République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud MAMMERI, Tizi-Ouzou

Faculté de Génie Electrique et d'Informatique
Département d'Automatique

Mémoire de Fin d'Etudes

En vue de l'obtention du diplôme

Master académique en automatique
Option commande des systèmes

Thème

Etude des contours actifs et introduction aux multi-objets

Dirigé par :

M^elle^ CHILALI

Présenté par :

M^elle^ DALI Lilia

M^r^ DEBIANE Mohamed Amine

Soutenu le : 19 / 09 /2012

Promotion 2012

Remerciments

En premier, nous adressons nos plus vifs remerciements à notre promotrice Melle Chilali pour les conseils avisés qu'elle nous a toujours prodigués, pour les connaissances dont elle nous a fait bénéficier, son suivi attentif et sa confiance qui nous a été très précieuse. Mais aussi nous tenons à la remercier pour nous avoir fait l'honneur de nous encadrer.

Nos remerciements les plus sincères vont à l'ensemble des membres de jury qui nous ont fait l'honneur de juger ce travail.

Nous tenons à remercier également l'ensemble des enseignants qui ont contribué à notre formation.

Nous tenons aussi à remercier l'ensemble des administrateurs du département d'automatique, à leur tête Mr Ben Sidhoum.

Nos sincères remerciements vont aussi à toutes les personnes qui, de prés ou de loin ont contribué à la réalisation de ce travail.

Nous voudrions enfin remercier en particulier toute la promotion d'automatique 2012.

Lilia & Mohamed amine

Dédicace

A ma chère mère.

A mon cher père.

A ma chère grand-mère.

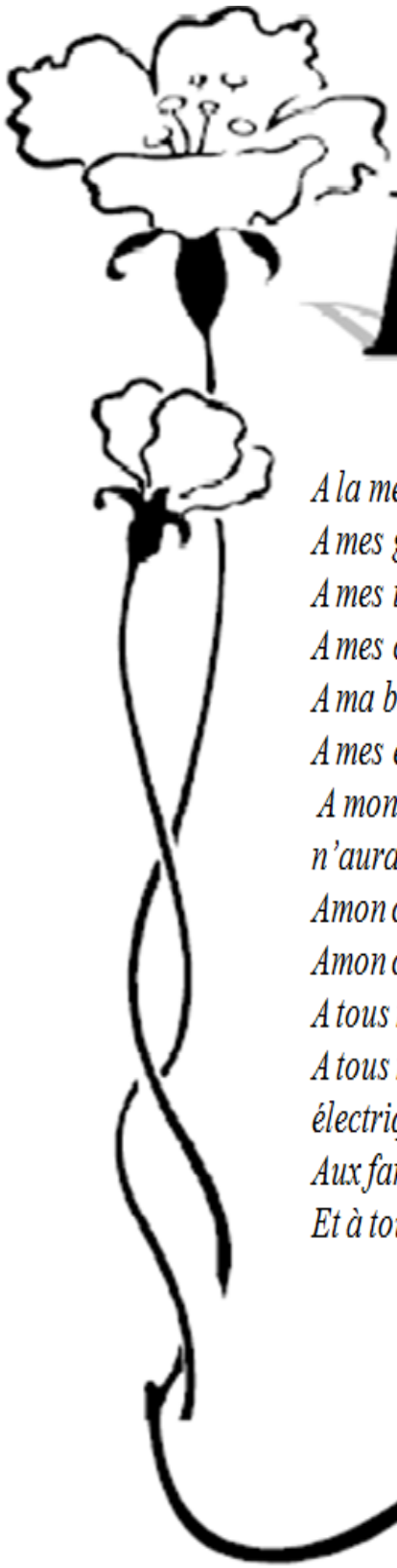
A mes chers frères.

A tous mes amis et amies.

Et à tous ceux qui me sont chers.

je dédie ce modeste travail

Lilia DALI



Dédicace

A la mémoire de mon défunt grand-père.
A mes gands parents.
A mes très chers parents.
A mes chers frères Anis et Abd Elhalim.
A ma belle sœur Kahina.
A mes encles et tantes.
A mon binôme lilia qui sans elle ce travail n'aurait pas été fait.
Amon cher ami Talem hand.
Amon cousin le grand Ahmed.
A tous mes amis de la ligue LAPALEJ.
A tous mes amis et amies de la faculté du génie électrique .
Aux familles DEBIANE ET DALI.
Et à tous ceux qui me sont chers.

Je dédie ce modeste travail

Mohamed Amine DEBIANE

Résumé

Les contours actifs font partie des méthodes de segmentation les plus répandues. Ils sont utilisés dans de nombreuses applications de reconstruction, de détection ou encore de suivi d'objet aussi bien 2D que 3D. On discerne deux familles de représentation des contours actifs. La représentation paramétrique échantillonne la courbe selon son abscisse curviligne de manière à guider la déformation du contour actif à l'aide de points de contrôle. La représentation implicite, appelée également représentation en ensembles de niveaux, détermine la déformation du contour actif en analysant le comportement d'un modèle de dimension supérieure. Le principal avantage de la représentation paramétrique est la rapidité de segmentation alors que la représentation implicite reste la plus efficace en termes de précision, autorisant la courbe à gérer naturellement les changements de topologie. L'objectif de ce mémoire est de proposer des améliorations pour la représentation paramétrique afin de la rendre apte à effectuer des changements de topologie automatiquement tout en gardant sa propriété de rapidité de segmentation. Pour cela nous avons adopté une approche appelée contours actifs multi objets, qui consiste en la réalisation de deux étapes : la détermination des points critiques et des paires de points critiques où serra, ensuite, effectués les deux processus de division et de raccordement des points, de manière à obtenir plusieurs contours à partir de l'initialisation d'un seul contour.

SOMMAIRE

CHAPITRE 3 : Test et résultats

Le traitement d'images voit ses débuts dans les années 1920 dans la transmission de données par câble, mais ne connaît de vrai essor que dans les années 1960 avec le développement des ordinateurs. Au départ, les techniques de traitement d'images sont, essentiellement, des méthodes de restauration et de compression d'images. Puis se développent, avec les progrès de l'informatique, des techniques de détection de primitives (contours, point d'intérêt, lignes d'intérêt, etc.) et de nombreux autres traitements dans les domaines aussi variés que le médical, la télévision, l'imagerie satellitaire, le multimédia. C'est dans les années 2000 que l'image numérique, et par conséquent le traitement d'images, devient omniprésent. Que cela soit sur internet, au cinéma, à la télévision, sur les téléphones, dans le domaine médical, l'image est partout. Aujourd'hui, il ne s'agit plus uniquement de traiter les images pour les améliorer mais aussi de les comprendre et de les interpréter. C'est dans ce contexte que la reconnaissance d'objets dans les images devient un sujet de recherche important. Et pour reconnaître des objets afin d'interpréter les images, il faut au préalable les segmenter.

Le processus de segmentation consiste à isoler une ou plusieurs structure(s) d'intérêt présente(s) dans une image. Il est courant de classer les méthodes de segmentation en deux catégories : les approches contour (ou frontière) et les approches région. Les approches contour abordent la segmentation comme la recherche des frontières entre les objets et le fond. A l'inverse, les approches région réalisent la segmentation en partitionnant l'image en zones vérifiant un critère d'homogénéité. Ces deux approches réalisent une segmentation globale de l'image en se basant uniquement sur des caractéristiques basées sur les pixels. La notion de forme n'est pas intégrée dans le processus de segmentation, ce qui peut être un inconvénient lorsque l'on cherche à déterminer les frontières d'un objet en particulier. De plus, dans le cas d'images réelles, les données sont généralement incertaines pour diverses raisons : dégradation par le bruit, frontières indistinctes dues à un phénomène de flou, problèmes d'occultations, etc. La segmentation ne peut plus être réalisée selon des critères bas niveau relatifs aux pixels.

L'introduction d'un modèle de la forme recherchée est alors nécessaire. Dans ce contexte, les contours actifs permettent l'adjonction de contraintes et de connaissances *a priori* sur les objets à segmenter. Un contour actif est une structure géométrique évoluant itérativement de manière à s'ajuster aux frontières des objets recherchés. Il est, généralement, représenté par une courbe dans une image 2D ou une surface dans une image 3D. La forme et la position initiale de la courbe ou de la surface sont fournies de manière manuelle ou

automatique. Dans le formalisme des contours actifs, la segmentation est la plupart du temps formulée comme un problème d'optimisation. Le but étant de déterminer la courbe ou la surface minimisant une fonction objectif. Cette fonction, souvent appelée fonctionnelle d'énergie, est la traduction mathématique des propriétés recherchées sur les objets à segmenter.

Les éléments clés de la méthode des contours actifs sont l'élaboration de la fonctionnelle d'énergie, le choix d'une procédure de minimisation, et d'une représentation géométrique. Dans ce mémoire, nous aborderons ces trois éléments. Nous allons définir les deux principales représentations des contours actifs et nous verrons que le principal avantage de la représentation paramétrique est la rapidité de segmentation, alors que la représentation implicite reste la plus efficace en termes de précision autorisant la courbe à gérer naturellement les changements de topologie. Nous allons, aussi, essayer d'améliorer la représentation paramétrique, en la rendant apte à effectuer des changements de topologie automatiquement tout en gardant sa propriété de rapidité de segmentation.

Pour se faire, nous avons structuré notre travail en trois chapitres.

Dans le 1er chapitre, nous allons nous intéresser à la forme générale d'un contour actif. Nous verrons qu'un contour actif est constitué d'une énergie définissant ses propriétés géométriques et d'une autre relative à l'image, et qu'il se possède deux représentations. Nous allons nous intéresser, dans ce chapitre, à la représentation implicite et mettrons l'accent sur la représentation par ensembles de niveaux.

Dans le 2ème chapitre, nous allons nous intéresser à la représentions paramétrique, notamment les *snakes*. Nous verrons que l'utilisation des *snakes* dans une application donnée passe par 3 étapes fondamentales, qui sont la définition de la fonctionnelle d'énergie, la détermination de l'équation d'évolution, et enfin implémentation de la méthode. Parmi les méthodes d'implémentation nous mettrons l'accent sur l'algorithme de *Greedy*. A la fin du chapitre, nous introduisons le multi-objets.

Quant au dernier chapitre, nous allons le consacrer aux différents tests et résultats, tout en détaillant l'algorithme *Greedy* et l'approche multi-objets.

Nous terminerons notre mémoire par une conclusion générale.

1.1 Introduction

Depuis leur apparition dans [35], les contours actifs sont devenus une des méthodes de segmentation les plus populaires. Leur capacité de segmentation est utilisée dans de nombreux domaines tels que le suivi d'objet [18], la segmentation d'images médicales [19] ou encore la reconstruction 3D [20]. En outre, indépendamment du domaine d'application, la théorie même des contours actifs a donné lieu à de nombreuses contributions, tant à ses débuts que récemment [21].

Une conséquence directe de cet engouement pour la segmentation par contour actif est la grande diversité de modèles présents dans la littérature. Ce qui nous amène dans ce chapitre à définir, tout d'abord, d'une manière générale, les contours actifs pour, ensuite, s'intéresser aux contours représentés de manière implicite. Nous allons définir le concept fondamental de la théorie d'évolution de courbe. Nous allons voir la représentation, l'évolution, la mise en œuvre numérique des *level set*, et l'analyse des caractéristiques de ce type de contours actifs.

1.2 Définition et principe des contours actifs

Un contour actif est représenté par une courbe paramétrée C(s), fermée ou ouverte à extrémités fixes ou non. Elle est définie dans une image Ω_I, associant au paramètre s un pixel $v = (x, y)$ tel que :

$$C : \Omega_I = [0,1] \rightarrow \mathbb{R}^2 \quad s \rightarrow C(s) = v(s) = \begin{bmatrix} x(s) \\ y(s) \end{bmatrix}, \forall\, s \in \Omega_I \tag{1.1}$$

Le domaine de définition de la courbe est normalisé à [0,1]. s est l'abscisse curviligne représentant le paramètre de position spectrale le long de la courbe, et x et y sont les coordonnées cartésiennes (figure1.1).

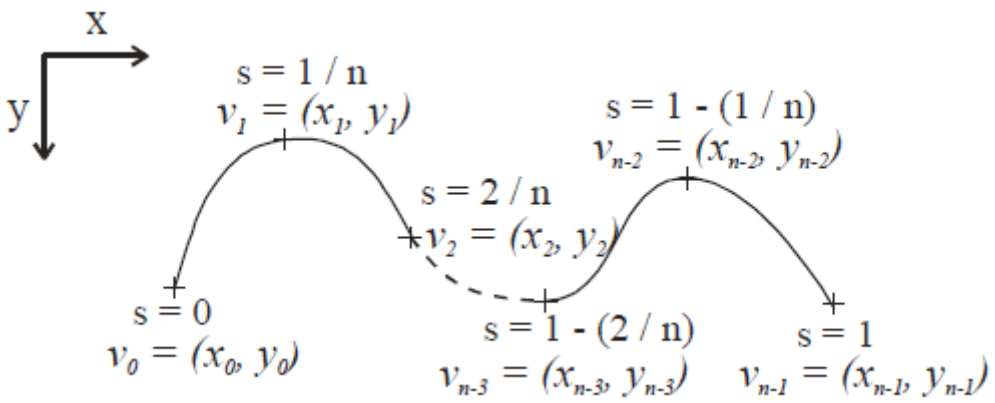


Fig. 1.1. *Exemples de coordonnées cartésiennes et abscisses curvilignes d'un contour actif.*

$C(s)$ est initialisée dans l'image Ω_I et évolue sous certaines contraintes selon ses directions normale et tangentielle jusqu'à ce qu'elle s'arrête sur les bords de l'objet recherché. Ainsi, le contour actif est une courbe déformable spatialement et temporellement [22] d'où son appellation actif du fait de son évolution au cours du temps, de son initialisation jusqu'à son état final (figure 1.2). Dans ce cadre, $C(s,t)$ représente alors la famille de courbes obtenues durant toute l'évolution du contour [1]. Cette évolution est régie par une fonctionnelle d'énergie qui lui est associée. Elle est composée de critères appartenant à deux principales familles :

a) Les critères énergétiques intrinsèques : ils prennent en compte les contraintes géométriques et structurantes sur C et vont permettre de contrôler la géométrie du contour. Ils sont généralement définis comme des énergies déterminées par la courbure de C, son périmètre ou encore son aire intérieure.
b) Les critères extrinsèques : appelés également critères d'attache aux données. Ils représentent les contraintes d'adéquation du modèle à l'image et vont permettre de guider la déformation du contour vers les zones d'intérêt puis de la stopper en prenant en compte les caractéristiques de l'image.

