Une droite discrète L et l'ensemble despoints (x, y) de Z qui satisfait la double inégalité suivante : $\mu <= Ax - By < \mu + wavecA, B, \mu \in Z, w \in N$.

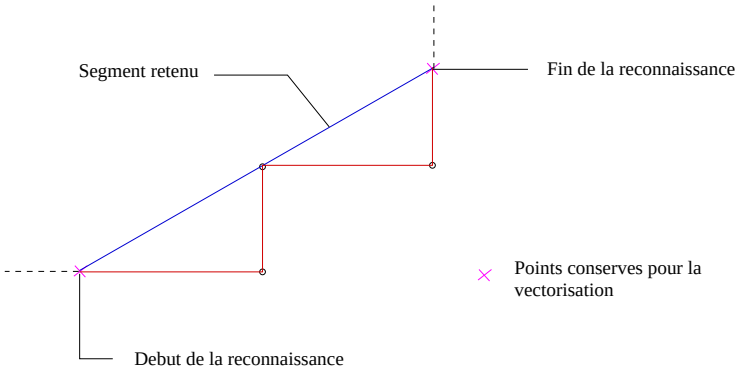

FIG. 1.8 – Reconnaissance de droite discrete

Inconvénients Mais de telles méthodes peuvent faire apparatre des points anguleux :

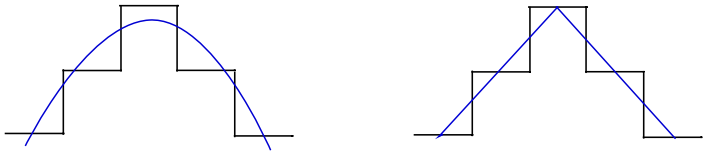

FIG. 1.9 – Point anguleux

Aussi, elles peuvent dépendre du pixel de départ pour effectuer la polygonalisation.

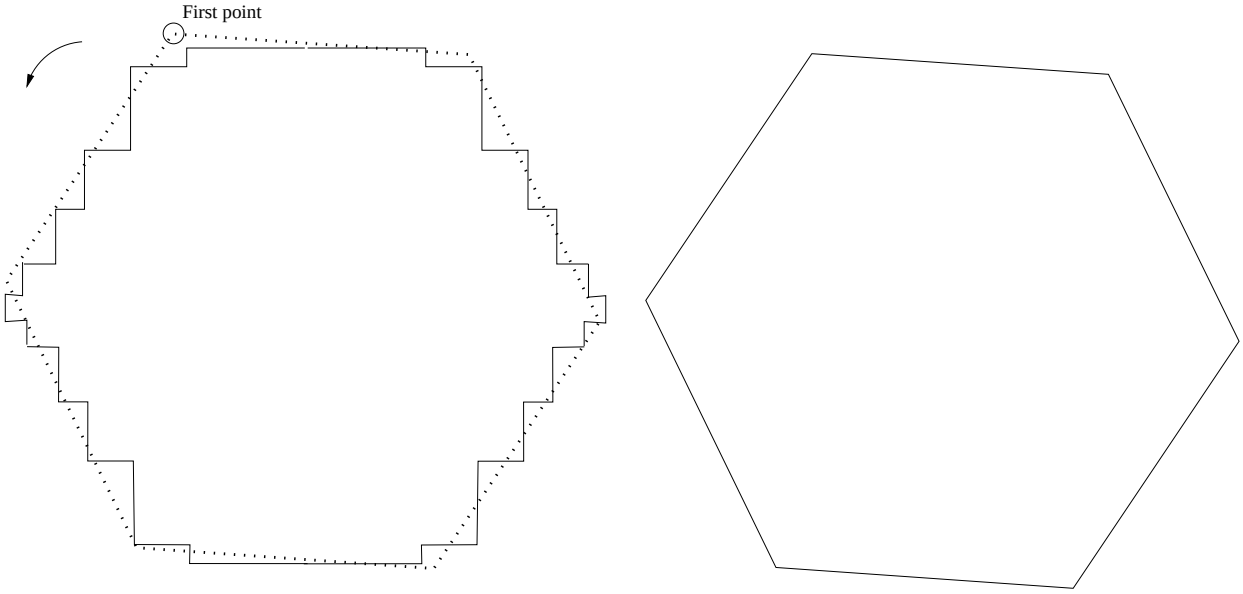

FIG. 1.10 – Importance du point de départ

Il est clair dans cet exemple, que changer de point de départ modifiera l'orientation de ce polygone.

Perte d'informations : si on ne retient qu'un sous-ensemble de points pour la vectorisation, on ne pourra pas retrouver le chemin discret initial.

Les courbes polynomiales

Pour éviter les problèmes de points anguleux et de dépendance du point de départ liés à l'iterpolation, on introduit les courbes polynomiales.

Interpolation polynomiale A partir d'un ensemble de points, l'interpolation polynomiale (linéraire, bilinéaire, Lagrange ?) fournit un polynôme qui passe exactement par chacun de ces points. Ici, on se contentera de passer au voisinage de ces points, car nous cherchons plutot à lisser une série de points et non pas à trouver quel polynôme représente le mieux ces points.

Splines Une fonction spline est une fonction continue définie par morceaux par des polynômes.

- L'approximation
  Les splines d'approximation peuvent passer par certains points de contrôle, mais en général, ne passent pas par tous les points. Cette technique pose donc un choix des points a traiter. Ce qui lisse encore mieux, mais qui implique inévitablement une perte de localisation. Le lissage devant être réversible, c'est-à-dire que l'on doit pouvoir retrouver tous les points initiaux, cette technique ne nous concernera pas.
- L'interpolation
  L'interpolation, quant à elle traite tous les points que l'on appelle points de contrôle. On ne demande pas à la nouvelle représentation du contour de passer exactement par le point, elle peut passer dans un voisinage du point.

Courbes de Bézier Les splines sont une généralisation des courbes de Bézier qui sont des courbes polynomiales paramétriques. La courbe est à l'intérieur de l'enveloppe convexe des points de contrôle. La courbe commence par le point P0 et se termine par le point PN mais ne passe pas a priori par les autres points de contrôle qui déterminent cependant l'allure générale de la courbe. Les courbes de Bézier cubique (de degré 3 et définies par ses points de contrôle P0, P1, P2, P3) sont les plus utilisées.
Sa forme paramétrique est :

$$B(t) = P_0(1-t) + 3P_1t(1-t) + 3P_2t(1-t) + P_3t, t \in [0,1]$$

B-splines B-Spline désigne une courbe continue polynomiale par morceaux. Une courbe B-Spline est constituée d'une chaîne continue de courbes de Bézier du même degré 3 connectés entre elles. Chaque courbe est tangente aux autres en ses points de début et fin.
Un exemple de B-Spline où les Pi sont les points de contrôles :

Ceci dit, l'ensemble des points de contrôle étant important, on est confronté à de fortes oscillations qui pourraient être evitées en augmentant fortement le degré des polynômes. Et cela devient assez coûteux.

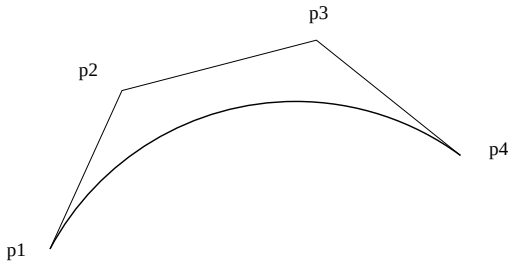

FIG. 1.11 – Courbe de Bézier

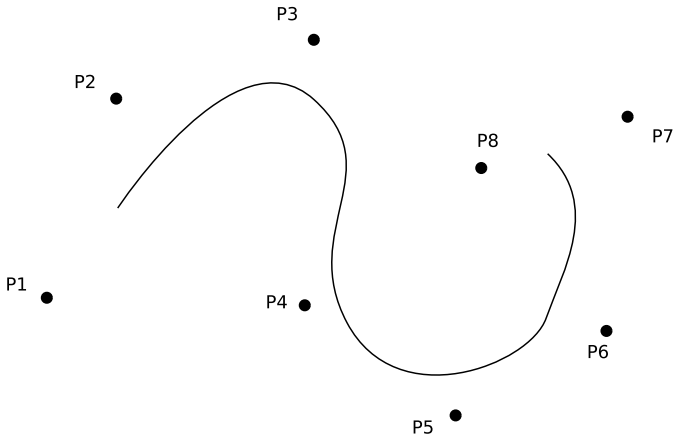

FIG. 1.12 – Courbe spline

Les chemins euclidiens

Le problème de réversibilité nous impose que le nouveau point du contour interpolé ne doit pas dépasser un demi pixel de façon à retrouver le contour discret initial. On a autant de points initiaux que de points euclidiens.

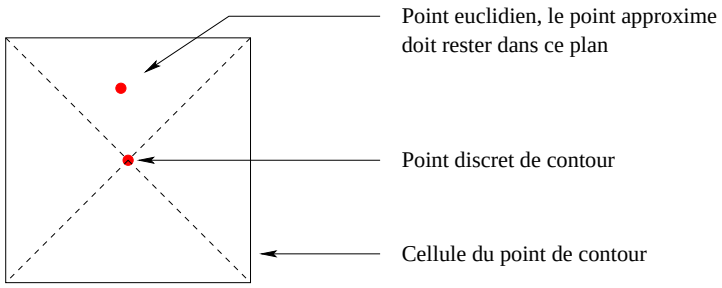

FIG. 1.13 – Point euclidien

Il existe plusieurs méthodes permettant de générer un chemin euclidien, toutes assurant la réversibilité. Nous utilisons les chemins euclidiens dirigé par la tangente,cette méthode imposée par notre client est de toute façon la plus appropriée car elle ne dépend ni du point de départ, ni du sens de parcours du contour et reste peu coûteuse. Aussi l'utilisateur n'a pas besoin de spécifier la longueur du voisinage, celle-ci dépendant uniquement du contour.

Chemin euclidien dirigé par la tangente Longueur du voisinage : déterminé par le motif du contour.
Pour chaque point discret Pi du contour, on cherche d'abord le segment de droite discrète. Le plus long centré en Pi. Le calcul de la tangente discrète déduite ne dépend alors que de lui-même et de son voisinage. A partir de cette tangente discrète on peut approximer une tangente réelle qui nous donnera alors le point euclidien. La construction du chemin euclidien sera donc bien indépendante du point de départ et du sens de parcours. Le nouveau contour étant constitué de points euclidiens, on est assuré que l'on peut retrouver le contour discret initial. Les deux caractéristiques présentées au début sont donc bien respectées.

Bilan

Un contour continu tel que les courbes splines augmente le risque d'oscillations, ou bien devient très coûteux alors que la polygonalisation fait apparaître des points anguleux et peut dépendre du point de départ. Dans ce projet, nous resterons donc sur une représentation de type polygonal, mais en approximant chaque point au mieux de la frontière réelle de l'objet en déplaçant dans le plan euclidien les points frontière discret et dirigé selon la tangente locale.