La forme générale de la fonctionnelle d'énergie associée à un contour actif est donnée par [1] :

$$E\big(v(s,t)\big) = \int_0^1 (E_{interne} + \lambda E_{ext})ds \qquad (1.2)$$

L'énergie interne $E_{interne}$ est déterminée par les critères géométriques intrinsèques de $v(s,t)$ et l'énergie externe $E_{externe}$ (qui se compose de $E_{image}(v) + E_{contexte}(v)$) est déterminée par les critères extrinsèques d'attache aux données, λ, un coefficient de pondération entre les deux énergie.

Le principe du contour actif est alors, l'évolution de la courbe d'une position initiale par minimisation successive de $E\big(v(s,t)\big)$ jusqu'à atteindre un état stable représenté par une courbe dont l'énergie sera un minimum de $E\big(v(s,t)\big)$ [1].

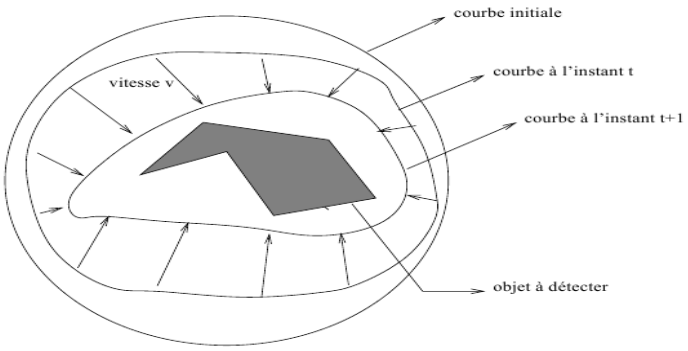


***Fig. 1.2**. Principe des contours actifs.*

Même s'il existe de nombreux modèles de contours actifs, ceux-ci suivent tous la même forme générale, c'est-à-dire, qu'ils sont définis par une fonctionnelle d'énergie associée à la courbe et ont un mode de représentation. La fonctionnelle d'énergie définira le comportement géométrique du modèle ainsi que les données de l'image dont il se servira pour évoluer. La représentation déterminera la manière dont le contour sera implémenté et influencera son comportement durant l'évolution. Alors qu'il existe presque autant de fonctionnelles d'énergie que de modèles de contours actifs, seules deux représentations différentes sont principalement utilisées (figure 1.3). La première, qualifiée de formulation Lagrangienne, représente les contours actifs de manière explicite [1] (les contours sont le plus souvent paramétriques), cette approche sera développée en détail dans le chapitre suivant. Dans ce chapitre nous développerons la seconde représentation, qualifiée de formulation Eulérienne, et qui représente les contours actifs de manière implicite [2].

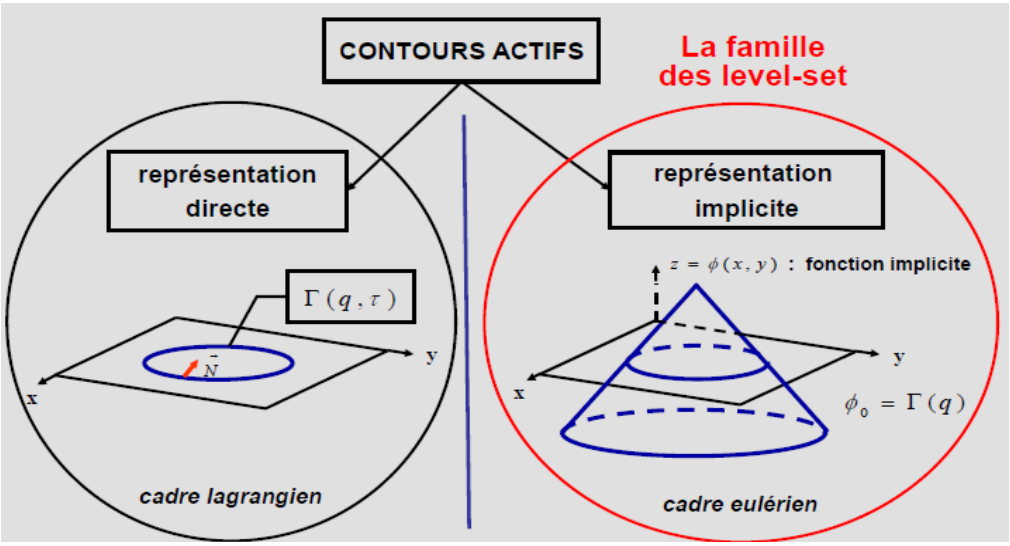


***Fig. 1.3.** Arbre généalogique des contours actifs.*

Alors que l'approche Lagrangienne repose sur le fait de déformer un modèle dans un espace jusqu'à un état final, le principe de l'approche Eulérienne est de déformer l'espace entier et non plus un sous-ensemble. Les déformations du modèle seront alors définies implicitement au travers des déformations de l'espace tout entier. Cette approche a été adaptée dans le cadre de la segmentation dans [21] sous le nom d'approche par ensembles de niveaux (*level set*) [1].

Les modèles implicites de contours actifs ont été proposés initialement et indépendamment par Caselles et *al.* [24] et Malladi et *al.* [25]. Ces modèles sont basés sur la théorie d'évolution de courbe (d'où vient l'appellation: les modèles géométriques) et la méthode des *level set*. Les courbes et surfaces évoluent en utilisant uniquement les mesures géométriques, résultant d'une évolution qui est indépendante de la paramétrisation. Ce qui permet aux courbes et surfaces d'être représentées implicitement comme un ensemble de niveaux d'une fonction de dimension supérieure. Par conséquent, les changements de topologie peuvent être traités automatiquement. Pour cela, et dans la partie suivante, nous allons définir le concept fondamental de la théorie d'évolution de courbe [11].

1.3 Théorie d'évolution de courbe

Le but de la théorie d'évolution de courbes est d'étudier la déformation des courbes utilisant uniquement les mesures géométriques comme la normale unitaire et la courbure par opposition aux quantités qui dépendent de paramètres telle que la dérivée d'une courbe paramétrique [11].

Soit une courbe mobile $C(s,t) = [X(s,t), Y(s,t)]$, avec s un paramétrage (l'abscisse curviligne) et t est le temps, sa normale unitaire intérieure est notée N et sa courbure k. L'évolution de la courbe le long de sa direction normale peut être caractérisée par l'équation aux dérivées partielles (EDP) suivante :

$$\frac{\partial C}{\partial t} = V(k)\, N \tag{1.3}$$

Où $V(k)$ est appelée fonction de vitesse, étant donné qu'elle détermine la vitesse de l'évolution de courbe.

Les déformations de courbe les plus étudiées dans la théorie d'évolution de courbe sont la déformation de courbure et la déformation constante [11].

La déformation de courbure est donnée par l'équation dite équation géométrique de la chaleur [11] :

$$\frac{\partial C}{\partial t} = \alpha k\, N \tag{1.4}$$

Où α est une constante positive et k la courbure qui permet de maintenir la régularité des contours.

Les courbes, évoluant selon l'équation (1.4), possèdent des propriétés de lissage. Malheureusement, sur le long terme, toute l'information est perdue alors que les contours se resserrent et finissent par disparaître [1]. L'utilisation de la déformation de courbure a un effet similaire à l'utilisation de la force interne élastique des contours actifs paramétriques, qu'on verra dans le chapitre suivant [11].

La déformation constante est donnée par l'équation suivante :

$$\frac{\partial C}{\partial t} = V_0\, N \tag{1.5}$$

Où V_0 est un coefficient déterminant la vitesse et la direction de déformation. La déformation constante joue le même rôle que la force de pression (force ballon) dans les contours actifs paramétriques [11].

Les propriétés de la déformation de courbure et de la déformation constante sont duales. La déformation de courbure supprime les points isolés en lissant la courbe, tandis que la déformation constante peut créer des singularités dans une courbe initialement lisse.

L'idée de base des contours actifs géométriques est de coupler la vitesse de déformation (utilisant la courbure et / ou la déformation constante) avec les données de l'image, de telle sorte que l'évolution de la courbe s'arrête aux contours de l'objet. Cette évolution est implémentée par la méthode des ensembles de niveaux (*level set*) que nous allons voir dans la section qui suit.

1.4 Ensembles de niveaux

Le principe général de la représentation en ensembles de niveaux est de décrire un objet de dimension n comme un niveau particulier d'une fonction de dimension $n+1$. L'évolution de l'objet de dimension n (i.e. le contour actif) ne sera non plus calculée directement, mais déduite de celle de la fonction de dimension supérieure, appelée fonction d'ensembles de niveaux ou *level set* [1].

En d'autres termes, la méthode des *level set* est un cadre de travail analytique œuvrant sur l'évolution géométrique d'objets. Au lieu d'employer une représentation Lagrangienne classique pour décrire les géométries, la méthode des *level set* les décrit à travers une fonction

scalaire définie sur une grille fixe. L'équation d'évolution, formalisée sous forme d'une EDP, est contrainte par un champ de vitesse imposé dans le sens de la normale au contour. Ce champ est construit de manière à attirer le modèle vers les objets à extraire dans l'image sous contraintes de régularisation géométrique. La normale et la courbure du contour en chaque point sont facilement définies à partir des propriétés différentielles géométriques de cette représentation [12].

1.4.1 Représentation

Soit $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}$ la fonction d'ensembles de niveaux appliquée à la segmentation par contour actif d'une image en deux dimensions. Cette fonction peut être représentée comme une fonction en trois dimensions définie sur les deux dimensions de l'image, plus une hauteur correspondant à différents niveaux de plans. En d'autres termes, à chaque pixel x = $(x, y)^T$ de l'image, la fonction $\phi(x)$ fait correspondre un niveau dans la représentation en trois dimensions. Le contour recherché sera alors donné par l'intersection de ϕ et du plan défini par ϕ = 0. Initialement, le contour $C(s, 0)$ est donné par le niveau zéro initial de $\phi(x, 0)$ (figure 1.4). Cette relation étant conservée avec le temps, nous avons:

$$\forall\, x \in \mathbb{R}^2, \forall\, s \in [a, b], C(s, t) = \{x \in \mathbb{R}^2 | \, \phi\,(x, t) = 0\} \qquad (1.6)$$

Ainsi, pour tout pixel x appartenant au contour $C(s, t)$, défini selon l'abscisse curviligne s, $\phi\,(x, t) = 0$.

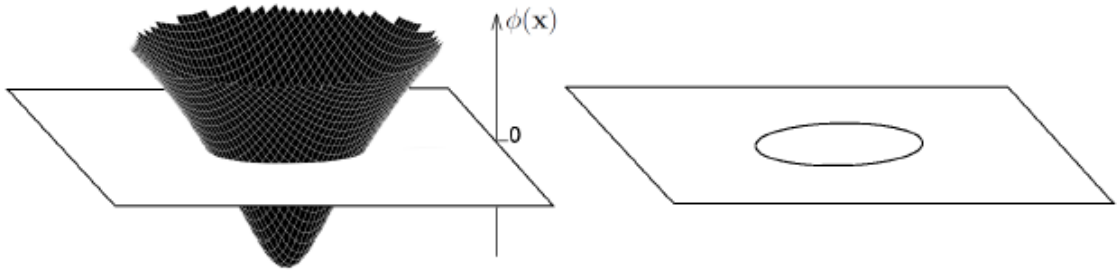


Fig. 1.4. *Représentation de $\phi(x, t)$ et de son intersection avec le plan de niveau zéro : la forme en trois dimensions représentée sur l'image de gauche est la fonction d'ensembles de niveaux $\phi(x, t)$ définie pour tout pixel x de l'image. Le contour $C(s, t)$ représenté sur l'image de droite est déterminé par l'intersection de $\phi(x, t)$ et du plan de niveau zéro.*

En appliquant le principe de l'approche Eulérienne, ce n'est plus C qui va se déformer mais ϕ. L'évolution de C sera alors directement donnée par celle du niveau zéro de ϕ. L'avantage d'utiliser le niveau zéro (level zéro), est qu'un contour peut être défini comme les

frontières entre une surface positive et une surface négative. Ainsi le contour peut être identifié par la vérification du signe de la fonction *level set* $\phi(x, t)$ [8].

Avant de s'intéresser à l'évolution de la fonction d'ensembles de niveaux, il est nécessaire de déterminer sa valeur initiale.

La fonction ϕ est généralement initialisée en chaque pixel de l'image comme la distance Euclidienne au point de contour initial le plus proche. Ces distances sont signées par convention comme suit [1] (figure 1.5):

$$\begin{cases} \phi(x,t) > 0 & \text{si s est à l'exterieur de la courbe} \\ \phi(x,t) < 0 & \text{si s est à l'interieurde la courbe} \\ \phi(x,t) = 0 & \text{si s appartien à la courbe} \end{cases} \quad (1.7)$$

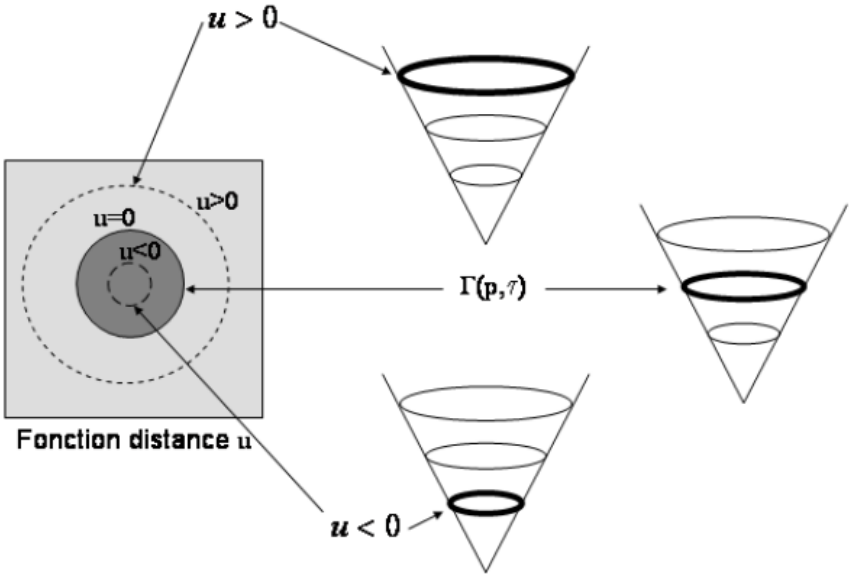


Fig.1.5. *fonction de distance signée* ϕ *(dans cette figure appelée u) au contour C (appelé dans la figure* Γ*).*

Ainsi, initialement ϕ est la carte des distances signées au contour et sa représentation en trois dimensions, dans le cadre d'une initialisation d'un contour circulaire, prend une forme de cône tel qu'il est montré à la figure 1.6.

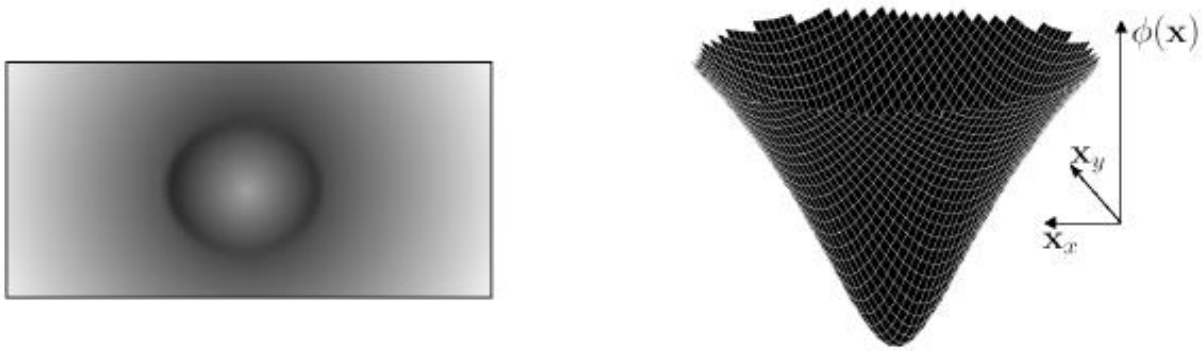


Fig. 1.6. *Initialisation de ϕ pour chaque pixel x, ϕ (x, 0) est définie comme la distance signée au contour. L'image de gauche représente la valeur absolue de ϕ lors de l'initialisation pour chaque pixel de l'image (normalisée entre 0 et 255), l'image de droite illustre la forme de cône alors prise par ϕ.*

Les figures 1.7 et 1.8 illustrent les différentes valeurs de ϕ lors de l'initialisation selon plusieurs formes initiales. Pour plus de détails sur les différentes initialisations de la fonction ϕ, nous conseillons le lecteur de voir [26].

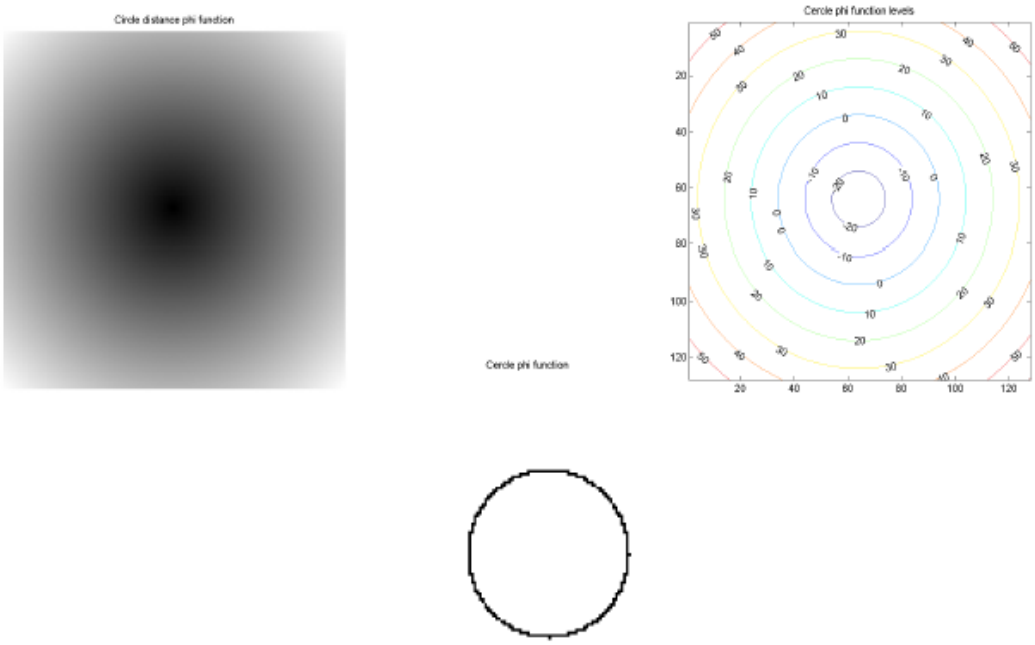


Fig. 1.7. *Illustration les différentes valeurs de ϕ lors de l'initialisation dans le cas d'un cercle.*

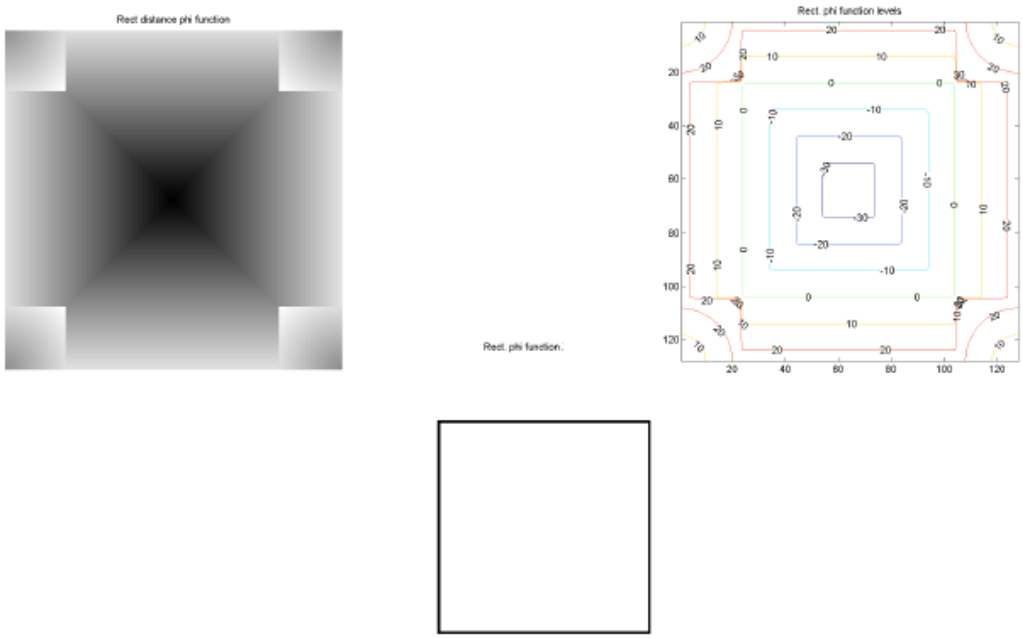


Fig. 1.8. *Illustre les différentes valeurs de ϕ lors de l'initialisation dans le cas d'un rectangle.*

Nous avons vu la représentation des ensembles de niveaux et l'initialisation du contour en utilisant cette approche, nous allons maintenant définir l'équation d'évolution.

1.4.2 Evolution

Un contour représenté de manière implicite est directement défini par une équation d'évolution, appelée également équation de mouvement. Celle-ci définit les déformations subies par la courbe C(s, t) selon ses directions normale et tangentielle (figure 1.9) lors de la minimisation de son énergie E(C(s, t)) selon l'équation :

$$\frac{\partial C(s,t)}{\partial t} = F_{N(s,t)}\, N(s,t) + F_{T(s,t)}\, T(s,t) \qquad (1.8)$$

Avec: $N(s,t)$ et $T(s,t)$ sont respectivement l'expression du vecteur normal et du vecteur tangentiel à la courbe $C(s,t)$ au point d'abscisse curviligne s [1]. Le déplacement appliqué à chaque point est exprimé dans le repère de Frenet (T,N), séparé en deux composantes : les fonctions $F_{N(s,t)}$ et $F_{T(s,t)}$ appelées fonctions vitesses, régissent respectivement l'évolution de la courbe selon ses directions normale et tangentielle. Il est énoncé dans [27] que seule la composante normale affecte la géométrie du contour, la composante tangentielle n'agissant que sur la répartition des points le long de la courbe, autrement dit, n'ayant d'incidence que

sur sa paramétrisation, la déformation du contour a lieu uniquement dans sa direction normale [13]. L'équation (1.8) peut donc être simplifiée :

$$\frac{\partial C(s,t)}{\partial t} = F_{N(s,t)}\, N(s,t) \tag{1.9}$$

Ainsi, seule la fonction vitesse définie selon la normale F_N (désormais notée F par soucis de simplification) influence la déformation. De manière similaire à la fonctionnelle d'énergie, *F* sera composée de termes géométriques et de termes d'attache aux données.

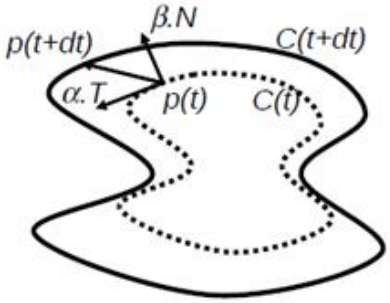


Fig. 1.9. *Illustration des deux composantes tangentielle et normale*

Les déformations de la fonction d'ensembles de niveaux sont déterminées grâce à l'équation (1.9) d'évolution du modèle de contour actif. Afin de déterminer l'équation d'évolution de la fonction d'ensembles de niveaux, il est nécessaire d'exprimer la dérivée spatiotemporelle de l'équation (1.6). En utilisant, dans un premier temps, la dérivée spatiale, nous obtenons :

$$\nabla_s C(s,t) \cdot \nabla_{\mathrm{x}} \phi(C(s,t),t) = 0 \tag{1.10}$$

$\nabla_s C(s,t)$ étant la dérivée de $C(s,t)$ par rapport à son abscisse curviligne s. Elle représente l'expression de la direction de la tangente à $C(s)$ au point s. Ainsi, comme le produit scalaire $\nabla_s C \cdot \nabla_{\mathrm{x}} \phi$ est nul, alors $\nabla_{\mathrm{x}} \phi$ et le vecteur normal au contour N sont colinéaires. N s'exprime donc en fonction de ϕ comme :

$$\mathrm{N} = -\frac{\nabla \phi}{\|\nabla \phi\|} \tag{1.11}$$

La dérivée temporelle de l'équation (1.6) nous donne :

$$\frac{\partial(\phi(C(s,t),t))}{\partial t} + \nabla \phi(C(s,t),t) \cdot \frac{\partial(C(s,t),t)}{\partial t} = 0 \tag{1.12}$$

En remplaçant l'équation (1.9) dans l'équation (1.12) on obtient alors :

$$\frac{\partial(\phi(C(s,t),t))}{\partial t} = -\nabla\phi(C(s,t),t)\cdot F(s,t)N \tag{1.13}$$

Qui devient d'après (1.11) :

$$\frac{\partial(\phi(C(s,t),t))}{\partial t} = F(s,t)\,\|\nabla\phi(C(s,t),t)\| \tag{1.14}$$

La fonction vitesse $F(s,t)$ dépend :

- Du type de déformation vu au 1.3 : courbure et/ou constante.
- Des données de l'image.

Si la fonction vitesse F est définie sur toute l'image, l'équation (1.14) peut être étendue à tout le domaine de définition de l'image Ω_I :

$$\frac{\partial(\phi(\mathrm{x},t))}{\partial t} = F(\mathrm{x},t)\,\|\nabla\phi(\mathrm{x},t)\| \quad \forall\, \mathrm{x} \in \Omega_I \tag{1.15}$$

Cette extension ne conserve pas la fonction distance. Il est donc nécessaire de réinitialiser régulièrement les valeurs de ϕ en fonction de la position de son niveau zéro (et donc de $C(s,t)$). Pour cela, il est possible d'utiliser un algorithme de calcul de distance d'un point à un objet du type transformation de distances [28].

1.4.3 Mise en œuvre numérique

L'implémentation d'un contour actif à l'aide des ensembles de niveaux nécessite de nombreuses discrétisations pouvant parfois s'avérer complexes. Dans cette partie nous allons tenter d'éclaircir certains calculs.

Comme il est précisé à l'équation (1.11), le vecteur normal à la courbe N s'exprime comme le gradient normalisé de la fonction d'ensembles de niveaux. La définition de $\nabla\phi(\mathrm{x},t)$ est donné par :

$$\nabla\phi(\mathrm{x},t) = \begin{pmatrix}\phi_x\\ \phi_y\end{pmatrix} \tag{1.16}$$

Avec :

$$\phi_x = \frac{\partial\phi(x,y)}{\partial x} \quad \text{et} \quad \phi_y = \frac{\partial\phi(x,y)}{\partial y} \tag{1.17}$$

ϕ_x est l'approximation centrée de la dérivée partielle du premier ordre de la fonction d'ensembles de niveaux ϕ suivant la coordonnée x, et ϕ_y son équivalent suivant la coordonnée y. Les expressions de ϕ_x et ϕ_y sont données par :

$$\phi_x = \frac{\phi(x+1,y)-\phi(x-1,y)}{2\Delta x} \tag{1.18}$$

$$\phi_y = \frac{\phi(x,y+1)-\phi(x,y-1)}{2\Delta y} \tag{1.19}$$

Ainsi, après discrétisation, la norme du gradient de ϕ devient :

$$\|\nabla\phi(x,t)\|^2 = \phi_x{}^2 + \phi_y{}^2 \tag{1.20}$$

Généralement, afin de garantir un contour lisse, la fonction vitesse F des contours actifs implémentés à l'aide des ensembles de niveaux possèdent un terme de courbure k. L'expression de k se détermine à l'aide de l'opérateur de divergence du gradient normalisé de ϕ :

$$k = div\frac{\nabla\phi(x,t)}{\|\nabla\phi(x,t)\|} = \frac{\phi_{xx}\phi_y^2 - 2\phi_{xy}\phi_x\phi_y + \phi_{yy}\phi_x^2}{\left(\phi_x^2+\phi_y^2\right)^{3/2}} \tag{1.21}$$

Où: ϕ_{xx} , ϕ_{yy} et ϕ_{xy} sont les expressions des dérivées centrées d'ordre deux de la fonction ϕ selon chaque coordonnée, définies comme :

$$\phi_{xx} = \frac{\partial^2\phi(x,t)}{\partial x^2} = \frac{\phi(x+1,y) - 2\phi(x,y) + \phi(x-1,y)}{4\Delta x^2} \tag{1.22}$$

$$\phi_{yy} = \frac{\partial^2\phi(x,t)}{\partial y^2} = \frac{\phi(x,y+1) - 2\phi(x,y) + \phi(x,y-1)}{4\Delta y^2} \tag{1.23}$$

$$\phi_{xy} = \frac{\partial^2\phi(x,t)}{\partial x\partial y} = \frac{\phi(x+1,y+1) - \phi(x+1,y-1) - \phi(x-1,y+1) + \phi(x-1,y-1)}{4\Delta x\Delta y} \tag{1.24}$$

A l'instar des différents termes présentés ici, l'équation d'évolution finale (1.15) nécessite également une discrétisation afin de permettre une évolution itérative du modèle. Celle-ci est réalisée en faisant intervenir un pas de temps Δt :

$$\phi(\mathrm{x})^{t+1} = \phi(\mathrm{x})^{t} + \Delta t\, F(\mathrm{x})^{t} \|\nabla\phi(\mathrm{x})^{t}\| \tag{1.25}$$

Nous allons maintenant nous intéresser à la fonction de vitesse, car la plupart des recherches concernant les contours actifs géométriques sont portées sur la conception de la fonction de vitesse.