1.3 Le Projet

Il existe deux grandes catégories d'images. Les images vectorielles sont constituées d'entités, de formes géométriques qui sont représentées par des formules mathématiques, et qui ne posent donc aucun problème dans tout redimensionnement tel que agrandissement, rétrécissement, ou encore élargissement. L'autre type d'image est l'image dite bitmap. Les informations sont stockées dans une matrice qui associe à chacun des pixels sa couleur RGB. Ici, on considérera ce deuxième type d'image, non compressé et sans perte d'informations comme l'est par exemple JPEG. On se restreindra aux formats ppm et png.

Dans ce projet, on etudiera des images simples formées de une ou plusieurs formes sur un fond initial. Notre travail sera simplifié par l'utilisation d'images avec etiquette (c'est-à-dire que chaque pixel de l'image sera etiquetté de façon à savoir à quel objet il appartient.) qui permettra ainsi le suivi des contours de manière relativement aisée.

L'agrandissement d'une image de largeur L et de hauteur H en une image plus grande de nouvelle largeur L' et de nouvelle hauteur H' impose de devoir calculer les couleurs des pixels manquants. Cela se fera au moyen d'un filtre de reconstruction qui nous permettra alors d'interpoler ces données. Un bon filtre semble être celui de Mitchell.

Mais un autre problème survient lors du grossissement du contour, sur le bord de la région etudiée apparaît un important effet de marche d'escalier, si aucun pré-traitement n'a été effectué.

Dans ce projet, on se donne pour but de lisser au mieux la frontière réelle de l'objet avant agrandissement de celui-ci de facon à minimiser l'erreur dû à la pixellisation. Il existe

FIG. 1.14 – Contour discret et euclidien sur l'oeil de Lenna

un algorithme de complexité linéaire permettant de lisser géométriquement ce chemin par de petits déplacements de chacun de ces points : le chemin euclidien. Cette technique nous garanti une meilleure approximation de la frontière réelle que les méthodes existantes tels que la polygonalisation ou bien les courbes splines. Ainsi, à la discrétisation du nouveau contour agrandi lissé, on aura conservé des informations plus précises.

Pour réincruster l'objet, on utilisera la technique d'antialiasing sur les bords de l'objet. Cela consiste à faire apparaître les pixels du contour d'une couleur de teinte intermédiaire selon un certain coefficient entre celle de l'objet traité et celle de sa région voisine. Le chemin euclidien du contour permet de calculer ce coefficient alpha. Chaque pixel est en effet doté de sa couleur RGB mais aussi d'un canal alpha qui sert à ajouter divers niveaux de transparence à la couleur.

Le traitement de l'image se déroulera de la manière suivante :
- Suivi du contour de l'objet (facilité ici par l'étiquette de l'image)
- Construction du contour lissé puis agrandissement de ce contour
- Discrétisation du nouveau contour lissé agrandi
- Reconstruction du nouvel objet agrandi avec son nouveau contour
- Incrustation de l'objet sur un fond, ou un autre objet.

Enfin l'algorithme utilisé (chemins euclidiens) est relativement récent et donc encore peu utilisé. La thèse d'Anne Vialard est l'unique référence que nous avons sur le sujet.

Les figures 1.15, page 22 et 1.16, page 23 illustrent le fonctionnement du projet.

1.3.1 Les différents formats de fichiers image

source : [2, 4]

On considère ici uniquement les formats en mode point (bitmap) et non les formats vectoriels, qui ne nous intéressent pas dans le cadre de ce projet.

Gif

Graphics Interchange Format.
Le format gif n'est plus sous brevet depuis peu, il permet de coder 256 couleurs à l'aide d'une palette. Il propose un algorithme de compression sans perte, ainsi que le support de la transparence et des images animées.

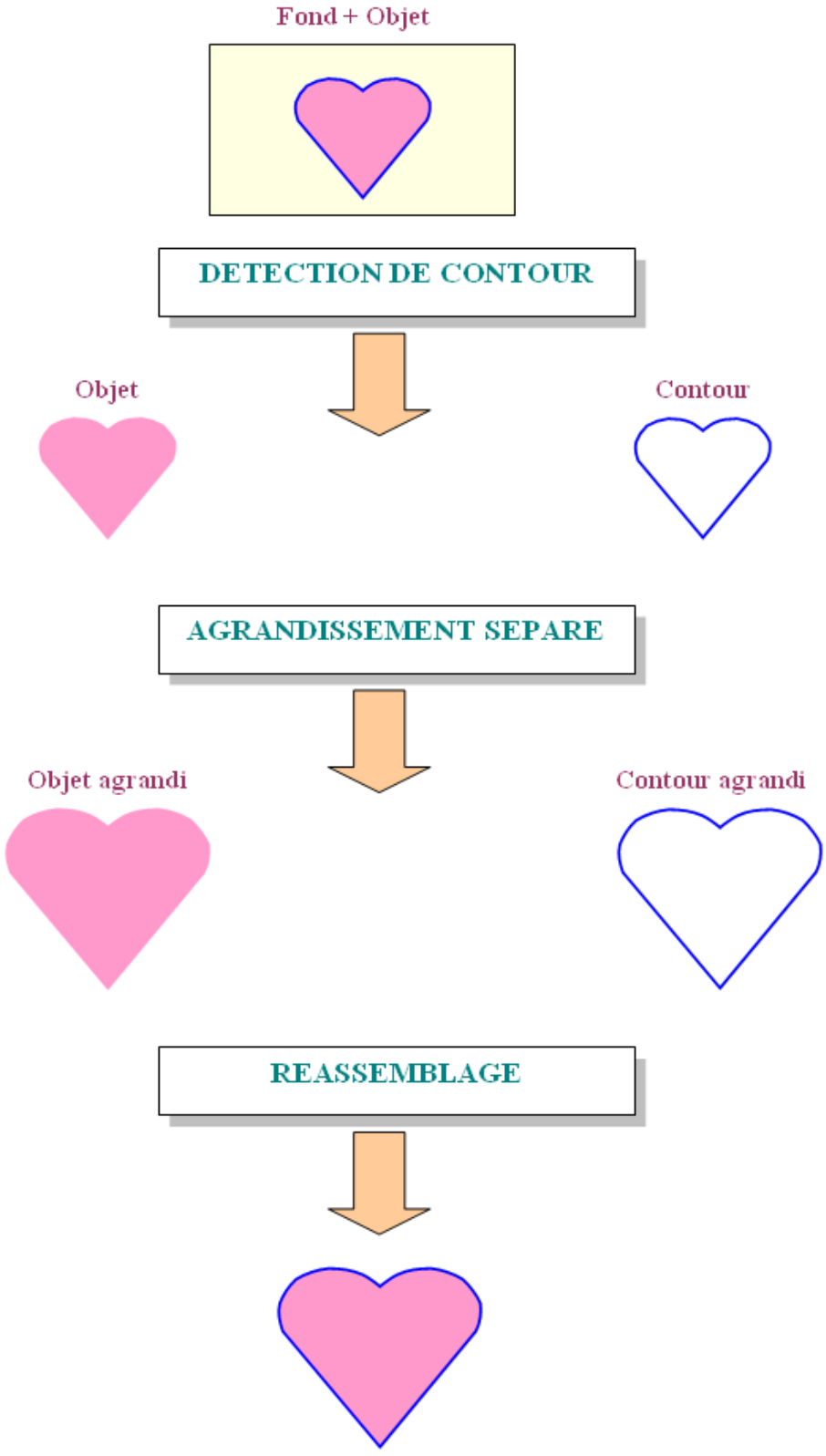

FIG. 1.15 – Diagramme de fonctionnement 1

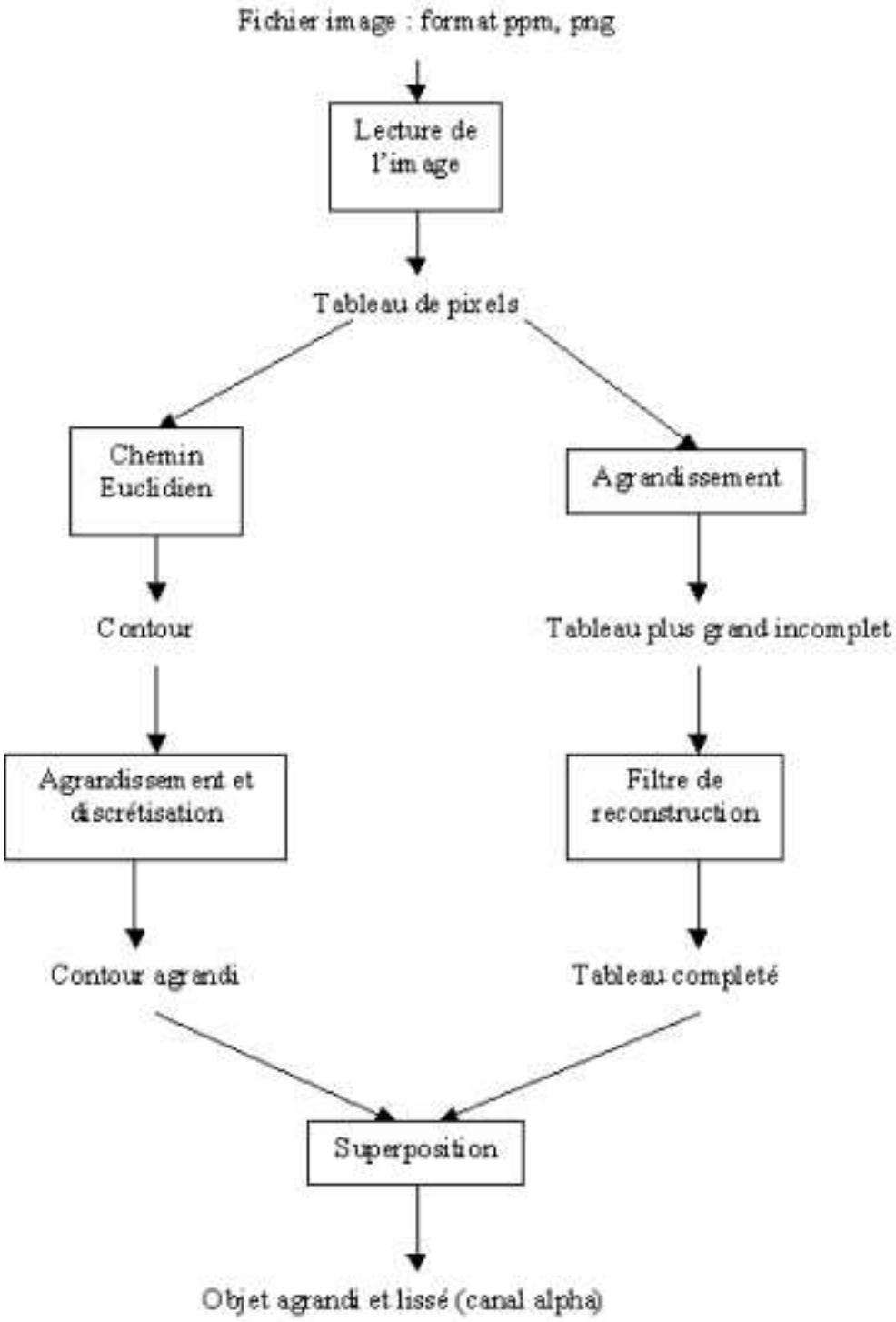

FIG. 1.16 – Diagramme de fonctionnement 2

256 n'étant pas un nombre de couleur suffisant pour notre travail, nous avons écarté ce format.