1.4.4 Fonction de vitesse

La formulation des contours actifs implicites originaux (contours actifs géométriques), proposée par Caselles et *al.* et Malladi et *al.* est donnée comme suit :

$$\frac{\partial\phi}{\partial t} = g(I)(k + V_0)\, \|\nabla\phi\| \tag{1.26}$$

Avec :

$$g(I) = \frac{1}{1+|\nabla(G_\sigma * I)|} \tag{1.27}$$

Le terme constant V_0 accélère et maintient l'évolution des contours par la minimisation de la surface fermée permettant, ainsi, à la courbe de se dilater si la valeur de V_0 est négative, et se contracter si sa valeur est positive, avec la même force en chaque point du contour. L'évolution de la courbe est couplée avec les données de l'image représentées par le terme d'arrêt g prenant des valeurs faibles aux endroits d'intérêt [1], car lorsque le point considéré se trouve loin du bord de l'objet, le gradient est faible et la valeur de g est proche de 1, par contre, sur le bord de l'objet, cette valeur tend vers 0 [8]. Ce terme est efficace lorsque l'image est bien contrastée. Toutefois, lorsque les contours de l'objet ne sont pas très visibles ou incomplets, le contour géométrique à tendance à ne pas s'arrêter sur ces derniers et à les dépasser car g ne fait que ralentir la courbe près des frontières de l'objet plutôt que de la stopper, et une fois que la courbe dépasse le contour de l'objet, il ne sera pas possible de l'attirer en arrière afin qu'elle entoure correctement l'objet [1]. De plus, dans la formulation (1.26), la vitesse du contour actif ne dépend en rien des données de l'image, mais seulement des caractéristiques géométriques de la courbe. L'image n'intervient qu'au niveau de la fonction d'arrêt du contour actif [8].

C'est ainsi, que sont apparus les contours actifs géodésiques qui apportent une solution aux problèmes des contours actifs géométriques et d'une manière élégante, font un lien entre approche par évolution des courbes et approche par minimisation d'énergie. L'équation d'évolution du modèle de contour actif géodésique est alors [1] :

$$\frac{\partial C(s,t)}{\partial t} = \Big(g\big(I(s)\big)(k + V_0) \; - \nabla g \cdot N(s,t)\Big) N(s,t) \tag{1.28}$$

Où k représente la courbure du modèle.

Le contour actif est représenté implicitement, et est implémenté à l'aide des ensembles de niveaux. En considérant $F(\mathrm{x},t) = g\big(I(\mathrm{x})\big)(k + V_0) \; - \nabla g \cdot N(\mathrm{x},t)$, l'expression de l'équation d'évolution appliquée à la fonction d'ensembles de niveaux devient :

$$\frac{\partial\big(\phi(\mathrm{x},t)\big)}{\partial t} = \Big(g\big(I(\mathrm{x})\big)(k + V_0) \; - \nabla g \cdot \frac{\nabla\phi}{\|\nabla\phi\|}\Big) \|\nabla\phi\| \tag{1.29}$$

D'où :

$$\frac{\partial\big(\phi(\mathrm{x},t)\big)}{\partial t} = g\big(I(\mathrm{x})\big)\,(k + V_0) \; \|\nabla\phi\| + \nabla g \cdot \nabla\phi \tag{1.30}$$

La fonction vitesse résultante comporte un terme d'arrêt supplémentaire $\nabla g \cdot \nabla\phi$, qui ne figurait pas dans le modèle géométrique correspondant à l'équation (1.26). Ce terme permet au modèle géodésique de stabiliser le contour sur les bords de l'objet [8], c'est-à-dire de retenir le modèle s'il dépasse le contour de l'objet [11]. Un exemple de résultat obtenu avec ce type de contour est fourni dans la figure 1.10.

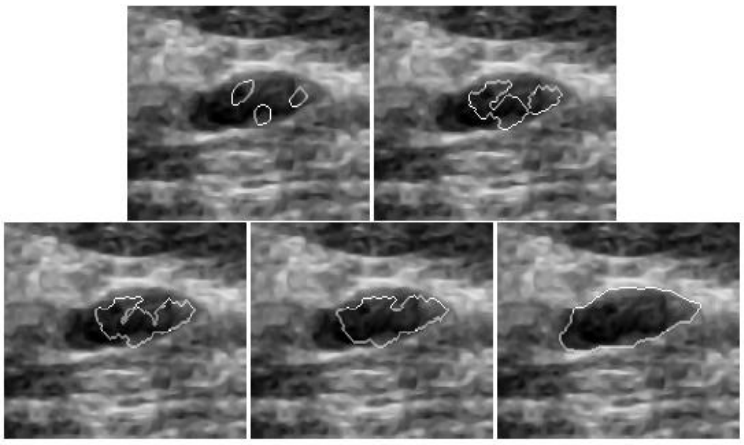


Fig. 1.10. *Exemple de résultats obtenus [11].*

Cette approche possède l'avantage de s'affranchir du réglage des nombreux paramètres présents jusque là dans les précédents modèles. Contrairement aux termes d'attache aux

données classiquement utilisés, le terme ∇g ne se contente pas de stopper le contour sur les zones possédant de forts changements d'intensités, il va également l'attirer vers celles-ci. Ces travaux furent parmi les premiers à unifier les approches énergétiques (où le contour actif est défini par une fonctionnelle d'énergie) et les approches par ensembles de niveaux (où le contour actif était jusque là dit géométrique, défini directement par son équation d'évolution). Cependant, les contours actifs géodésiques possèdent quelques inconvénients tels que, la sensibilité au bruit (l'image devra donc être filtrée au préalable) et la qualité de convergence dépend de la fonction d'arrêt g .

D'autres modèles ont pu voir le jour. Parmi le plus répondu est sans doute celui de Chan et Vese [29]. Ce modèle de contour actif basé région peut être considéré comme un cas particulier du modèle de Mumford-Shah [30]. L'approche est définie comme un problème de minimisation d'énergie afin de déterminer la partition optimale de deux régions séparées par un contour C (figure 1.11). Les auteurs définissent c_1 comme la moyenne des intensités de niveaux de gris dans la région intérieure au contour C, et c_2 son équivalent pour la région extérieure au contour. Des critères intrinsèques sont également ajoutés afin de garantir un contour lisse et régulier tels que la longueur de la courbe C ou la taille de sa région intérieure. La fonctionnelle d'énergie attachée au contour est donnée par [1] :

$$E(c_1, c_2, C) = \mu \cdot Longueur\,(C) + v\,Aire(C) + E_1(c_1, C) + E_2(c_2, C) \qquad (1.31)$$

Avec :

$$E_1(c_1, C) = \lambda_1 \int_{Interieur(C)} |I(\mathrm{x}) - c_1|^2 d\mathrm{x} \qquad (1.32)$$

$$E_2(c_2, C) = \lambda_2 \int_{Exterieur(C)} |I(\mathrm{x}) - c_2|^2 d\mathrm{x} \qquad (1.33)$$

Avec $\mu, v, \lambda_1, \lambda_2$ des coefficients qui permettent de pondérer l'influence de chaque terme. $\mu, v \geq 0$, $\lambda_1, \lambda_2 > 0$. E_1 et E_2 sont les critères d'attache aux données utilisés pour guider la courbe vers les frontières de l'objet à segmenter en utilisant c_1 et c_2, les moyennes des intensités de niveaux de gris respectivement à l'intérieur et à l'extérieur du contour C. Dans un schéma idéal, lorsque la courbe C se place sur la frontière de l'objet à segmenter, alors $E_1 \approx E_2 \approx 0$. Ainsi, minimiser l'équation (1.31) revient à déterminer la meilleure partition de l'image en deux régions homogènes séparées par la courbe C. L'objet à segmenter correspondra alors à une des deux régions et son contour sera la courbe C.

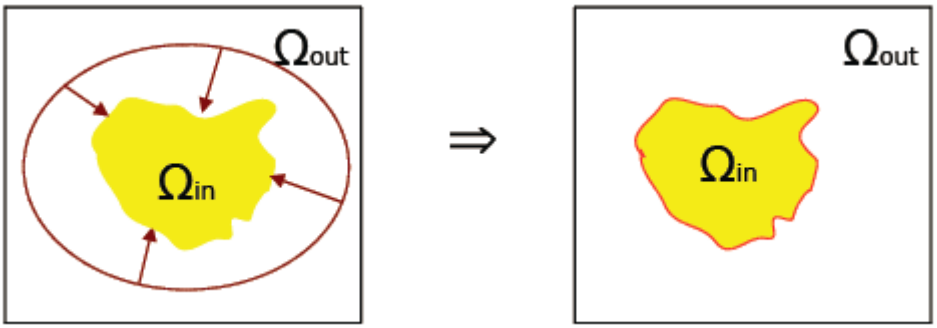


Fig.1.11. *Illustration de deux régions séparées par un contour.*

Afin de représenter leur modèle de manière implicite les auteurs introduisent dans l'équation (1.31) des versions modifiées de la fonction de Heavyside $H(z)$ et de la mesure de Dirac $\delta_0(z)$ définies par [1] :

$$H_\epsilon(z) = \frac{1}{2}\left(1 + \frac{2}{\pi}\, arctan\left(\frac{z}{\epsilon}\right)\right) \qquad \delta_\epsilon(z) = \frac{d}{dz}\, H_\epsilon(z) \tag{1.34}$$

L'introduction de $H_\epsilon(z)\ et\ \delta_\epsilon(z)$ dans la fonctionnelle d'énergie permet aux auteurs de simplifier les intégrales de région et d'exprimer chaque terme de l'équation (1.31) en fonction de la fonction d'ensembles de niveaux $\phi(x)$. L'énergie $E(c_1, c_2, \phi)$ du modèle formulé de manière implicite devient alors :

$$\begin{aligned} E_\epsilon(c_1, c_2, C) = \mu \int_\Omega\ \delta_\epsilon\big(\phi(x)\big)|\nabla\phi(x)|\, dx + \nu \int_\Omega\ H_\epsilon\big(\phi(x)\big)dx \\ + \lambda_1 \int_w |I(x) - c_1(\phi)|^2\, H_\epsilon\big(\phi(x,y)\big)\, dx \\ + \lambda_2 \int_w |I(x) - c_2(\phi)|^2 (1 - H_\epsilon\big(\phi(x,y)\big))\, dx \end{aligned} \tag{1.35}$$

avec: $c_1(\phi)$ et $c_2(\phi)$ sont alors données par la moyenne des intensités des pixels en fonction du signe de ϕ :

$$\begin{cases} c_1(\phi) = moyenne\big(I(x)\big)\ pour\ \phi(x) \geq 0 \\ c_2(\phi) = moyenne\big(I(x)\big)\ pour\ \phi(x) < 0 \end{cases} \tag{1.36}$$

En fixant c_1 et c_2 dans l'équation (1.36) et en minimisant $E_\epsilon(\phi)$, l'équation d'Euler-Lagrange du modèle est alors déterminée :

$$\frac{\partial\phi}{\partial t} = \delta_\epsilon(\phi)\left[\mu\, div\left(\frac{\nabla\phi}{|\nabla\phi|}\right) - v - \lambda_1(I - c_1)^2 + \lambda_2(I - c_2)^2\right] \tag{1.37}$$

Ce modèle s'affranchit donc de toute information de frontière et n'utilise que les moyennes des intensités de niveaux de gris dans les régions intérieure et extérieure comme critères d'attache aux données. Cette approche a été étendue à la segmentation d'images multi-canaux dans [29]. Ce nouveau modèle de contour actif basé région s'est montré très efficace pour segmenter des images possédant des objets ne pouvant pas être séparés par des gradients d'intensité et a été utilisé dans de nombreux domaines [31]. En revanche, le terme d'attache aux données basé sur les moyennes d'intensités reste limité pour une application sur des images texturées. En effet deux textures différentes peuvent avoir des moyennes d'intensités sensiblement égales, ce qui rend l'utilisation du modèle de Chan et Vese délicate. Cependant, nous allons implémenter ce modèle de contours actifs, pour sa simplicité et sa puissance de changement de topologie, afin qu'il soit une référence pour nos différents tests.

Pour plus d'informations sur les différents modèles de contours actifs implicites le lecteur peut trouver pas mal de travaux dans la littérature.

1.5 Avantages et inconvénients des level set

Le principal avantage des contours actifs représentés implicitement est leur capacité à gérer, naturellement, les changements de topologie. En effet, l'intérêt de la représentation Eulérienne du modèle est que ce n'est plus le contour qui se déforme mais la fonction d'ensembles de niveaux. Ainsi, bien qu'un niveau change de topologie, la fonction en elle même reste dans la topologie de départ (figure 1.12, 1.13 et 1.14). Donc ce n'est pas le modèle déformé qui change de topologie à proprement parler, mais simplement le sous-ensemble qui est observé (le contour actif) [1].

D'autres avantages enrichissent ce type de contour actif, ils seront [14] :

- Grandeurs géométriques intrinsèques (normales entrante/sortante, courbure) faciles a calculer.
- Les résultats obtenus sont très précis.
- Extension à la 3D simple: il suffit d'ajouter une coordonnées a l'équation d'évolution de la fonction ϕ: on a alors un volume $\phi(x, y, z, t)$.
- Discrétisation de Φ avec une grille définit dans le domaine de l'image, i.e. valeurs possibles pour (x, y)
- Utilisation des méthodes numériques connues pour calculer les dérivées.

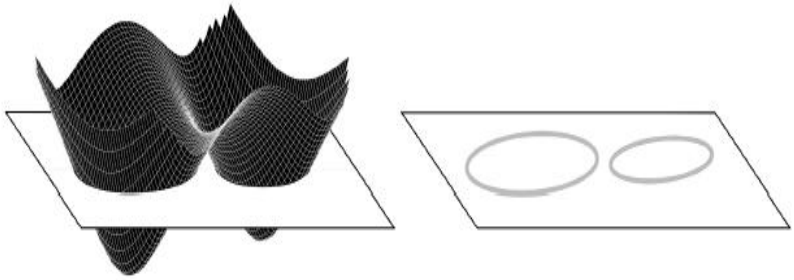


Fig. 1.12. *Illustration de la gestion des changements de topologie de l'implémentation en ensembles de niveaux : la fonction ϕ se déforme et, bien qu'elle ne change pas de topologie directement, le sous-ensemble observé (son intersection avec le plan de niveau zéro) change de topologie en donnant deux contours distincts.*

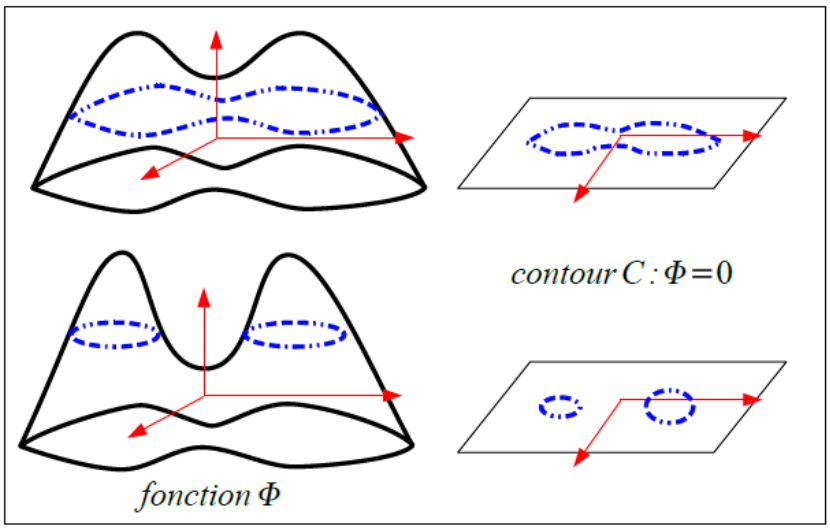


Fig. 1.13. *visualisation de la fonction ϕ.*

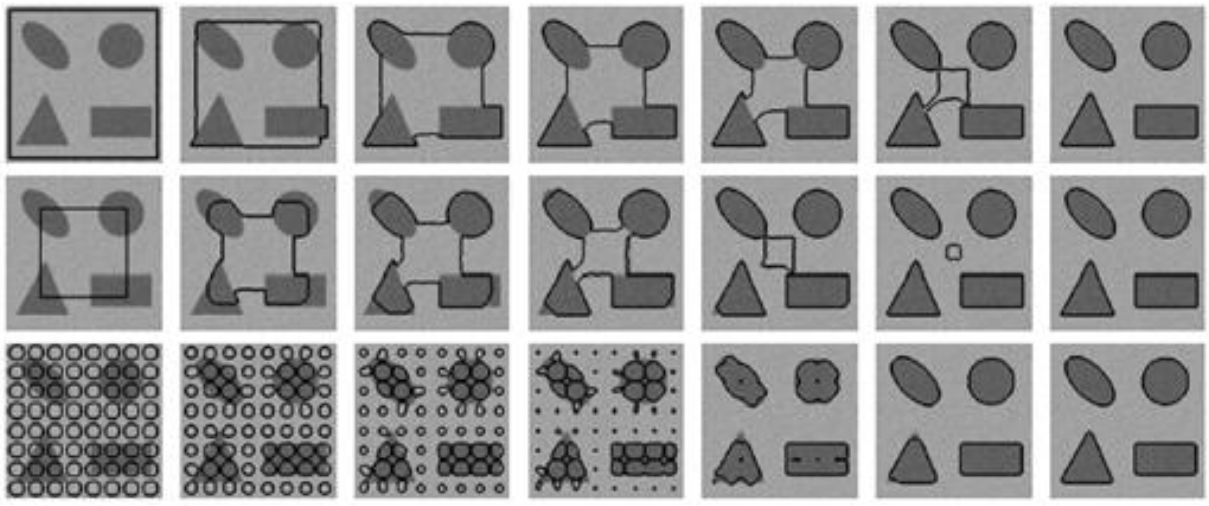


Fig.1.14. *plusieurs initialisation du contour initial qui convergent tous vers le même résultat [50].*

L'inconvénient majeur de cette représentation est la complexité algorithmique de son évolution. En effet la représentation implicite d'un contour actif accroît la complexité de la méthode en $O(n^2)$, où n est la taille de l'image. Un contour actif représenté de manière implicite sera donc plus lent qu'un paramétrique. Cela est notamment dû à la résolution des EDP et au calcul de k qui sont réalisés pour chaque pixel de l'image à chaque itération [1]. D'autres inconvénients dérangent ce modèle:

- la construction de la fonction initiale $\phi(x, t) = 0$ doit être faite de manière à ce que son niveau zéro corresponde a la position initiale du contour.
- L'équation d'évolution n'est dérivée qu'au *level set* zéro, la fonction vitesse *v* n'est donc pas définie (en générale) pour les autres *level sets.*
- La déformation constante peut causer la formation de coins saillants sur le modèle initialement lisse. Ce coin peut perturber les déformations successives, puisque la définition de la normale devient ambigüe.
- Parfois il est nécessaire de recalculer la fonction distance par rapport au niveau zéro.

Plusieurs solutions ont été trouvées pour ces inconvénients. En effet, en raison de la complexité algorithmique des *level set*, qui reste le problème majeur, de nombreux algorithmes d'accélération ont été développés : la méthode Fast Marching [32], l'implémentation en bande étroite [51], l'algorithme Hermes [18] ou récemment l'algorithme de Shi et Karl permettant de s'affranchir de la résolution des EDP [52]. Toute fois, ce problème majeur de ces contours actifs reste toujours posé.

1.6 Conclusion

Dans ce chapitre, nous avons tenté de dégager une structure générale pour le contour actif, puis de présenter de manière exhaustive l'un de ses modèles, à savoir le contour actif représenté implicitement. Nous avons pu voir que la représentation implicite est basée sur la théorie d'évolution de courbes, et implémentée par la méthode des ensembles de niveaux. L'évolution de la courbe se faisant uniquement en utilisant les mesures géométriques, donc indépendamment de la paramétrisation, permet à la courbe d'être représentée implicitement comme un ensemble de niveau d'une fonction de dimension supérieure, ce qui lui confère son aptitude à changer de topologie naturellement et donc à détecter de manière précise plusieurs objets.

Cependant, en raison de la complexité algorithmique de son évolution, le temps d'exécution reste très lent malgré les quelques méthodes d'accélération proposées. Ainsi, une étude détaillée de l'autre représentation, à savoir la représentation paramétrique, est intéressante.

2.1 Introduction

Le mode de représentation explicite ou paramétrique, est le premier apparu dans la littérature. Il fut introduit dans [35] pour le premier modèle de contour actif le *″ snake ″*. Les contours actifs paramétriques représentent les courbes et les surfaces explicitement dans leurs formes paramétriques au cours de la déformation. Cette représentation permet une interaction directe avec le modèle et peut conduire à une représentation compacte pour une mise en œuvre en temps réel rapide. La représentation paramétrique est qualifiée de formulation Lagrangienne qui considère le contour non plus, comme une entité statique, mais comme une entité dynamique qui évolue dans le temps. Cependant, le changement de topologie, telle que la fusion ou la séparation de contours, est difficiles contrairement aux modèles développés au chapitre précédent.

A cet effet, nous allons, dans ce chapitre, commencer par introduire la représentation paramétrique, puis, nous allons nous intéresser aux *snakes* et définir sa fonctionnelle d'énergie, son équation d'évolution, et les méthodes de discrétisation et d'implémentation utilisées. Enfin nous allons terminer ce chapitre par une analyse des caractéristiques du *snake*, et l'introduction aux multi-objets.

2.2 Représentation paramétrique

La forme la plus simple de représenter une courbe en deux dimensions est la forme explicite :

$$y = f(x) \tag{2.1}$$

Presque toutes les courbes simples peuvent être représentées en utilisant la forme explicite, mais pour des courbe plus complexes avec deux ou plusieurs valeurs de *y* correspondant à une valeur unique de *x*, la forme explicite échoue [6]. En outre, la forme explicite a des problèmes avec les courbes de modélisation parallèles à l'axe des y, puisque pour une telle courbe la pente tend vers l'infini [6]. Ces lacunes excluent l'utilisation de la forme explicite pour les courbes telles que les cercles, ellipses et autres coniques. L'équation de la courbe implicite bidimensionnelle est :

$$f(x,y) = 0 \tag{2.2}$$

peut être utilisée pour représenter n'importe quelle courbe, évitant ainsi les limites de la courbe explicite. Malheureusement, la forme inhérente de l'équation de la courbe implicite ne nous permet pas de calculer les points sur la courbe directement, nécessitant souvent la solution d'une équation non linéaire pour chaque point. Cela nous amène à considérer la forme paramétrique d'une courbe, qui, sous forme de vecteur, est spécifiée comme suit :

$$v\,(s) = \begin{bmatrix} x(s) \\ y(s) \end{bmatrix} \tag{2.3}$$

La courbe paramétrique n'a qu'un seul paramètre indépendant s dont on fait varier la valeur sur un certain intervalle (généralement [0,1]). En utilisant cette représentation, nous évitons les problèmes que les deux formes explicites et implicites ont [6]. Par exemple, nous pouvons avoir plusieurs valeurs de y pour une valeur unique de x, cela est facile à voir dans la forme paramétrique d'un cercle unité de centre à l'origine :

$$v\,(s) = \begin{bmatrix} x(s) \\ y(s) \end{bmatrix} = \begin{bmatrix} \cos s \\ \sin s \end{bmatrix} \quad \text{avec } s \in [0,2\pi] \tag{2.4}$$

La définition de la courbe C dans l'équation (1.1) correspond à une courbe continue dans le plan. Une image étant une grille discrète et régulière de pixels, C va être discrétisée et sous-échantillonnée selon son paramètre s. Le contour actif sera ainsi représenté par des points reliés entre eux par une notion de voisinage deux à deux. Les déformations successives devant être appliquées au contour seront déterminées uniquement pour les points et l'évolution complète du contour actif suivra alors celle des points [1]. La figure (2.1) illustre les étapes de construction de la représentation paramétrique discrète d'un contour actif.

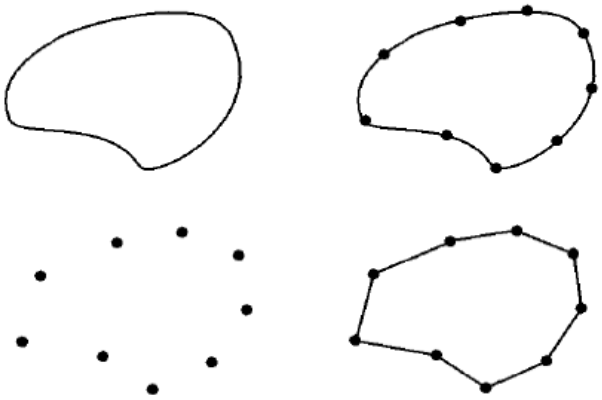


Fig. 2.1. *Principe d'échantillonnage de la représentation paramétrique d'un contour actif.*

Représenter un contour de manière paramétrique présente ainsi plusieurs autres avantages :

– les différentes opérations à réaliser sur l'ensemble du contour (calcul de déplacements, de caractéristiques, etc.) ne sont réalisées que sur les points de contrôle, ce qui entraine un gain de temps d'exécution important.

– les composantes intrinsèques telles que la courbure, la normale ou la tangente restent utilisables, car calculables grâce à la notion de voisinage deux à deux des différents points de contrôle.

Cependant, cette représentation possède aussi ses limites :

– le gain de temps de calcul apporté par la discrétisation amène en contrepartie une perte de précision sur la reconstruction du contour de l'objet à segmenter. Cela conduit à un dilemme pour choisir entre le temps de réponse du modèle et sa précision, lié au nombre de points. En effet, augmenter le nombre de points permettra d'améliorer la précision du contour mais entrainera alors une baisse de sa vitesse d'évolution.

– lorsque le contour a tendance à changer de taille de manière importante, un rééchantillonnage de la courbe devient nécessaire de manière à conserver une précision suffisante. Pour certains modèles, cela signifie recalculer un certain nombre de paramètres dépendant directement du nombre de points.

– ce type de représentation ne permet pas de gérer naturellement les changements de topologie du contour. Des algorithmes additionnels doivent être implémentés pour permettre à plusieurs contours de fusionner ou de se séparer afin, par exemple, de segmenter plusieurs objets, qui sera notre objectif dans ce travail.

Afin de bien comprendre tous cela, une étude détaillée des contours actifs se basant sur la représentation explicite est très importante. Pour cela, et dans la section suivante, nous allons développer les *snakes*.

2.3 Contour actif classique « *snake* »

Il est représenté comme un modèle élastique déformable contrôlé par une contrainte de continuité dont les mouvements de glissement lors de la déformation lui ont valu le nom de *snake* qui veut dire serpent. Historiquement, le *snake* a été introduits par Kass, Witkin et Terzopoulos en 1988 dans leur article « *Snakes*: Active Contour Models » et fait figure de référence en la matière, car il représente le premier modèle de contours actifs à être utilisé en traitement d'image pour la segmentation. Parmi les modèles paramétriques de contours actifs,

on peut considérer que les *Snakes* représente le premier exemple de structure de contour actif dont les propriétés intrinsèques (telles que la rigidité, l'élasticité, courbure) sont choisies et, ou modifiées pour améliorer la minimisation de l'énergie [2].

Les principales étapes d'utilisation du *snake* pour une application donnée sont :

- Définir une fonctionnelle à minimiser pour le problème donné.
- Déduire de la fonctionnelle d'énergie l'équation d'évolution du contour actif.
- Implémenter cette équation d'évolution avec des méthodes appropriées.

2.3.1 Définition énergétique du contour actif « *snake* »

La définition de la fonctionnelle d'énergie constitue une étape fondamentale dans le processus de segmentation par contour actif. Elle définira le comportement géométrique du modèle ainsi que les données de l'image dont il se servira pour évoluer. Le choix de la fonctionnelle d'énergie associée au contour détermine en grande partie les capacités de segmentation du modèle. Il est donc indispensable qu'elle modélise au mieux le comportement souhaité du contour actif. La fonctionnelle d'énergie rattachée au contour actif ou *snake* se compose comme suit :

$$E_p(v) = E_{interne}(v) + E_{image}(v) + E_{contexte}(v) \tag{2.5}$$

a) Energie interne

L'énergie interne gère la cohérence de la courbe. Elle maintient la cohésion des points et la raideur de la courbe [3], en empêchant des nœuds individuels sur le contour de se balader trop loin de leurs nœuds voisins. Elle est définie comme suit :

$$E_{interne}(v) = \int_a^b [\alpha(s)\,|v'(s)|^2 + \beta(s)\,|v''(s)|^2]ds \tag{2.6}$$

Les termes v' et v'' sont les dérivées première et seconde de v par rapport à s. $\alpha(s)$: Coefficient d'élasticité. $\beta(s)$: Coefficient de rigidité. Ces deux coefficients permettent de pondérer l'influence de chaque terme. Ils sont généralement considérés comme étant constants. Les deux termes v' et v'' de régularisation sont en réalité un cas particulier de l'opérateur de Tikhonov d'ordre deux en choisissant des poids w_k constants [36] :

$$E_{Tikhonov}(v(s,t)) = \sum_{k=0}^{n} \int_0^1 w_k(s) \left|\frac{\partial^k v(s,t)}{\partial s^k}\right|^2 \tag{2.7}$$

Où n est l'ordre du stabilisateur, $w_k(s)$ est un coefficient de pondération et a et b sont les limites du contour. Le choix de l'ordre du stabilisateur est lié à la régularité requise sur la courbe. Kass et *al.* utilisent un stabilisateur d'ordre 2 (n=2) ce qui permet de définir les termes de l'énergie interne [7].