Jpeg

Joint Photography Experts Group.
Ce format est libre de droits et permet de représenter 16 millions de couleurs. Malheureusement la compression associée implique des pertes d'informations même au taux le plus faible, c'est pourquoi nous n'avons pas travaillé avec ce format d'image.

Png

Portable Network Graphics.
Png propose une compression sans perte, la représentation de 16 millions de couleur ainsi que le support de la transparence. De plus ce format est libre et s'avère donc être un bon choix pour nos travaux.

Tiff

Tagged Image File Format.
Utilisé pour l'archivage d'images de qualité. Utilise un algorithme de compression sans perte. Fichiers de relativement grandes dimensions.

Pnm

Portable Any Map.
Pnm regroupe en réalité 3 formats : ppm, pgm et pbm, qui sont respectivement utilisés pour les images en couleurs, en niveaux de gris et en monochrome. Ce format peut être codé en Ascii ou en Brut, cette deuxième solution est assez simple d'utilisation car un fichier peut alors être tout simplement ouvert avec un éditeur de texte afin de voir les valeurs des composantes de chaque pixels. Ce format est sans compression et ne supporte pas la transparence. Cependant sa simplicité d'utilisation en fait un bon choix pour débuter notre projet.

Jpeg2000

Le format jpeg2000 a été créé pour palier aux défauts du format jpeg. Il permet une compression sans perte des données, mais ne supporte pas la transparence.

1.4 L'importance du découpage par région

Comme cela a été précisé précédemment, nous savons combler les pixels manquant lors de l'agrandissement d'une image. Ceci est fait grâce à un filtre de reconstruction qui se sert des pixels voisins pour donner une valeur au nouveau pixel. Il y a pour cela différentes méthodes existantes.

L'approche la plus simple consiste à réutiliser les valeurs de pixels déjà existants et de les reproduire pour les pixels manquants. Le résultat est peu convaincant et on obtient une image très pixellisée, avec des contours très aliasés, mais dont la netteté est préservée.

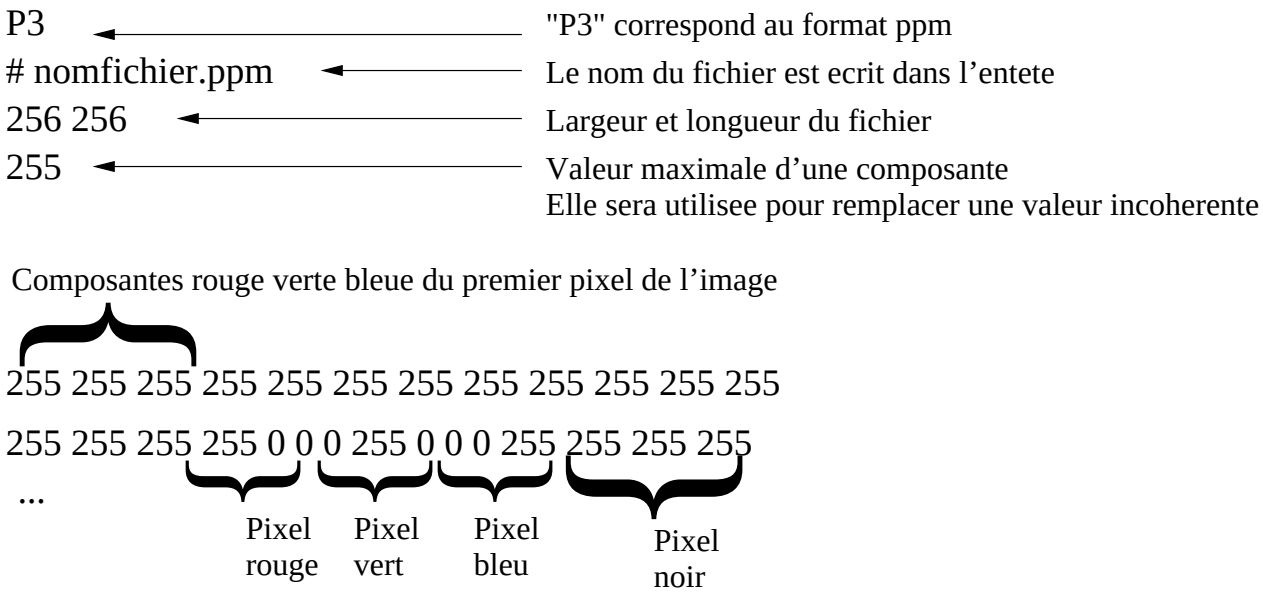

FIG. 1.17 – Structure d'un fichier ppm

On peut aussi se servir des pixels voisins et en faire une moyenne pour trouver la valeur du nouveau pixel. Cela a un inconvénient qui est de donner un aspect flou à l'image, ce qui est tout de même moins gêenant qu'un aspect pixellisé. Cet effet de flou vient du fait que près d'un contour, le filtre peut baser son calcul de la valeur du nouveau pixel sur des voisins qui appartiennent à une autre région. Cela a pour effet de donner des valeurs qui sont des moyennes entre deux régions et donc de ne pas préserver la netteté.

Nous souhaitons donc palier ces défauts, c'est à dire reconstruire une zone sans que celle-ci se pixellise mais aussi préserver des contours bien nets. Or nous savons parfaitement reconstruire une zone unie ou dégradée, sans changement brusque de couleur. Nous pouvons même appliquer par la suite un filtre de lissage sur son contour pour un résultat encore amélioré.

L'idée est donc de ne pas appliquer le traitement sur l'image en sa globalité, mais de la découper en autant de régions (objets) relativement uniformes que nécessaire, et traiter ces régions une par une avant de les réassembler.

Cependant pour débuter, notre approche consistera à agrandir uniquement un objet de l'image.

1.5 Besoins non-fonctionnels

Complexité

Nous devons veiller à préserver la linéarité des différents algorithmes que nous employons, et aussi que leur assemblage dans l'application reste linéaire. Ceci est nécessaire à la rapidité du traitement des images.

FIG. 1.18 – Image de test

Robustesse

L'application doit prendre en charge les entrées inattendues ou incorrectes et renvoyer des erreurs correctement gérées à l'utilisateur. Une partie de ces erreur concernant les entrées/sorties avec le format ppm est gérées par un module d'exceptions fourni par notre client.

Fiabilité

L'application doit permettre un redimensionnement correct, conservant l'aspect original de l'image. Pour ce faire, les algorithmes les plus fiables en terme de qualité (filtre de Mitchell, chemins Euclidiens) sont utilisés afin d'obtenir un résultat de grande précision.

Réutilisabilité

Le code doit être modulaire, et donc réutilisable et améliorable. Faciliter l'ajout de nouveaux formats d'image ou l'implémentation de nouveaux algorithmes concernant le suivi de contours, le redimensionnement ou encore le lissage.

1.6 Besoins fonctionnels

Formats supportés

Nous avons choisi d'opter pour les formats ppm tout d'abord pour sa simplicité d'utilisation, puis png pour son support de la transparence. Ces deux formats sont non-compressés et libres de droits.

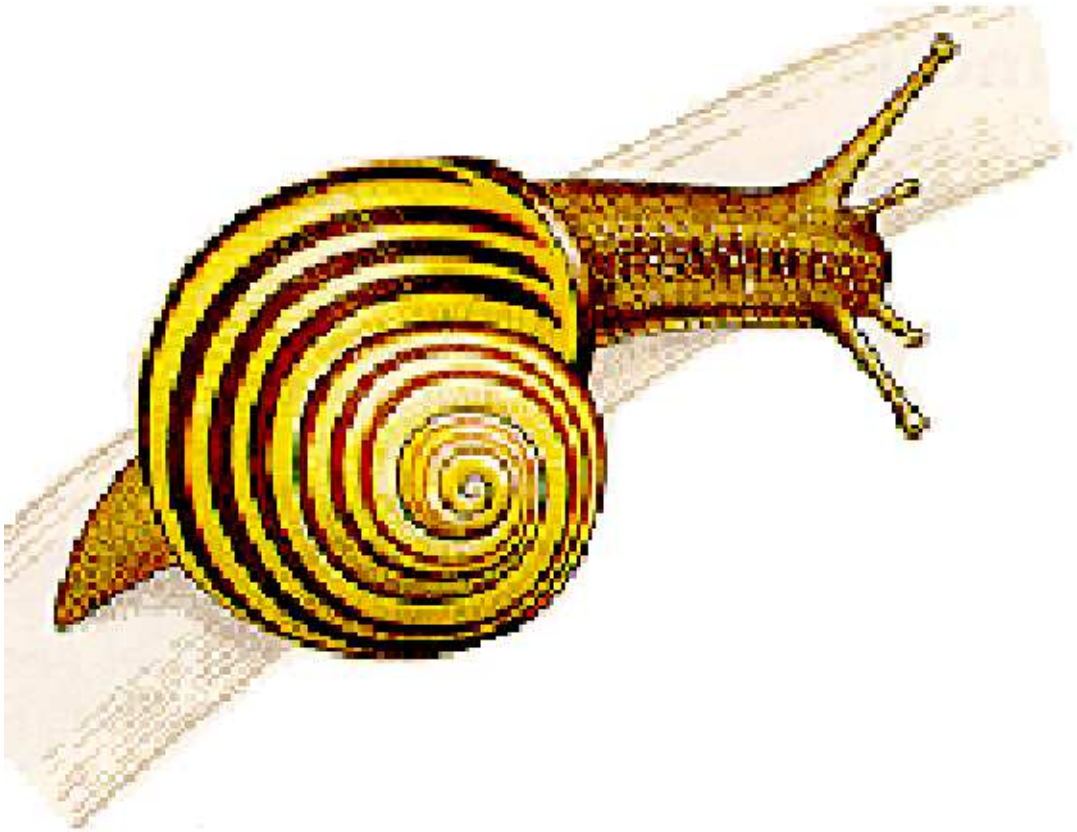

FIG. 1.19 – Agrandissement x2 au plus proche voisin - Pixellisation

FIG. 1.20 – Agrandissement x2 et filtre de Gauss - Effet de flou

Entrées/sorties

L'application peut charger des images au formats ppm et png en mémoire, avoir accès au différentes données concernant l'image telles que la largeur, la hauteur, la représentation de couleurs et bien sûr les valeurs correspondant à chaque pixel (composantes, transparence...).
Aussi il est possible d'écrire un fichier image à partir des données en mémoire.

Extraction de l'objet

L'application sait opérer un suivi de contour selon une ou plusieurs étiquettes, situées dans un fichier étiquette au format ppm ou png. De cette exctraction de contours, on obtient un code de Freeman et un point de départ pour chaque étiquette. Cela permet alors d'isoler le contenu de l'objet afin de l'agrandir, et aussi de créer un chemin euclidien afin d'agrandir le contour discret obtenu.

Agrandissement du contenu de l'objet

L'application agrandit le contenu de l'objet à l'aide du filtre passé en paramètre (Mitchell si aucun n'est spécifié). L'objet est interpolé de quelques pixels sur les bords afin d'éviter l'utilisation de mauvaises donnéees (hors objet) par le filtre. Le facteur d'agrandissement est basé sur un nombre réel qui doit être supérieur à 1.

Création et agrandissement du chemin euclidien

Le chemin euclidien est correctement calculé à partir de contour discret extrait auparavant. Il est ensuite possible de l'agrandir en fonction de facteur d'agrandissement spécifié.

Recomposition et lissage

On peut enfin choisir de recoller l'objet sur un fond dont on spécifie la couleur, et ce afin d'effectuer un lissage des bords de l'objet et éliminer tout effet d'aliasing.

Chapitre 2

Exemples de fonctionnement

Pour plus de détails sur l'utilisation de l'application, se référer au manuel d'utilisation en annexe.

2.1 Ligne de commande

Notre application étant destiné à un usage très spécifique, son exécution est très rapide et ne comporte que peu d'options. C'est pourquoi l'implémentation d'une interface graphique n'était pas indispensable.

De plus notre client nous a suggéré une application en ligne de commande, et nous a précisé que l'utilisation d'une interface graphique ne lui était pas utile. Nous avons donc avant tout développé notre application dans cette optique, en proposant quelques options décrites ci-après.

2.2 Les options

Notre application comporte certains paramètres qui ont une influence significative sur le résultat obtenu. Tout d'abord le facteur de redimensionnement de l'image qui est bien sûr le plus important, mais aussi le filtre de reconstruction employé.

Le facteur d'agrandissement (notre application n'étant pas conçue pour le rétrécissement d'image) est tout simplement un réel qui n'a théoriquement pas de véritable limite dans le cadre d'une utilisation habituelle, l'agrandissement d'une image par un facteur 20 étant déjà énorme.

Le choix du filtre de reconstruction employé lors de l'agrandissement de l'image est aussi laissé à l'utilisateur, bien que le filtre de Mitchell offre généralement le meilleur résultat, l'utilisation d'autres filtres comme Bell, Bspline, Lanczos ou Bartlett peut être justifiée dans des cas spécifiques ou encore à titre expérimental.

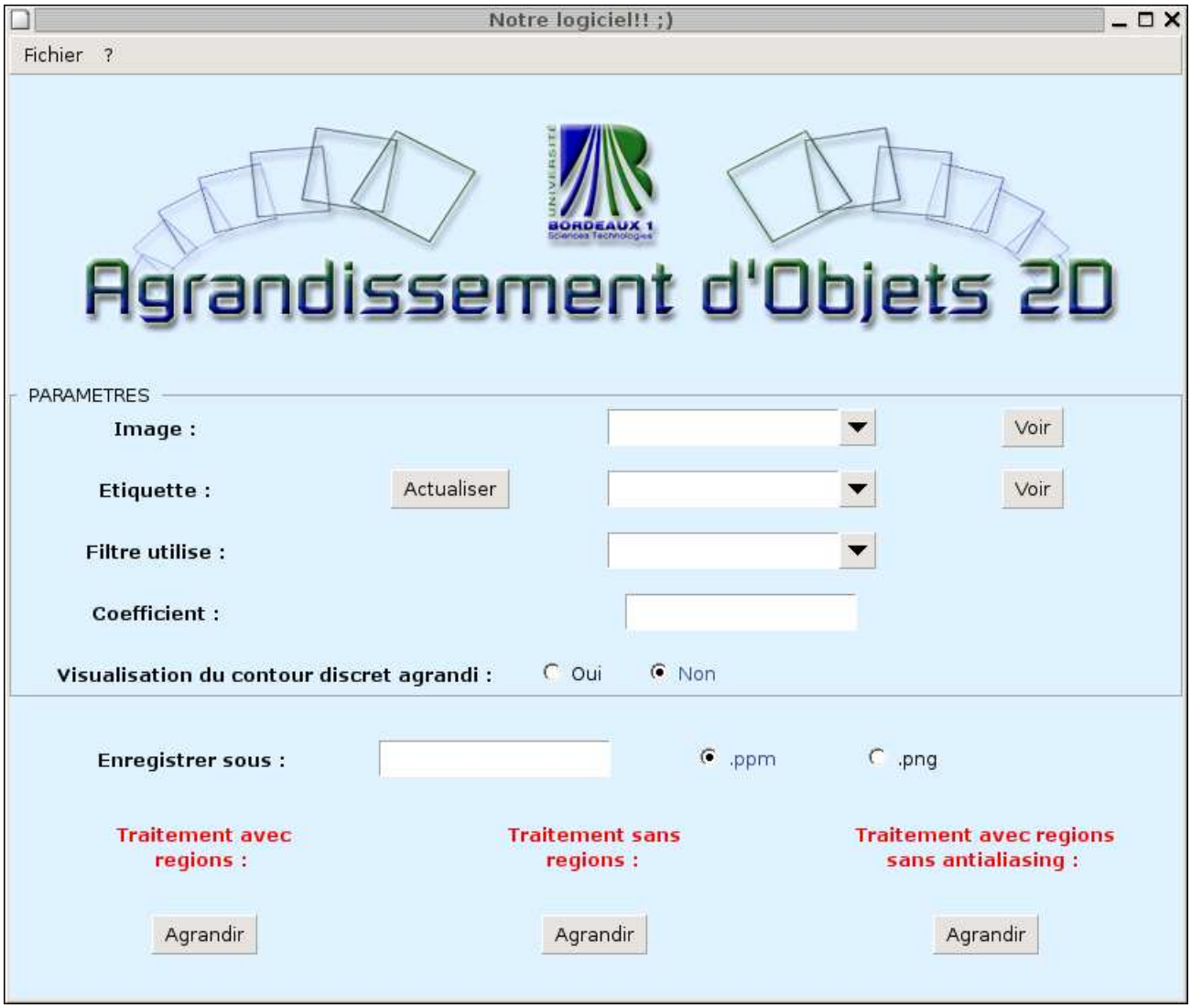

FIG. 2.1 – Interface

2.3 Interface graphique

Une fois arrivés à un stade avancé du développement de l'application, nous avons choisi de concevoir une interface graphique légère et simple d'utilisation, et ce afin de pouvoir directement visualiser le résultat du traitement de l'image, mais aussi comparer avec un traitement plus classique ou encore visualiser le contour discret.

L'interface permet à l'utilisateur de choisir une image, une étiquette et de les visualiser. Ensuite il est possible de spécifier le facteur d'agrandissement et de procéder au traitement de l'image. L'opportunité de visionner clairement le contour discret est aussi offerte. Enfin le résultat obtenu par notre application et le résultat issu d'un traitement "basique" sont affichés pour comparaison.

2.4 Fonctionnement

Exemple de ligne de commande pour lancer l'application figure exfonc.

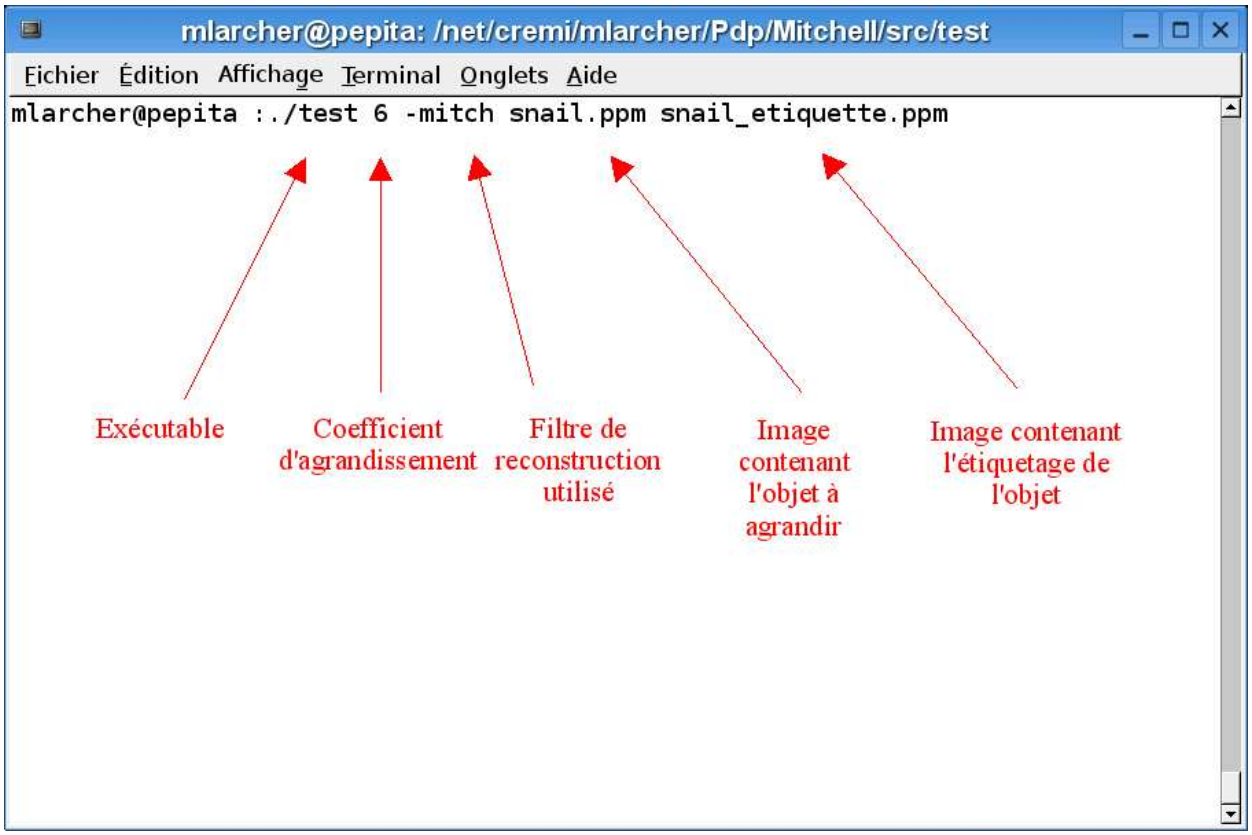

FIG. 2.2 – Exemple de fonctionnement

Ici on agrandit l'objet décrit par l'étiquette "snail_étiquette.ppm" (figure 2.4) qui est contenu dans l'image "snail.ppm" (figure 2.3), par un facteur de 4. On obtient en sortie,un fichier ppm incrusté par défaut sur un fond gris.
Il est aussi possible de retourner deux fichiers, l'un contenant l'objet agrandi et l'autre son canal alpha, ou encore directement un fichier png qui gère lui-même la transparence.