Le critère intrinsèque du premier ordre de l'équation (2.6) ajoute une contrainte élastique sur la courbe $v(s,t)$. Sa minimisation correspond en physique à l'équation d'évolution d'une membrane pour des déplacements faibles [1].

Le terme du second ordre est une contrainte sur la rigidité de la courbe. Sa minimisation en physique est similaire à l'équation d'évolution d'une plaque mince pour des déplacements faibles [1].

Le premier critère intrinsèque agit donc sur la longueur de la courbe alors que le deuxième permet de contrôler sa courbure. [1]

La propriété d'élasticité peut être illustrée en examinant le terme de la dérivée première $|v'(s)|^2$, qui n'est autre que le module au carrée du vecteur tangent à la courbe :

$$|v'(s)|^2 = \left(\frac{dx(s)}{ds}\right)^2 + \left(\frac{dy(s)}{ds}\right)^2 \tag{2.8}$$

D'autre part la longueur d'une courbe définie paramétriquement peut être exprimée par :

$$L = \int_a^b \sqrt{\left(\frac{dx(s)}{ds}\right)^2 + \left(\frac{dy(s)}{ds}\right)^2}\, ds \tag{2.9}$$

Les deux expressions en (2.8) et (2.9) montrent que la longueur du contour n'est qu'une simple intégration du module de la dérivée première le long de la courbe. La minimisation de la dérivée première conduit à une minimisation de la longueur globale du contour, ce qui reflète une certaine élasticité entre les différents nœuds du contour, c'est-à -dire les nœuds sont attirés entres eux [4]. Puisque le poids relatif de ce terme est contrôlé par α(s), plus ce paramètre est grand, plus l'élasticité est grande, et plus la tendance du contour à se contracter est grande.

De même, la rigidité est représentée par la dérivée seconde, $\left|v''(s)\right|^2$, qui n'est autre que le taux du changement de la valeur de la tangente à la courbe. Minimiser ce module revient à réduire la possibilité d'un changement brusque en n'importe quel nœud [4].

b) Energie image

L'énergie d'image correspond au critère d'attache aux données. Elle prend en compte les caractéristiques de l'image. Elle augmente à mesure qu'on se rapproche des contours d'objets recherchés qui sont donc les points de fort gradient, entrainant ainsi la diminution de l'énergie globale du *snake*. Les différentes formulations de l'énergie externe sont :

- **Gradient**

  La fonction suivante est introduite pour la recherche des zones de fort contraste dans l'image.

$$E_{externe}(v) = -\int_0^1 \left\| \nabla \mathrm{I}\big(v(s)\big) \right\|^2 ds \tag{2.10}$$

  Où $\nabla I\big(v(s)\big)$ représente le gradient de l'image I en $v(s)$.

  Très souvent, c'est le gradient gaussien qui est utilisé :

$$E_{externe}(v) = -\int_0^1 \left\| \nabla \left(\mathrm{g}_\sigma * I\right)\big(v(s)\big) \right\|^2 ds \tag{2.11}$$

  Où g_σ est la gaussienne centrée d'écart type σ.

- **Intensité**

  Cette énergie, au contraire, permet de sélectionner les zones sombres ou claires selon le signe choisi :

$$E_{intensité} = \pm \int_0^1 \big(I\big(v(s)\big) - i_0\big)^2 ds \tag{2.12}$$

  La valeur i_0 introduit ou non, un certain seuillage. On peut ainsi favoriser la position du contour dans une zone donnée.

c) Energie de contexte

L'énergie de contexte, parfois appelée énergie de contrainte, permet d'introduire des connaissances *a priori* sur le contour. C'est une énergie qui exprime des contraintes supplémentaires qui peuvent être imposées par l'utilisateur pour obtenir le contour recherché.

Le *snake* représente donc le premier modèle de contours actifs à être utilisé et sa fonctionnelle d'énergie qui régie son processus de déformation, prends, après avoirs définis les différents termes qui la constitue, la forme suivante :

$$E\big(v(s,t)\big) = \alpha \int_a^b |v'(s,t)|^2 ds + \beta \int_a^b |v''(s,t)|^2 ds - \gamma \int_a^b P\, ds \qquad (2.13)$$

Avec α, β et γ des coefficients qui permettent de pondérer l'influence de chaque terme.

2.4 Minimisation de la fonctionnelle d'énergie des *snakes*

On rappelle que l'objectif consiste à résoudre la minimisation de l'énergie définie en équation (2.13) qui ne connait pas de solution analytique. Elle nécessite alors pour sa résolution un schéma itératif. Ainsi, l'approche Lagrangienne est utilisée [8]. Elle fournit une représentation explicite de la déformation. Elle considère le modèle, non plus comme une entité statique, mais comme une entité dynamique qui évolue dans le temps. Elle rend compte des mouvements et des évolutions du modèle : on parlera donc de la position, de la vitesse et de l'accélération du modèle.

De manière plus formelle, considérons le contour de l'image comme un système mécanique de densité massique μ, soumis à une énergie potentielle E_p. On rappelle que lagrangien L du système est défini par :

$$L = E_c - E_p \qquad (2.14)$$

Où E_c est l'énergie cinétique définie par :

$$E_c = \frac{1}{2}\, \mu\, v_t^2 \qquad (2.15)$$

Avec :

$$v_t = \frac{ds}{dt} \qquad (2.16)$$

L'action intégrale In du système se définie alors comme suit :

$$In = \int_{t_1}^{t_2} L(t)\, dt \qquad (2.17)$$

On suppose que le système est non conservatif. Il est alors soumis à des forces de frottements, l'équation (2.17) devient alors :

$$In = \int_{t_1}^{t_2} (L + W)\, dt \qquad (2.18)$$

Où *W* correspond au travail des forces dissipatives.

Dans le cas contours actifs, la seule force dissipative que l'on considère est une force de frottements visqueux appelée force dissipative de Royleigh [8], proportionnelle à la vitesse d'avancement des points du modèle et donnée par :

$$F = -\gamma\, v_t \tag{2.19}$$

Cette force permet d'amortir le mouvement du contour actif, donc de diminuer ses oscillations, et ainsi lui assurer une convergence vers un état d'équilibre.

L'évolution réelle du système entre le temps t_1 et le t_2 correspond au mouvement pour lequel l'action *In* est extrémale. La condition nécessaire et suffisante pour que *In* soit extrémale est que pour tout déplacement infinitésimale on ait :

$$\delta In = 0 \equiv \delta\left(\int_{t_1}^{t_2}\left(E_c - E_p + W\right)dt\right) = 0 \tag{2.20}$$

Le minimum de cette équation est donné par le théorème d'Euler- Lagrange, portant sur le calcul des variations. Pour une densité de masse constante μ et un coefficient de dissiption constant γ, l'application du théorème d'Euler-Lagrange à l'équation (2.20), donne :

$$\mu\, v_{tt} + \gamma\, v_t + \frac{\partial(E_p)}{\partial v} = 0 \tag{2.21}$$

Compte tenu de l'équation (1.2), l'équation (2.21) s'écrit :

$$\mu\, v_{tt} + \gamma\, v_t + \frac{\partial(E_{int})}{\partial v} = -\lambda\, \frac{\partial(E_{ext})}{\partial v} \tag{2.22}$$

Généralement pour plus de simplicité, on considère que le modèle et sans inertie ($\mu = 0$). On obtient alors une équation du 1er ordre et on parlera d'évolution lagrangienne. En effet, la formulation de Kass et *al.* revient à négliger le terme d'inertie ($\mu = 0$), tout en conservant le terme de frottement γ et prendre λ =1. Le facteur γ peut alors être considéré comme un pas de temps. Dans ce cas l'équation (2. 22) donne lieu à deux équations indépendantes en x et y :

$$\begin{cases} \gamma\, \dfrac{\partial v}{\partial t} - \alpha\, \dfrac{\partial(x_s)}{\partial s} + \beta\, \dfrac{\partial^2(x_{ss})}{\partial s^2} = -\dfrac{\partial(E_{ext})}{\partial s} \\ \gamma\, \dfrac{\partial v}{\partial t} - \alpha \dfrac{\partial(y_s)}{\partial s} + \beta\, \dfrac{\partial^2(y_{ss})}{\partial s^2} = -\dfrac{\partial(E_{ext})}{\partial s} \end{cases} \tag{2.23}$$

Le terme $F = \frac{\partial(E_{ext})}{\partial s}$ correspond au gradient de l'énergie de l'image et peut ainsi être assimilé à une force d'attraction vers les caractéristiques de l'image.

Afin de déterminer la solution de l'équation (2.23) il est nécessaire de passer par la discrétisation à l'aide d'une méthode utilisée couramment appelée différence finie, on obtient :

$$v(t) = (A + \gamma I_d)^{-1} \left(\gamma v(t-1) - \frac{\partial E_{ext}(t-1)}{\partial s}\right) \quad (2.24)$$

*A*vec $v(t-1)$ et $v(t)$ représentent, respectivement, la position de snake aux instants *t-1* et *t*. A et I_d représentent respectivement, une matrice pentadiagonale et une matrice identité que nous verrons par la suite.

La mise en œuvre d'une telle approche a donné lieu à de nombreuses implémentations. Ces dernières peuvent être divisées en deux catégories, méthodes d'optimisation locale, et méthode d'optimisation globale. Avant d'examiner ces différentes implémentations, nous allons préciser comment les différentes méthodes de discrétisation permettent de résoudre informatiquement le problème de l'implémentation de notions définies dans un espace réel.

2.5 Méthodes de discrétisation

Il existe plusieurs méthodes de discrétisation. Parmi les plus utilisées, on trouve les différences finies et les éléments finis.

2.5.1 Différences finies

Les éléments de la courbe sont réduits en des points auxquels sont attachés les éléments mécaniques (masse, raideur, etc.) considérées en ces points. Cette méthode consiste à remplacer les dérivées apparaissant dans le problème continu par des différences finies [5]. Un point quelconque de la courbe est représenté par :

$$v(s_i) = v_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \quad (2.25)$$

La dérivée première des coordonnées par rapport au paramètre s peut être approximée par différences finies et le carré de la norme du vecteur des premières dérivées devient :

$$\|v'_i(s)\|^2 = \left\|\frac{d\, v_i}{ds}\right\|^2 \quad (2.26)$$

L'énergie de continuité est alors liée à :

$$\|v_i - v_{i-1}\|^2 = (x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 \quad (2.27)$$

La dérivée seconde s'approxime par :

$$\|v''_i(s)\|^2 = \left\|\frac{d^2 v_i}{ds^2}\right\|^2 \tag{2.28}$$

L'énergie de courbure dépend alors de :

$$\|v_{i-1} - 2v_i + v_{i+1}\|^2 = (x_{i-1} - 2x_i + x_{i+1})^2 + (y_{i-1} - 2y_i + y_{i+1})^2 \tag{2.29}$$

2.5.2 Eléments finis

Laurent David Cohen et Isaac Cohen [37] ont montré que la recherche de solutions dans un espace de Sobolev, était équivalente à celle dans un espace des fonctions polynomiales de dimension finie. Ils obtiennent, avec des éléments finis, des calculs similaires aux calculs utilisant les différences finies. Selon Laurent David Cohen, la méthode par éléments finis est moins coûteuse et plus stable. Le contour C est découpé en un nombre fini d'éléments. Chaque élément possède un nombre de nœuds n_n déterminé par le degré des dérivées de l'équation générale (2.13). Deux éléments voisins possèdent un nœud commun. Les coordonnées du vecteur position, $x(s)$ et $y(s)$, dans chaque élément peuvent être exprimées par :

$$x(s) = \sum_{j=1}^{n_n} x_j \varphi_j(s) \tag{2.30}$$

$$y(s) = \sum_{j=1}^{n_n} y_j \varphi_j(s) \tag{2.31}$$

$\varphi(s)$ est la fonction de chaque élément.

2.6 Méthodes d'optimisation locales

Il existe plusieurs méthodes d'optimisation locales. Parmi elles on peut citer celles de descente de gradient, les méthodes variationnelles et l'algorithme glouton ou Greedy.

2.6.1 Méthode de la descente du gradient

Les méthodes de descente du gradient sont caractérisées par l'évolution d'une solution (courbe par exemple) initiale v_0 dans la direction de la plus forte pente (énergie par exemple). La solution évolue alors selon l'équation

$$v_{k+1} = v_k - \Delta t \nabla E_p (v_k) \tag{2.32}$$

Où Δt est le pas du gradient et $\nabla E_p\ (v_k)$ est le gradient de l'énergie potentielle. Δt peut être déterminé de deux manières :

- Par la méthode du Pas de descente : Δt est détérminé de manière à avoir $E_p(v_{k+1}) < E_p(v_k)$. Généralement, la valeur de Δt obtenue est contenue dans un intervalle, alors le choix de la valeur de Δt s'effectue arbitrairement.
- Par la méthode du Pas de descente optimale : c'est-à-dire que l'on considère un problème d'optimisation avec comme fonction à minimiser $E_p(v_{k+1})$ selon le critère de minimisation Δt, cette méthode permet d'avoir une valeur exacte de Δt.

Ces méthodes font converger vers le minimum local de l'énergie. La solution initiale est donc primordiale. D'autre part, cela fait évoluer le modèle à pas constant, ce qui ne conduit pas forcément au minimum global (figure 2.2).

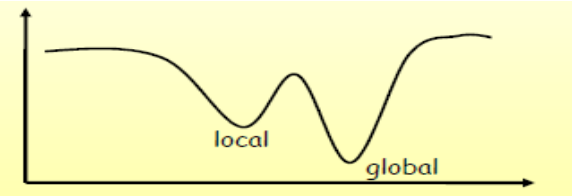


Fig. 2.2. *Représentation du minimum local et global d'une fonction.*

Il existe cependant des méthodes permettant d'accélérer la convergence vers le minimum global à chaque itération. Parmi ces méthodes nous pouvons citer la méthode du gradient conjugué, la méthode de Newton ou encore la méthode de Lavenberg-Marquart.