Voici les résultats obtenus, avec le fichier image et le fichier étiquette suivants. L'exemple suivant est principalement destiné à montrer le lissage sur le bord de la coque du bateau. La zone définie par l'étiquette n'est pas totalement uniforme.

FIG. 2.3 – Fichier image

FIG. 2.4 – Fichier etiquette

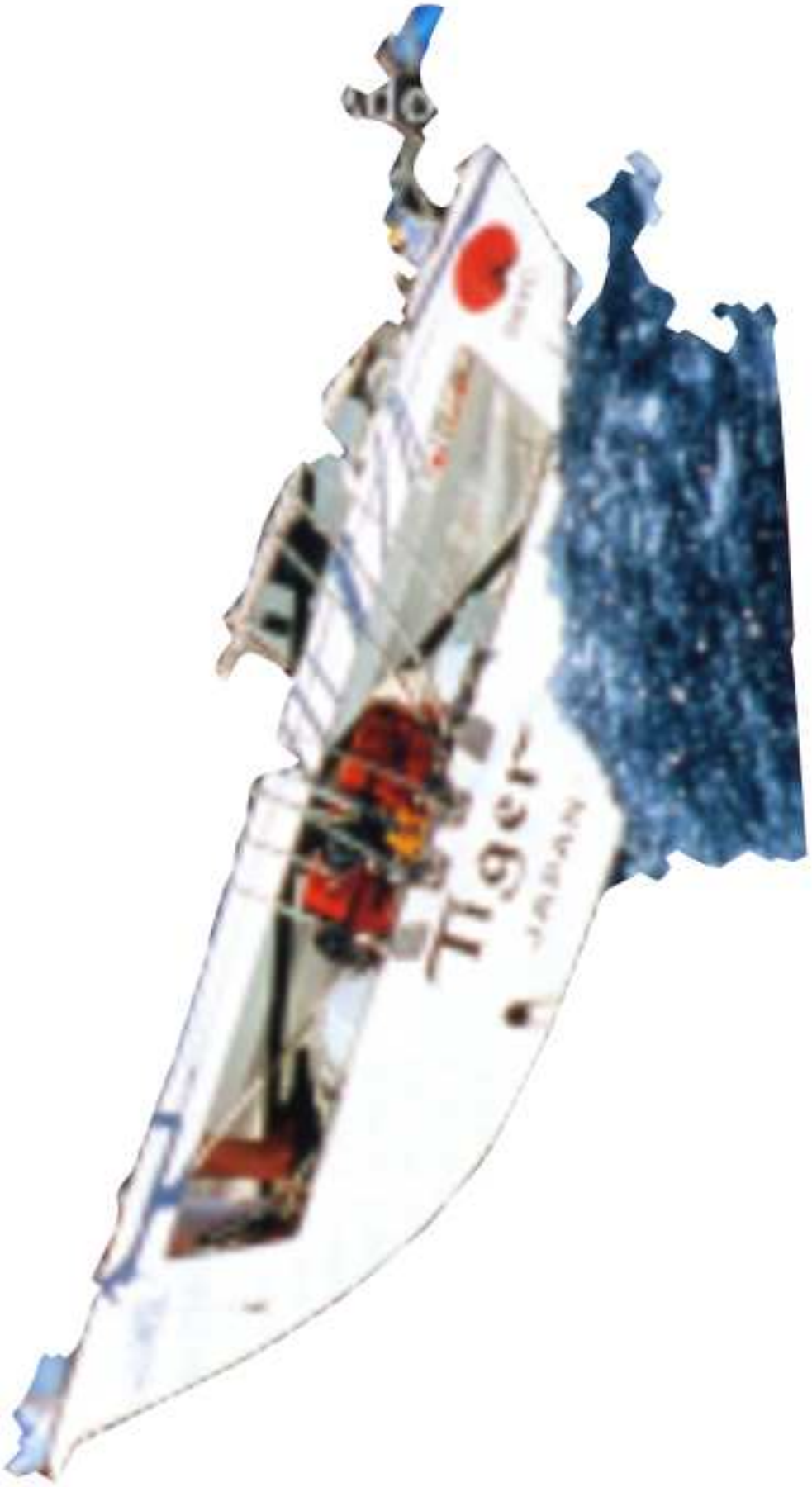

FIG. 2.5 – Objet agrandi : on remarque le lissage sur les bords de la coque du bateau

Chapitre 3

Architecture et découpage modulaire

L'architecture a été conçue de manière à permettre des extensions au programme. Ainsi la structure TabPixel permet de faire abstraction du format de fichier utilisé, et même si pour le moment nous nous sommes contentés de pnm, d'autres formats peuvent être ajoutés facilement sans affecter le fontionnement du programme.

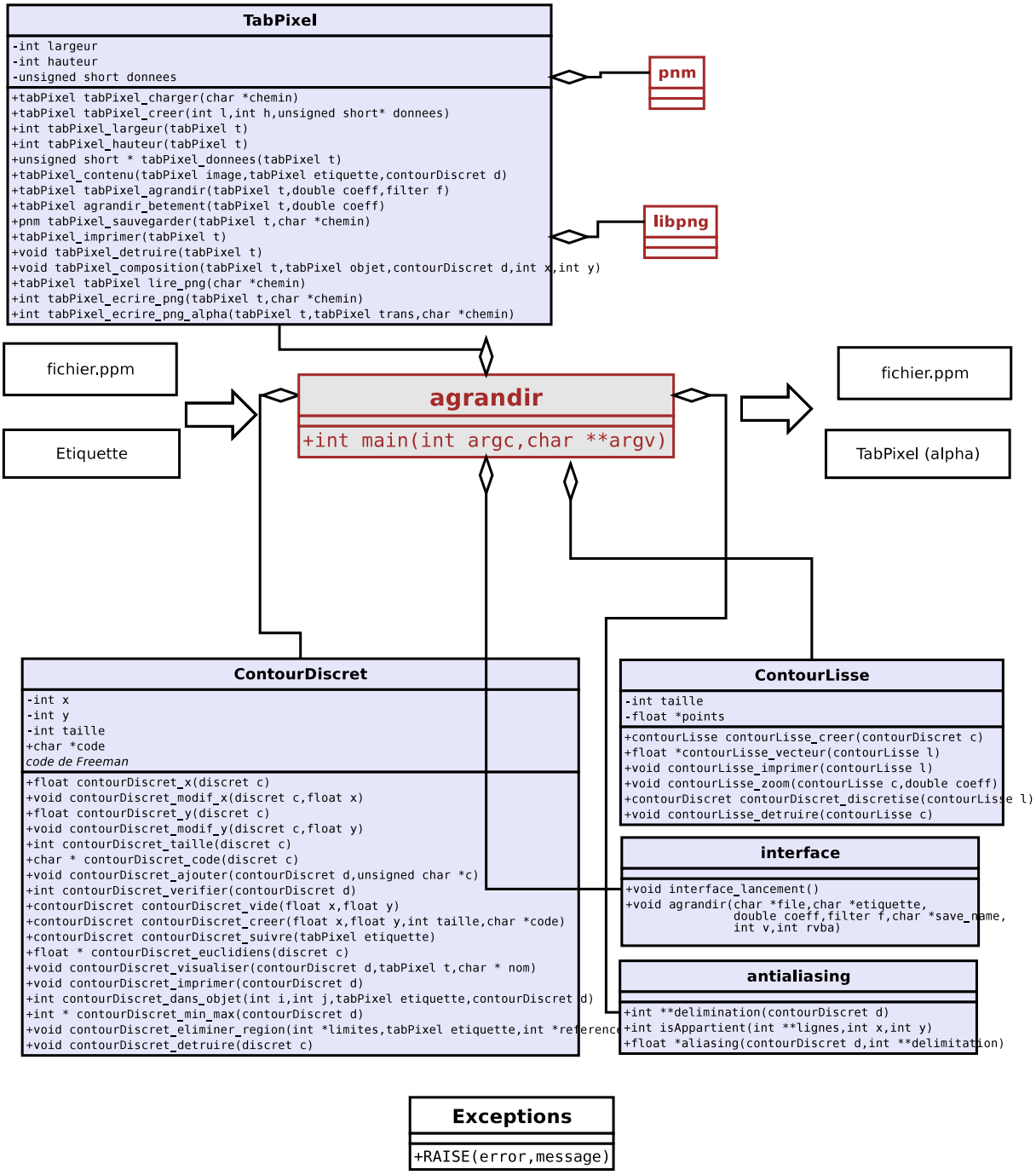

FIG. 3.1 – Architecture

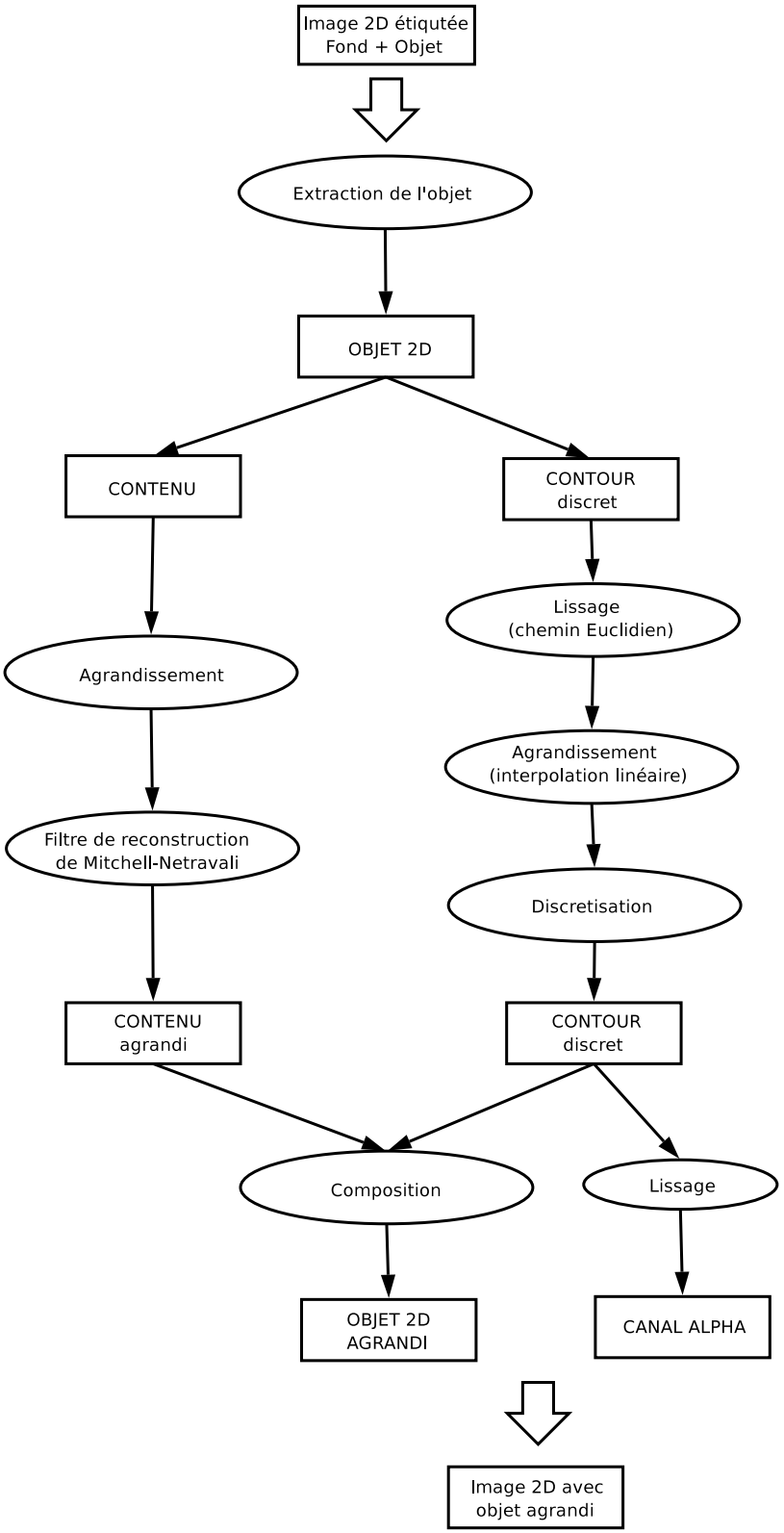

FIG. 3.2 – Déroulement

Chapitre 4

Structures de données et algorithmes

4.1 Structures

4.1.1 TabPixel

TabPixel est la structure de données qui nous permet de représenter l'image. Toutes les opérations de lecture et d'écriture d'image passent par elle.

largeur, hauteur

La largeur et la hauteur sont représentées par deux entiers et correspondent aux dimensions de l'image. Ces deux éléments sont très souvent utilisés notament pour ajuster la valeur de "shift_Y"nécessaire au parcours des données contenues par le TabPixel.

données

Données est un pointeur sur "unsigned short"qui contient les valeurs des composantes rouge, verte et bleue de chaque pixel de l'image représentée par le TabPixel. Les données associées à ce pointeur sont les plus utilisées tout au long du programme. En effet chaque pixel de l'image a deux coordonnées : x et y. Cependant ils sont ensuite stockés dans un tableau a une dimension à raison de trois valeur par pixel (une chaque composante), et pour retrouver le pixel souhaité à partir de sa position, on a recours à deux valeurs shift_X et shift_Y.

Par exemple, pour retrouver les trois composantes du pixel à la position (i, j), on lit le pointeur données aux positions suivantes : i*shift_X + j*shift_Y (rouge), i*shift_X + j*shift_Y + 1 (vert), i*shift_X + j*shift_Y + 2 (bleu), où shift_X = 3 (trois composantes), et shift_Y est égal à la largeur de l'image.

4.1.2 ContourDiscret

ContourDiscret permet de représenter le contour de notre objet, à l'aide d'un point de départ et d'un code de Freeman.

x, y

Les deux entiers x et y sont les coordonnées du point de départ du contour, il est intéressant de note qu'un contour étant situé "entre les pixels", ne possède pas de coordonnées entière mais demi-entières.

taille

Nous avons besoin de connaitre le nombre de déplacements nécessaire au suivi du contour afin de le parcourir, et cela à nombreuses reprises. C'est pourquoi nous avons choisi d'intégrer un entier représentant ce nombre à la structure de données afin d'y accéder facilement.

code

Le pointeur sur caractère "code" contient le code de Freeman relatif au contour de notre objet, tout simplement une suite composée de valeurs 0 (droite), 1 (haut), 2 (gauche) ou 3 (bas).

4.1.3 ContourLisse

ContourLisse contient les coordonnées de notre contour discret sous forme réelle, ceci avant et après son agrandissement.

nb_points

De la même manière que ContourDiscret contenait le nombre de déplacements contenus dans "code", ContourLisse contient le nombre de points du contour lisse, ceci afin de faciliter le parcours de ces points.

points

Points est un pointeur sur réel (float), qui contient la liste des points à coordonnées réelles du contour.

4.1.4 filter.h et pnm.h

Les fichiers filter.h et pnm.h nous ont été fournis par notre client. Le premier contient un certain nombre de filtres de reconstruction utilisés lors de l'agrandissement d'images. Ils nous a donc suffit de reprendre ce fichier afin de l'intégrer à notre programme et proposer à notre tour l'utilisation de ces différents filtres.
Quant au second, il permet de manipuler les fichiers au format pnm : lecture, écriture et modification. Nous avons donc fait le lien entre ces fonctions relatives à un format de fichier et TabPixel qui lui est indépendant du format.

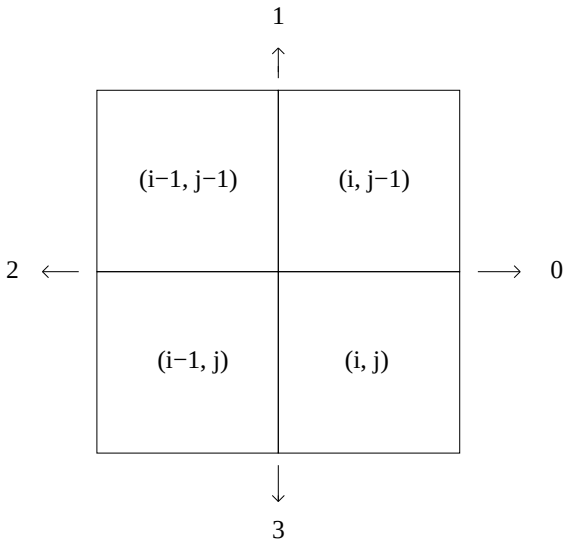

FIG. 4.1 – Suivi du contour et codage de Freeman

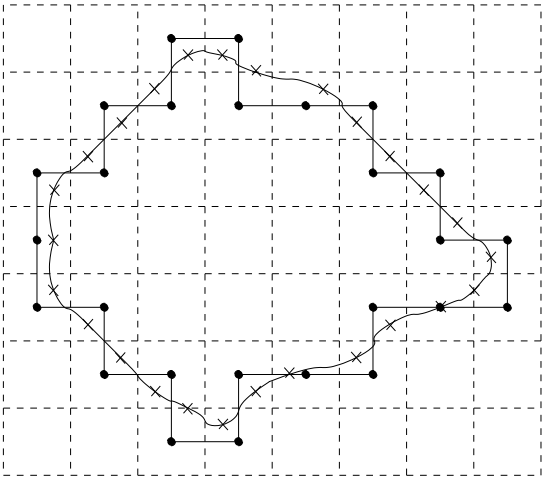

FIG. 4.2 – Contour discret et chemin euclidien.

4.2 Algorithmes et techniques

4.2.1 Suivi de contour

La technique employée pour le suivi de contour n'implique pas un grand nombre de calcul et ne comporte fondamentalement que quatres cas à traiter.

Ainsi après avoir initialisé notre structure TabPixel représentant le fichier étiquette, on parcourt les pixels en commençant par celui situé le plus en haut à gauche (coordonnées 0,0), il contient la couleur du fond. Puis cela ligne par ligne, nous nous arrêtons lorsque l'on rencontre un pixel de couleur différente de celle du fond. C'est alors que nous avons trouvé le premier pixel de notre étiquette.
Par convention, on choisit de suivre notre contour dans le sens trigonométrique.

Le code de Freeman du contour discret figure 4.2 ayant comme point de départ le point le plus en haut à gauche est le suivant :
$3 \downarrow 2 \leftarrow 3 \downarrow 2 \leftarrow 3 \downarrow 3 \downarrow 0 \rightarrow 3 \downarrow 0 \rightarrow 3 \downarrow 0 \rightarrow 1 \uparrow 0 \rightarrow 0 \rightarrow 1 \uparrow 0 \rightarrow 0 \rightarrow 1 \uparrow 2 \leftarrow 1 \uparrow 2 \leftarrow 1 \uparrow 2 \leftarrow 2 \leftarrow 1 \uparrow 2$

Il nous suffit alors de comparer les couleurs des pixels voisins, et d'ajouter la valeur codant le déplacement en cas de couleur différente.

4.2.2 Chemins euclidiens

Voir page 19 et document **??**.

4.2.3 Filtre de Mitchell

4.2.4 Antialiasing

On souhaite antialiaser les bords de la région extraite agrandi avant la composition de l'objet sur un fond. Le canal alpha est un 4ème canal (RGB) qui permet de stocker la transparence de chacun des pixels de l'image. Si alpha = 1, le pixel est opaque, et si alpha =0, le pixel est transparent, et si 0 ¡ alpha ¡ 1, la nouvelle couleur du pixel est un mixage de la couleur du pixel avec celle du fond, calculée selon alpha.

Filtrage

L'antialiasing peut être effectué grâce aux filtres utilisés en imagerie (tel que filtre de Gauss par exemple) sur le contour de l'objet qui va remplacer chacun des pixels traités par un mélange des couleurs de ses voisins. Mais il laisse encore des défauts, car le filtrage rend le crénelage flou mais toujours présent. On choisit donc d'effectuer l'antialiasing en approximant les frontières réelles.

Principe

Méthode proposée dans le document **??**. On se basera sur la construction du chemin euclidien dirigé par la tangente afin de calculer le canal alpha sur les bors de l'objet. Pour le reste des pixels contenus dans la bote englobante, cet alpha dépendera juste de leur appartenance à l'objet discret ou non.
Notre but est donc de calculer pour chaque pixel traversé par le chemin euclidien l'aire de recouvrement du domaine euclidien qui sera la valeur de alpha.

L'algorithme va s'effectuer sur tous les points du contour discret. La cellule qui encadre ce point Pi est à l'intersection de quatres quarts de pixels qui entourent eux-mêmes ce point et donc fournit des données pour quatre pixels. Il faudra donc recomposer pour chaque pixel ses quatre quarts de pixel trouver par les points de contour Pi.