2.6.2 Méthodes variationnelles

L'approche variationnelle est la méthode la plus utilisée dans les contours actifs. Elle a été introduite par Kass [35]. Elle a pour objectif de minimiser une fonctionnelle d'énergie, composée d'une énergie interne, d'une énergie externe, éventuellement d'une énergie de contexte:

$$\mathrm{E}(v) = \int_0^1 \left(\alpha\,|v'|^2 + \beta\,|v''|^2 + P\big(v\,(s)\big)\right) ds \qquad (2.33)$$

Un minimum vérifie les équations suivantes (2.34) qui sont obtenues après application à l'équation (2.33) le théorème d'Euler-Lagrange portant sur le calcul des variations d'où le nom de la méthode.

$$\begin{cases} -(\alpha\, v')' + (\beta\, v'')'' + \nabla P(v) = 0 \\ v(0), v'(0) \ \ données \end{cases} \tag{2.34}$$

Cette équation peut avoir plusieurs solutions puisque l'énergie peut avoir plusieurs minimums locaux. De ce fait, la solution que l'on recherche est localisée dans une région donnée et on suppose posséder une valeur approchée v_0 de la solution. On résout alors l'équation (2.33) en lui ajoutant un terme d'évolution $\gamma \frac{\partial v}{\partial t}$ pour former :

$$\gamma \frac{\partial v}{\partial t} - (\alpha v')' + (\beta v'')'' = -\nabla P(v) \tag{2.35}$$

En discrétisant l'équation (2.34) avec une pas *h* du paramètre *s*, et les dérivées seront remplacées par les différences finies, nous auront alors :

$$\begin{aligned} &\frac{1}{h}\left(a_i(v_i - v_{i-1}) - a_{i+1}(v_{i+1} - v_i)\right) + \frac{b_{i-1}}{h^2}(v_{i-2} - 2v_{i-1} + v_i) \\ &-2\frac{b_i}{h^2}(v_{i-1} - 2v_i + v_{i+1}) + \frac{b_{i+1}}{h^2}(v_{i+2} - 2v_{i+1} + v_i) + \nabla P(v_i) = 0 \end{aligned} \tag{2.36}$$

Où $v_i = v(ih)$ $\quad a_i = \dfrac{\alpha(ih)}{h}$ $\quad b_i = \dfrac{\beta(ih)}{h^2}$

Cette équation peut s'écrire :

$$AV=F \tag{2.37}$$

Où *A* est une matrice pentadiagonale, *V* représente les vecteurs de positionv_i, et *F* les forces externes $P(v_i)$ en ces points. *V* et *F* sont des matrices à deux colonnes, la première pour les composantes en *x* et la seconde pour *y*. Le terme d'évolution est aussi remplacé par les différences finies. La discrétisation s'effectue dans le domaine temporel. L'équation (2.35), au point s'indice *i*, devient :

$$\begin{aligned} &\gamma\left(v_i(t) - v_i(t-1)\right) + \tfrac{1}{h^2}\left(\alpha_i\left(v_i(t) - v_{i-1}(t)\right) - \alpha_{i+1}\left(v_{i+1}(t) - v_i(t)\right)\right) \\ &+\tfrac{1}{h^4}\begin{pmatrix}\beta_{i-1}\left(v_{i-2}(t) - 2v_{i-1}(t) + v_i(t)\right) - 2\beta_i(v_{i-1}(t) - 2v_i(t) + v_{i+1}(t)) \\ +\beta_{i+1}\left(v_i(t) - 2v_{i+1}(t) + v_{i+2}(t)\right)\end{pmatrix} = -\nabla P(v_i) \end{aligned} \tag{2.38}$$

En prenant $a_i = a$ et $\beta_i = \beta$ constants et *h =1*, on obtient, après des calculs appropriés, l'équation (2.39) ci –dessous.

$$\begin{aligned} &\gamma\left(v_i(t) - v_i(t-1)\right) + \beta v_{i-2}(t) + (-\alpha - 4\beta)v_{i-1}(t) + (2\alpha + 6\beta)v_i(t) + \\ &(-\alpha - 4\beta)v_{i+1}(t) + \beta v_{i+2}(t) = -\nabla P(v_i) \end{aligned} \tag{2.39}$$

Cette équation peut encore être écrite sous la forme matricielle :

$$v(t) = (A + \gamma I_d)^{-1} \left(\gamma\, v(t-1) - \nabla P\big(v(t-1)\big)\right) \tag{2.40}$$

Où la matrice *A* est une matrice pentagiagonale dont ses éléments sont fonctions de α, β . I_d est la matrice identité et γ représente l'inertie de la courbe par rapport aux déplacements.

On peut classer les *snakes* en trois types différents :

- Les contours actifs fermés où $v_0(t) = v_{n-1}(t)$, *n* étant le nombre de points de la courbe.
- Les contours actifs ouverts à extrémités libres.
- Les contours actifs ouverts à extrémités fixes où les positions v_0 et v_{n-1} sont des fixes dans le temps.

Pour un modèle de snake fermé, la matrice *A* est pentadiagonale, circulaire et symétrique de taille $n*n$ (figure 2.3).

$$\begin{bmatrix}
2\alpha+6\beta & -\alpha-4\beta & \beta & 0 & \cdots & 0 & \beta & -\alpha-4\beta \\
-\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \beta & \cdots & \cdots & 0 & \beta \\
\beta & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta & \cdots & \cdots & \cdots & 0 \\
0 & \beta & -\alpha-4\beta & 2\alpha+6\beta & \cdots & \cdots & 0 & \cdots \\
\cdots & \cdots & \cdots & -\alpha-4\beta & \cdots & \cdots & \beta & 0 \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & -\alpha-4\beta & \beta \\
\beta & 0 & \cdots & \cdots & \cdots & -\alpha-4\beta & 2\alpha+6\beta & -\alpha-4\beta \\
-\alpha-4\beta & \beta & 0 & \cdots & 0 & \beta & -\alpha-4\beta & 2\alpha+6\beta
\end{bmatrix}$$

Fig. 2.3. *Représentation de la matrice A d'un snake fermé.*

Cette méthode nécessite pour atteindre la solution, à chaque itération, l'inversion de la matrice $(A + \gamma\, I_d)$.

2.6.3 Algorithme Greedy

En cherchant à améliorer et à corriger certains des problèmes associés au modèle de Kass et *al*. et à son implémentation par la méthode variationnelle, DJ Williams et M. Shah ont élaboré un algorithme baptisé le Greedy snake ou l'algorithme Greedy [38]. L'un des problèmes, auquel se sont référés DJ Williams et M. Shah est celui des points du *snake*, du modèle Kass et *al.*, qui présentent l'inconvénient de ne pas être toujours régulièrement espacées. Parfois, les points de contrôle se regroupent en bouquet dans les zones du contour

qui présentent de fortes valeurs du gradient. Ce comportement provoque des problèmes avec le calcul de la courbure. L'algorithme Greedy aborde également le problème de l'instabilité numérique qui pourrait se produire si le pas temporel Δt est trop grand. En outre, les problèmes découlant de la discrétisation par les différences finies des termes d'élasticité et de courbure, sont évaluées et de nouvelles approximations discrètes sont proposées [6].

Son principe, inspiré de l'approche gloutonne utilisée en optimisation [1] (d'où l'appellation algorithme glouton), part de l'hypothèse qu'une succession de choix optimaux localement peut conduire à une solution optimale globalement. La position de chaque point $vi = (x_i, y_i)$ est donc modifiée en limitant l'espace de recherche à son voisinage. Cette implémentation permet de passer à une complexité en $O(nm)$ où n est la taille de l'image et m la taille du voisinage considéré pour chaque point de contrôle. Nous commencerons notre analyse de l'algorithme Greedy par une évaluation de l'énergie image.

Remarque:

Dés ce point, nous utiliserons, aléatoirement, les deux appellations de l'algorithme, à savoir: Greedy et glouton.

2.6.3.1 Le terme d'énergie image de l'algorithme glouton

L'énergie image est évaluée d'une manière légèrement différente pour les algorithmes gloutons [6]. Cette différence est principalement liée à la nature discrète de l'algorithme glouton. Son terme d'énergie image est calculé comme suit:

$$E_{img} = \|\nabla [G_\sigma(x,y) * I(x,y)]\|^2 \tag{2.41}$$

On retrouve dans les deux modèles presque la même formulation, sauf que dans l'algorithme glouton l'énergie image ne comporte pas de signe moins devant son terme. L'algorithme glouton va examiner le voisinage de chaque point de la courbe du snake, puis se déplacera vers le point ayant l'énergie la plus faible. Par conséquent, l'énergie de l'image dans le voisinage du point du *snake* v(si) doit être normalisée de manière à attribuer de grandes valeurs négatives aux pixels présentant des valeurs élevées du gradient, tout en assignant de petites valeurs négatives aux pixels présentant de faibles valeurs de gradient. Les grandeurs des gradients se présentent toutes dans l'intervalle [0,255]. La normalisation de l'intervalle est calculée comme suit [6] :

$$location(x,y) = \frac{(min - mag(x,y))}{(max - min)} \quad (2.42)$$

Où min est la valeur minimale du gradient dans le voisinage, max la valeur maximale et $mag(x,y)$ l'amplitude du gradient du point. Cette normalisation définit la magnitude de gradient la plus élevée dans le voisinage de -1 et la plus basse à 0. Le problème de cette méthode est que si le voisinage est à peu près uniforme en amplitude du gradient nous obtenons une grande différence dans les valeurs normalisées. Par exemple, si toutes les valeurs d'amplitude de gradient sont dans l'intervalle [46, 49], alors les valeurs normalisées serait 0, - 0,33, - 0,66 et -1, ce qui suggère une forte avance, même quand il n'y en a aucune. Afin de compenser ce problème, la valeur minimum min est fixée à $max - 5$ si $max - min < 5$. Un exemple d'un voisinage avec les mêmes valeurs d'amplitude de gradient et les valeurs correspondantes de l'énergie d'image, est donné dans la figure 2.4.

Fig. 2.4. *Illustration d'un voisinage 3x3 autour du point de $v(si)$ ainsi que l'énergie d'image. Les chiffres en rouge représentent l'intensité du gradient, tandis que les chiffres en bleu sont des valeurs normalisées de l'énergie image.*

La figure montre aussi comment le fait de fixer min à $max - 5$ conduit à une vue plus précise de la similitude des valeurs de gradient.

2.6.3.2 Le terme d'élasticité de l'algorithme glouton

Comme il a était vu précédemment, le terme du seconds ordre de l'image interne a pour effet de provoquer l'effondrement de la courbe sur elle-même. Ce terme est calculé d'une autre manière dans l'algorithme glouton. Réduisant ainsi l'effet de rétrécissement et permet aussi de faire en sorte que les points du *snake* ne se tassent pas dans les zones à fort gradient.

Dans l'algorithme glouton le terme d'élasticité est calculé dans le voisinage de chaque point de contrôle comme suit :

$$E_{ela} = \overline{d} - \|v_i - v_{i-1}\| = \overline{d} - \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \quad (2.43)$$

Où $\overline{d}$ est la distance moyenne entre deux points de contrôle consécutif. Une fois le terme ci-dessus calculé pour chaque pixel dans le voisinage du point de contrôle, toutes les valeurs sont divisées par la plus grande valeur dans le voisinage. Ce qui signifie que le voisinage est normalisé de sorte à ne contenir que des valeurs du terme d'élasticité entre [0,1]. L'énergie minimale sera obtenue lorsque l'équation suivante est satisfaite [6]:

$$\overline{d} - \|v_i - v_{i-1}\| = 0 \quad (2.44)$$

Cela est vrai pour les points où la distance moyenne est égale à la distance entre le point courant du *snake* et son prédécesseur. Le nouveau terme d'élasticité va donc encourager les points à être espacés régulièrement le long de la courbe en maintenant la courbe paramétrée par la longueur d'arc.

Tout comme dans la fonctionnelle d'énergie du modèle de Kass et *al.,* l'influence du terme d'élasticité est contrôlée par le paramètre $\alpha(s_i)$. Trouver une valeur adéquate du paramètre est tout aussi important dans l'algorithme glouton que dans le modèle de Kass et *al.* Si le paramètre est trop élevé le contour actif ne sera pas en mesure d'évoluer vers le contour de l'objet car la longueur de la courbe n'est pas modifiée.

2.6.3.3 Le terme de courbure dans l'algorithme glouton

Dans [38], Williams DJ et M. Shah évaluent une gamme de différentes méthodes de calcul de la courbure d'une courbe paramétrique discrète, telle que la courbe de l'algorithme Greedy. La discrétisation de l'expression de courbure par différences finies est donnée comme suit :

$$E_{curv} = \|v(s_i + 1) - 2v(s_i) + v(s_i - 1)\|^2 \quad (2.45)$$

Cette équation fait un compromis entre l'intensité de calcul et la précision de l'expression. La courbure est calculée pour chacun des points du voisinage du point de contrôle. C'est la même expression que nous obtiendrions si le terme $\|v''(s_i)\|$ est approximé en utilisant la formule des différences finies vue précédemment avec h = 1.

Ce terme ne mesure pas la courbure avec précision si le *snake* n'est pas paramétré par une longueur d'arc. Nous pouvons constater cela en observant un exemple discret. Dans la figure 2.5(a) nous avons amplifié une partie de la courbe où le terme vu en (2.45) serait supérieur à zéro. Puisque dans cette figure la courbe en fait s'infléchit, nous atteignons les résultats souhaités. Dans la figure 2.5 (b) les points de la courbe ne sont pas équidistants et même si la courbe ne s'infléchit pas, nous remarquons que le terme en (2.45) serait supérieur à zéro. Toutefois, les points de contrôle dans l'algorithme glouton semblent plus enclins à être régulièrement espacés que pour le modèle Kass et *al.*, puisque le terme d'élasticité encourage même l'espacement des points de contrôle. L'influence de la courbure est commandée par le paramètre β comme dans le *snake* de Kass et *al.*

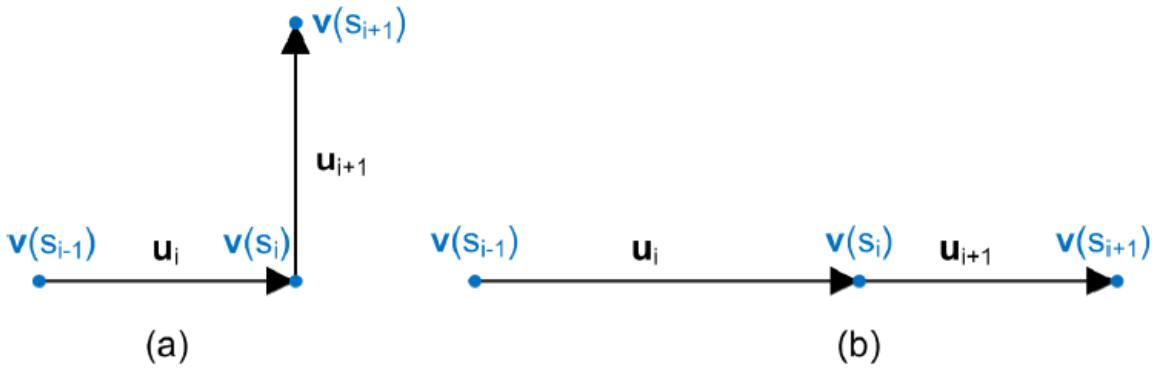


Fig. 2.5. *Illustration montrant deux parties différentes de la courbe du snake. (a) dans cette partie la courbe est pliée et les points sont équidistants puisque les deux vecteurs U_i et U_{i+1} ont la même longueur. (b) dans cette partie la courbure est égale à zéro tandis que les deux vecteurs ont des longueurs différentes.*

Une fois que la courbure a été calculée pour chaque point du voisinage du point courant, les valeurs sont normalisées en divisant par la plus grande valeur. Nous aurons alors des valeurs de courbure toutes réunies dans l'intervalle [0,1], comme pour l'énergie image et le terme d'élasticité.