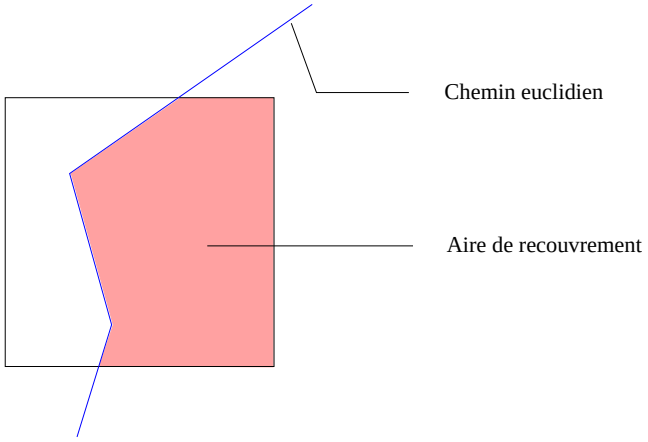

FIG. 4.3 – Aire de recouvrement du domaine euclidien d'un pixel

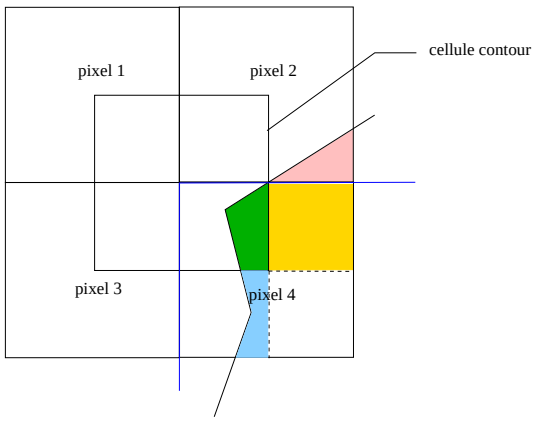

FIG. 4.4 – Aire de recouvrement du domaine euclidien sur quatre pixels

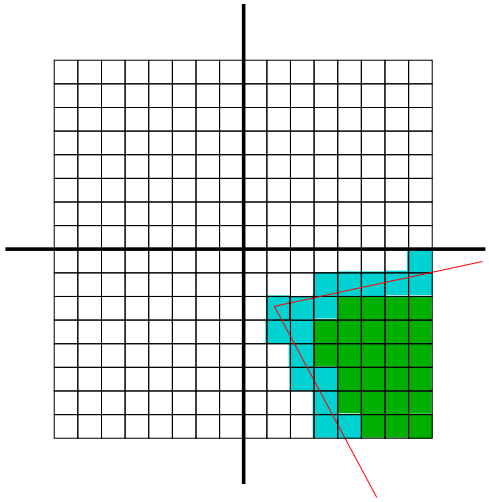

FIG. 4.5 – Aire de recouvrement d'une cellule

Pour chaque point Pi du contour discret, on doit donc calculer l'aire engendrée par les points euclidiens pi, pi-1 et pi+1. Pour cela, on utilise une méthode de suréchantillonage de la cellule contour. On a utilisé une résolution de 8*8 qui parait assez satisfaisante étant donné que l'on traite des pixels.

Ainsi on obtient alpha en comptant le nombre de sous-cellules qui appartiennent au domaine euclidien.

Chapitre 5

Complexité des principaux algorithmes

5.1 Chemins Euclidiens

Voir document [8].

5.2 Antialiasing

- D'abord, on fait un parcours du contour discret qui construira une structure de données permettant alors de savoir si un pixel x appartient ou non à l'objet discret de façon très rapide.
- L' algorithme traite tous les points du contour discret pour lesquels il construit la grille subdivisée de la cellule. Il doit ensuite trouver quelle partie de la grille appartient au domaine euclidien. Puis il ne lui reste plus qu'à compter les cases de cette partie.
- Tous les alphas des quarts de pixels trouvés sont stockées dans un tableau trié au fur et à mesure. Il faut encore réorganiser ce tableau de façon à obtenir les alphas des pixels entiers.

La complexité de ces trois étapes est fonction du nombre de points du contour discret.

Chapitre 6

Tests

6.1 Tests unitaires

6.1.1 Code de Freeman

Nous avons comparé les résultats obtenus lorsque nous suivons le contour de l'étiquette, avec ceux retournés par le module de suivi de contour qui nous été fourni par Anne Vialard, ceci bien sûr sur le même fichier étiquette. Ils sont pour ainsi dire identiques car la seule différence observée concerne le point de départ du contour qui n'est pas le même et donc le code de Freeman est d'écalé d'un déplacement.
Nous sommes donc assurés que notre travail se base sur un résultat exacts.

6.1.2 Chemins euclidiens

De même, les résultats obtenus lors du calcul des chemins euclidiens à partir du contour discret ont été comparés à ceux retournés par le module codé par Anne Vialard.
Ici la différence se fait plus remarquer, mais reste tout de même acceptable, en effet elle est à la hauteur de plusieurs centièmes.

6.1.3 Agrandissement

Vérifions l'agrandissement d'une image en comparant la taille avant et après agrandissement, et bien sûr le facteur.
L'agrandissement est effectué par un facteur 2.
Après ouverture des deux fichiers ppm avec un éditeur de texte, on compare les tailles spécifiées dans les entêtes :

```
Snail.ppm : 256 256
Snailx2.ppm : 512 512
```

Valeur cohérente.

Notre module		Module codé par A.Vialard	
x	y	x	y
78.470589	213.529412	78.375000	213.625000
77.764706	213.235294	77.714286	213.285714
77.058823	212.941176	77.050000	212.950000
76.352940	212.647059	76.350000	212.650000
75.647060	212.352941	75.650000	212.350000
74.941177	212.058824	74.950000	212.050000
74.235294	211.764706	74.250000	211.750000
73.529411	211.470588	73.529412	211.470588
72.823528	211.176471	72.823529	211.176471
72.117645	210.882353	72.142857	210.857143

FIG. 6.1 – Comparaison des résultats du calcul du chemin euclidien

```
>> scale < 1.0
Abandon
```

FIG. 6.2 – Facteur spécifié incorrect

6.2 Tests de robustesse

6.2.1 Facteur d'agrandissement maximal

Un petit objet de taille environ 50x50 pixels a été agrandi 100 fois avec succès, nous n'avons pas poussé plus loin nos tests du plus grand facteur d'agrandissement spécifiable considérant qu'un facteur supérieur déjà qu'agrandir 100 fois un objet est inutile.

6.2.2 Objet de forme peu commune

L'application a été testée avec différents fichiers étiquette de forme inattendue. Le comportement a été normal, et le résultat conforme avec les objets suivants :

- Objet dont la longueur ou la largeur est de 1 pixel.
- Objet dont la forme est complexe et dont les bords ne sont jamais droits.
- Objet composé d'un seul pixel.

Cependant l'exécution ne se déroulera pas si un ojet est situé sur le bord de l'image, nous avons exclu ce cas qui apporte différentes complication et pourrait être traité lors d'une eventuelle extension du programme.

6.2.3 Fichier image de format inconnu, et facteur d'agrandissement inférieur à 1

L'utilisation d'un facteur d'agrandissement inférieur à 1 fait appel au module de gestion d'erreurs fourni par notre client. Ainsi un message précisant que le facteur doit être supérieur à 1 apparait (figure 6.2).

6.3 Tests de validation

6.3.1 Fiabilité

Notre logiciel peut être considéré comme fiable car en suivant la chane de traitement de bout en bout comme illustré dans le déroulement [Fig 42], il est capabe de restituer fidèlement toutes les différentes caratéristiques de l'image source. Cette vérification est basée sur le résulat visuel de l'agrandissement d'une image donnée en entrée.

6.3.2 Rapidité

Nous nous sommes fixés comme moyenne en temps de réponse la seconde.Toutefois, un agrandissement d'image peut prendre un temps un peu plus long selon les paramètres d'entrée(nous restons néanmoins dans l'ordre fixé). Aussi compte tenu de la contrainte inhérente au temps de réponse, nous n'avons utilisé que des algorithmes de complexité linéaire.

6.3.3 Complexité finale

Comme vu auparavant, la complexité des trois principaux algorithmes employés lors du traitement des images est reconnue linéaire. Leur utilisation séquentielle est donc aussi de complexité linéaire.

Chapitre 7

Récapitulatif et extensions possibles

7.1 Travaux accomplis

Le chargement en mémoire d'images aux formats ppm et png est géré par notre application, la lecture de l'étiquette et le suivi du contour permettent l'extraction de l'objet et la représentation de son contour sous forme discrète. L'implémentation de différents filtres et notamment celui de Mitchell permet d'agrandir les régions de couleur unie ou dégradée avec un résultat visuel proche de la perfection. Enfin le calcul rapide des chemins euclidiens associé au lissage des bords délivre un résultat de grande qualité : une image avec des bords lisses et un contenu net.

De plus il est possible de traiter des images contenant plusieurs objets avec un résultat satisfaisant (l'implémentation de cette partie n'ayant pas pu être terminée faute de temps). Voir figure 7.1.

Comme on peut le voir figure 7.3 le résultat n'est pas encore tout à fait au point. L'inconvénient étant la gestion de frontières communes à plusieurs objets.
En effet , pour obtenir un bon résultat, les points de départ pour le calcul des chemins euclidiens doivent être les mêmes pour les deux objets le long de leur frontière commune. figure 7.4. Ici les chemins sont différents et on aperçoit donc des pixels blancs gênants.

FIG. 7.1 – Multiples objets

FIG. 7.2 – Etiquetage correspondant

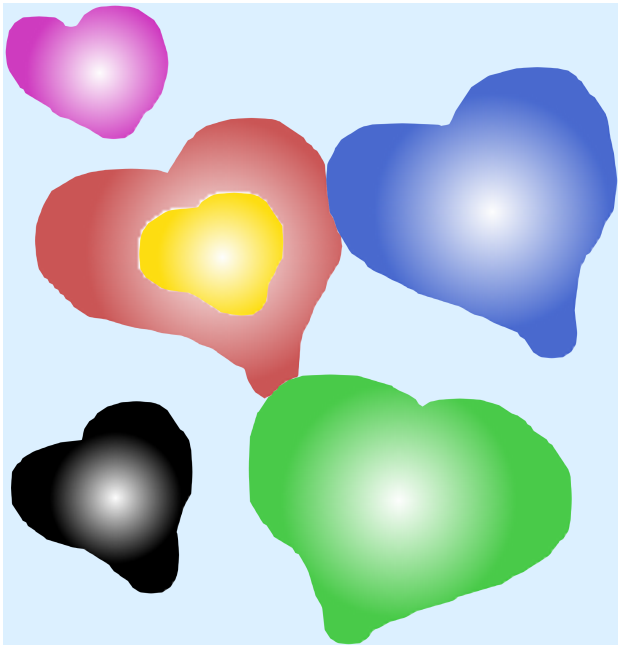

FIG. 7.3 – Résultat

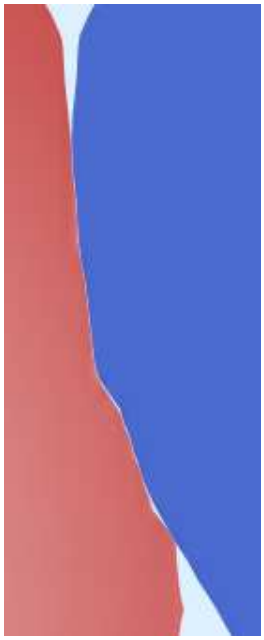

FIG. 7.4 – Détail d'une frontière commune à deux objets

Aussi on peut noter l'apparition des mêmes pixels blancs à l'intérieur de la région à trou. Le problème des régions à trous étant assez complexe nous n'avons pas eu le temps nécessaire pour le traiter correctement.

7.2 Application à plusieurs objets (totale)

Initiallement ce programme traite un seul objet, donc une image avec 2 étiquettes différentes (fond et objet), mais il a été étendu à plusieurs régions, donc un nombre détiquettes supérieur à 2. L'étiquette du premier pixel de l'image sera considérée comme celle du fond, la suivante comme la première région. A la rencontre de cette dernière, on traite celle-ci (suivi du contour, extraction de l'objet), puis on l'élimine en remplaçant son étiquette avec celle du fond et on relance la recherche de nouvelles régions jusqu'à ce que l'on n'en trouve plus.

Cependant, les régions à trous (traitées dans la section suivante) et celles à frontières communes méritent une attention particulière. Dans le cas de ces dernières, un traitement adapté de la partie commune de leurs frontiìres doit être fait. Le premier et le dernier point ne doivent être définis comme étant "fixes", c'est-à-dire qui ne doivent pas être déplacés par le lissage. Ainsi, nous sommes sûrs d'obtenir le même Chemin Euclidien dans les contours des deux régions et obtenir les bons résultats en effectuant l'antialiasing.

7.3 Application aux objets "à trous"

Les objets "à trous" correspondent à des régions à plusieurs contours. Un exemple est illustré sur la figure 7.6, page 54. Notre logiciel ne les traite pas actuellement, mais c'est

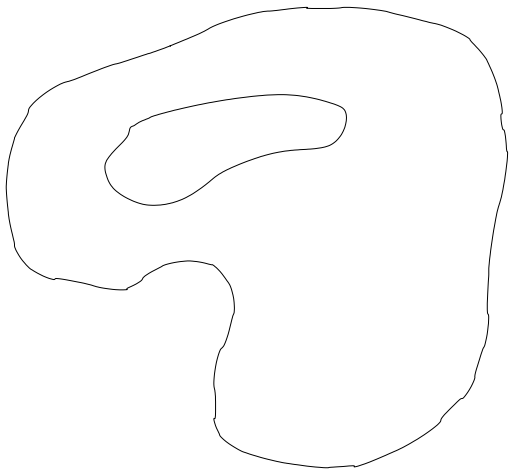

FIG. 7.5 – Région trouée

une extension possible. Il suffit de créer une nouvelle structure dans laquelle on pourrait attribuer à un objet plusieurs objets de type contourDiscret.

7.4 Application aux objets situés sur le bord de l'image

Une des conditions d'utilisation de notre logiciel est que les objets ne se situent pas sur le bord de l'image (c'est-à-dire avoir au minimum une ligne de pixels entre la région et le bord), car l'algorithme de suivi du contour devient très complexe dans le cas contraire.

Ainsi, si l'image à traiter contient des objets sur le bord, il suffit de rajouter au préalable une ligne supplémentaire de pixels pour assurer les bonnes conditions du suivi du contour.

7.5 Véritable détection de contours permettant de s'affranchir de l'étiquetage

Actuellement nous travaillons par régions définies au préalable, mais une des extensions possibles de notre logiciel est celle de la recherche de contours fermés dans l'image. Ainsi la recherche de régions se fera automatiquement par le logiciel et l'utilisateur pourrait donner seulement l'image à traiter en entrée.

Cependant les algorithmes actuels de détection de contours ne sont pas suffisants malgré leur relative efficacité. En effet ils permettent de détecter et dessiner les arêtes présentes à l'image mais nous sommes loin d'obtenir des r´´egions clairement définies (voir figure 1.6). Il faudrait donc ajouter un traitement supplémentaire pour dessiner des régions à partir des arêtes ce qui est relativement complexe.

L'autre possibilité considérable est d'utiliser un technique de segmentation d'image par couleur. On quadrille alors l'image et on obtient un certain nombre de points de départs, qu'on "propage" suivant les pixels voisins de couleur similaire.

Cette extension est donc la plus conséquente mais palierait le plus gros inconvénient pour l'utilisateur.

Manuel d'utilisation

Notre logiciel ne prend en compte que les images couleur au format ppm et/ou png

Options Il s'agit de décrire dans cette partie, le rôle et le fonctionnement de chacun des outils proposés par notre application. Une fois le programme compilé et exécuté, la page principale s'affiche.

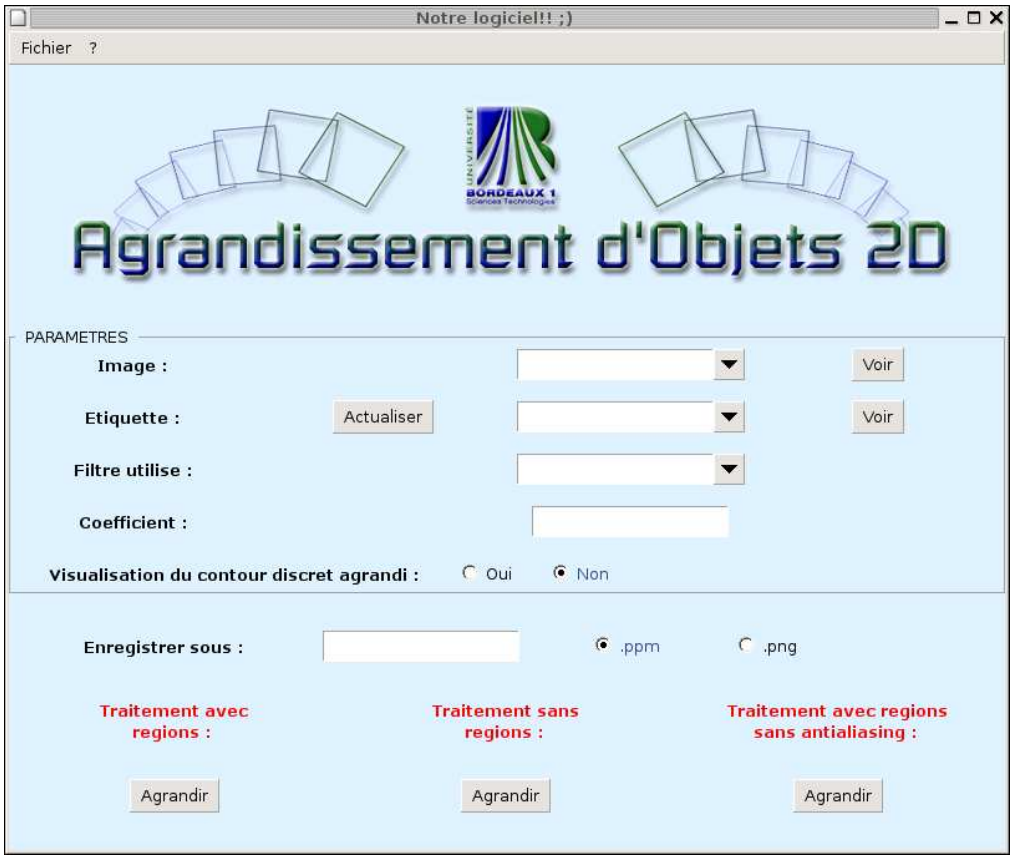

FIG. 7.6 – Page principale

Chargement et visualisation (image/étiquette)
Rôle et Fonctionnement.
Le bouton correspondant à "fichier à traiter" permet de charger une image/étiquette au format ppm ou png. On peut aussi spécifier un nom de fichier manuellement. Une fois le chargement effectué nous pouvons visualiser 7.7 l'image grâce au bouton voir.

FIG. 7.7 – Sélection du fichier image à traiter

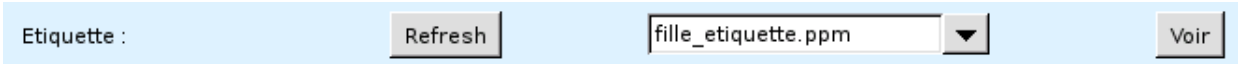

FIG. 7.8 – Selection du fichier étiquette

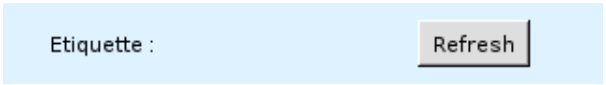

FIG. 7.9 – Bouton refresh

Refresh
Rôle et Fonctionnement.
Le bouton refresh a pour rôle de générer un étiquette relative à l'image visualisée 7.13. Une fois l'image à visualiser choisie, un clic sur le bouton refresh génère une liste d'étiquettes. On peut donc en sélectionner une et la visualiser.

Coefficient (d'Agrandissement)
Rôle et Fonctionnement.7.10
Avec cette option nous avons la possibilité d'effectuer un agrandissement de l'image avec le facteur de zoom désiré.

Filtres
Le logiciel permet l'utilisation de plusieurs filtres.
Fonctionnement.7.11

Agrandissements
- Régions avec lissage
- Régions sans lissage
- Sans régions 7.12

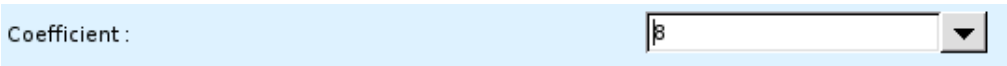

FIG. 7.10 – Coefficient

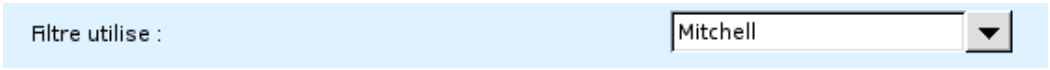

FIG. 7.11 – Différents filtres

FIG. 7.12 – Choix du format

FIG. 7.13 – Choix du type d'agrandissement

Bibliographie

[1] http://homepages.inf.ed.ac.uk/rbf/HIPR2/featops.htm. **Features detector**. 2005.

[2] http://www.bibl.ulaval.ca/vitrine/giri/mod1/1_4.htm. **Principaux formats de fichiers**.

[3] http://www.cse.nd.edu/ kwb/HeathSarkarSanockiBowyerPAMI_1997.pdf. **A Robust Visual Method for Assessing the Relative Performance of Edge-Detection Algorithms**. 2005.

[4] http://www.ext.upmc.fr/urfist/image_numerique/format.htm. **Format d'image**.

[5] http://www.gpa.etsmtl.ca/cours/sys844/Documents/Chapitre4.pdf. **Détection des arêtes d'une image**. 2005.

[6] http://www.ign.fr. **Site officiel IGN**. 2005.

[7] **Fanny Chevalier, Eric Hany, Gamou Seck, Sidiki Tall**. **Traitement d'images, agrandissement par régions**, 2003.

[8] **J.P. Braquelaire and A. Vialard**. Euclidean paths : A new representation of boundary of discrete regions, 1995.