2.6.3.4 Algorithme

Dans les sections précédentes, nous avons expliqué comment chacun des termes d'énergie est calculés pour l'algorithme Greedy. Dans cette section, nous allons décrire comment l'algorithme fait évoluer les points.

Pour chaque point/pixel dans le voisinage d'un point de contrôle v(si) les trois termes d'énergie sont calculés. Ensuite, l'algorithme résume les termes d'énergie pour obtenir l'énergie combinée :

$$E_{com}(x,y) = \left(\alpha(s_i)E_{ela}(x,y) + \beta(s_i)E_{curv}(x,y) + \gamma(s_i)E_{image}(x,y)\right) \qquad (2.46)$$

Où $E_{ela}(x,y)$ est l'énergie d'élasticité, $E_{curv}(x,y)$ est l'énergie de courbure, $E_{img}(x,y)$ est l'énergie d'image et (x,y) sont les indices des points dans le voisinage. En outre, nous voyons que l'algorithme glouton utilise aussi un paramètre γ pour le contrôle de l'influence de l'énergie image. Une fois que l'énergie totale est calculée pour chaque pixel appartenant à ce voisinage, celui possédant l'énergie minimale est alors choisi comme nouvelle position pour le point de contrôle (figure 2.6).

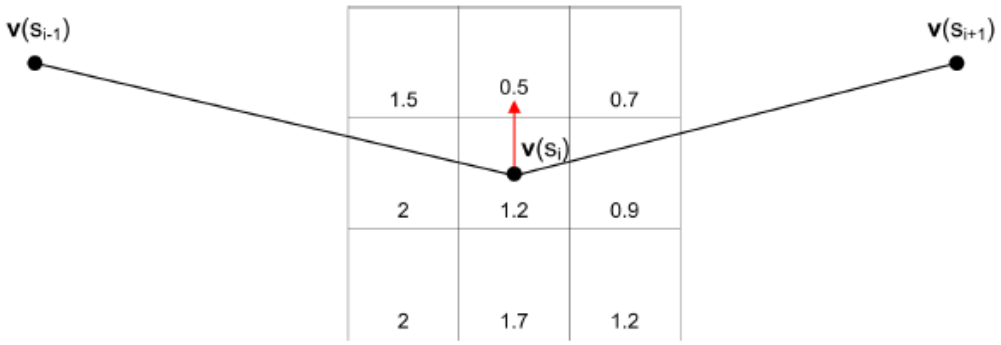


Fig. 2.6. *Illustration du déplacement du point de contrôle. Les valeurs contenue dans le carrée représente celles de l'énergie image.* $v(s_i)$*, représente le point de contrôle,* $v(s_i)$ *et* $v(s_i)$ *représentent le prédécesseur et le point suivant. La flèche en rouge désigne la position vers laquelle va évoluer le point de contrôle.*

Une fois que tous les points de contrôles le long de la courbe ont été déplacés vers les nouvelles positions, la courbure est de nouveau calculée, mais cette fois-ci uniquement pour les points de contrôle. La raison pour laquelle ce nouveau calcul est effectué est la localisation des endroits où la courbure est élevée, et relaxer le paramètre $\beta(s_i)$ pour ce point de contrôle, en d'autres termes mettre $\beta(s_i) = 0$, la présence de coin n'est donc pas autorisée à se former en ce point. Pour le calcul de la courbure nous utiliserons [6]:

$$\left\| \frac{u_i}{\|u_i\|} - \frac{u_{i+1}}{\|u_{i+1}\|} \right\|^2 \qquad (2.47)$$

Avec : $u_i = [x(s_i) - x(s_{i-1}), y(s_i) - y(s_{i-1})]^T$ et $u_{i+1} = [x(s_{i+1}) - x(s_i), y(s_{i+1}) - y(s_i)]^T$.

Cela donne une estimation plus précise de la courbure, puisque on normalise par la magnitude des vecteurs. Nous évitons ainsi le problème des points devant être espacées de manière égale pour obtenir une mesure fiable de la courbure. L'équation (2.47) n'est pas utilisée, initialement, pour le calcul de la courbure pour tous les points du voisinage de chaque point de contrôle du *snake* car elle comporte trop de calcul. Une fois que la nouvelle courbure a été calculée pour tous les points de contrôle du *snake*, le paramètre β est relaxé pour les points de contrôle du *snake*, où les conditions suivantes sont satisfaites.

En premier lieu, la valeur de la courbure au point $v(s_i)$ doit être plus grande que celle de ses deux voisins $v(s_{i-1})$ et (s_{i+1}) . En second lieu la courbure en ce point doit être supérieure à une valeur de seuil prédéfinie. Enfin, l'amplitude du gradient doit être aussi au-dessus d'un seuil. Si toutes ces conditions sont satisfaites, alors la valeur de β est assouplie (ou adoucie). Lorsque la valeur de β a été assouplie, la courbure au point de contrôle correspondant n'a plus aucune influence sur l'énergie combinée du *snake* et donc un angle aigu sera en mesure de se développer.

La dernière étape dans une itération de l'algorithme Greedy consiste à vérifier le critère d'arrêt. Si le nombre de points déplacés dans l'itération est inférieur à certain seuil alors l'algorithme va s'arrêter. Le contour actif est supposé avoir atteint l'énergie minimale lorsque la plupart des points de contrôle ont cessé de bouger.

Une variante, permettant de rendre l'algorithme encore plus rapide, est proposée par Lam et Yan [39]. Il s'agit, par exemple, pour un voisinage de 3×3 pixels sur les 8 voisins, de n'en examiner que quatre. Si l'un de ces quatre améliore l'énergie totale, alors il n'est pas nécessaire d'aller plus loin. Sinon, les quatre qui restent sont examinés. Cela augmente le nombre d'itérations pour atteindre la convergence mais décroît le temps de calcul de chaque itération.

2.7 Méthodes d'optimisation globales

Il existe plusieurs méthodes d'optimisations globales. Ces méthodes permettent d'éviter le piégeage des contours actifs dans un minimum local. Parmi ces méthodes, on peut citer la programmation dynamique, le recueil simulé, les algorithmes génétiques, etc.

Le recuit simulé est l'une des plus anciennes métaheuristique qui dérive de la méthode de descente de gradient. Le processus du recuit simulé introduit une composante aléatoire

dans le processus de convergence. A chaque itération, on choisit une solution s', au voisinage de la solution précédente s, et on calcul la variation $f(s') - f(s)$. Si cette variation est négative, s' remplace s, et on recommence sinon on peut remplacer s par s', avec la probabilité $p_t = 1 - e^{-k(T-T_0)}$ où k une constante donnée et T le paramètre de¨température¨ qui diminue d'une valeur initiale vers T_0 lorsque l'algorithme converge. Du fait de sa nature stochastique, la méthode peut s'avérer laborieuse en temps de convergence à condition que la solution soit prête de la solution voulue. De ce fait Berger à utilisé ce principe pour la convergence des *snakes*. Elle s'est aperçue que les résultats étaient peu convaincants, car le modèle doit être près de la solution pour converger. En effet, l'inconnue s est une courbe et il est difficile de définir une procédure de choix au hasard d'une courbe dans un voisinage, étant donné les innombrables possibilités.

La programmation dynamique est une méthode d'optimisation globale qui estime l'optimum d'une fonction de manière récursive [8]. Son application aux contours actifs est due à Amini, Weymouth et Jain [40]. C'est une alternative intéressante au calcul variationnel [3]. Contrairement à la méthode variationnelle, la résolution du problème des contours actifs par la méthode de programmation dynamique converge vers l'optimum en un nombre fini d'itérations car sa fonction de cout décroit de manière monotone [41]. En partant d'un premier point du contour, il est possible de traiter le problème global de minimisation comme un problème de minimisation qui, pour chaque ensemble fini d'étapes $(i_0, i_1, \dots, i_{n-1})$, prend une décision parmi un ensemble fini de solutions possibles.[3]. Sachant que la discrétisation de l'énergie interne (E_{int}) met en jeu un point du contour ainsi que son prédécesseur et son successeur, l'énergie (E_{total}) s'exprime alors :

$$E_{tot}(v_1, v_2, \dots, v_n) = E_1(v_1, v_2, v_3) + E_2(v_2, v_3, v_4) + \dots + E_{n-2}(v_{n-2}, v_{n-1}, v_n) \quad (2.48)$$

Où : $$E_{i-1}(v_{i-1}, v_i, v_{i+1}) = E_{ext}(v_i) + E_{int}(v_{i-1}, v_i, v_{i+1}) \quad (2.49)$$

v_i représente la position du point i du *snake*.

On se ramène donc à un problème d'optimisation d'une fonction numérique de plusieurs variables. Les variables représentant les positions des différents points du *snake*. La formulation standard sous forme récurrente de la programmation dynamique peut s'écrire de cette manière [3] :

$$S_i(v_{i+1}, v_i) = \min_{v_{i-1}}\{S_{i-1}(v_i, v_{i-1}) + \alpha|v_i - v_{i-1}|^2 + \beta|v_{i+1} - 2v_i + v_{i-1}|^2 + E_{ext}(v_i)\} \quad (2.50)$$

La programmation dynamique est une méthode qui autorise l'introduction de contraintes comme l'énergie ballon. Elle s'avère cependant, être optimale uniquement pour des contours ouverts. Les contours fermés nécessitent l'emploi d'une contrainte supplémentaire, comme par exemple que le premier et dernier point doivent être similaires. Dans ce cas, l'utilisation de la programmation dynamique est possible mais en apportant des modifications. D'abord, on doit fixer un premier point et trouver le meilleur contour qui se termine au même point. Après avoir testé tous les points initiaux possibles, il faut alors, sélectionner celui qui minimise le plus l'énergie totale.

Les algorithmes génétiques ont été développés en premier par John Holland à l'université du Michigan [42]. Le principe des algorithmes génétiques est de coder chaque solution potentielle d'un problème par un « chromosome ». L'ensemble des chromosomes ou « individus » forme alors la « population » qui va être amenée à évoluer au cours du temps. Une « génération » est l'état de la population à un instant t. La première étape de l'algorithme génétique est la construction d'une population initiale, un ensemble de chromosomes qui représentent des contours. Ces chromosomes sont déterminés aléatoirement, c'est-à-dire qu'une suite de points ordonnés par l'ordre du tirage est choisie aléatoirement pour chaque contour. Ces points ne forment pas un contour fermé contrairement aux conditions d'une initialisation classique comme dans la méthode variationnelle. La solution serait alors de réordonner les points, soit en modifiant directement les chromosomes de la population initiale (ce qui revient à faire une initialisation semi-aléatoire), soit à partir de la fonction d'évaluation, juste après le décodage du génome. La population évolue au cours des générations en suivant des lois de sélection, de croisement et de mutation. En informatique, on parle d'opérateur génétique. La différence entre l'algorithme génétique et les autres méthodes d'implémentation des contours actifs réside dans le fait que dans les algorithmes génétiques, on ne fait pas évoluer un *snake* mais une population de *snake*. Seul le meilleur *snake* de la population à un instant donné est considéré comme l'approximation du contour. Si la taille de la population est suffisamment élevée, alors il est possible d'initialiser les contours de manière totalement aléatoire dans l'image. C'est l'effet des croisements et des mutations qui permet de faire converger la population vers des individus intéressants, cela permet un affranchissement de toute connaissance structurelle à priori de l'image. Les *snakes* traités ne se déplacent donc pas sous l'effet des forces mises en jeu, mais sont recombinés entres eux et mutés afin d'en créer d'autres dont l'énergie est moindre et qui ont atteints une bonne stabilité en fonctions des forces mises en jeu. La notion de déplacement dans le voisinage de chaque point disparaît totalement [3].

Nous allons maintenant, mettre en évidences les principaux avantages et inconvénients des *snakes*.

2.8 Analyse des caractéristiques des *snakes*

L'application de la formulation *snake* présente quelques avantages comme sa simplicité et son extensibilité à de nombreuses applications telles que la segmentation, la détection de contours, le suivi temporel de formes, etc. Elle combine la détection et le suivi de contours en une seule opération et elle comble l'information là où elle est manquante grâce au principe de régularisation. Elle permet aussi l'interaction de l'utilisateur [8].

Cependant les *snakes* présentent quelques inconvénients propres à eux, outre les inconvénients liés à la représentation paramétrique vus précédemment [1].

- Le contour est très sensible à l'initialisation, qui doit être réalisée à proximité de l'objet à segmenter.
- Malgré la contrainte de régularité, les points de la courbe ont tendance à se regrouper sur les zones de fort gradient, donnant lieu à des auto-intersections que le modèle est incapable de gérer.
- Seules les formes convexes sont bien segmentées, le modèle est plus approximatif sur les formes concaves (figure 2.7 et 2.8) [23,43]
- Incapacité du modèle à changer de topologie, donc à détecter plusieurs objets (figure 2.9).
- Le terme d'ordre deux de l'équation 2.6 introduit un terme d'ordre quatre dans l'équation d'Euler-Lagrange, entraînant une instabilité numérique du modèle.
- Il n'existe pas de méthode pour déterminer les paramètres α, β et γ car ils sont dépendant de l'image. L'utilisateur doit s'en remettre à des choix empiriques.

Fig.2.7. *Illustration de l'incapacité des snakes à converger vers les concavités*.

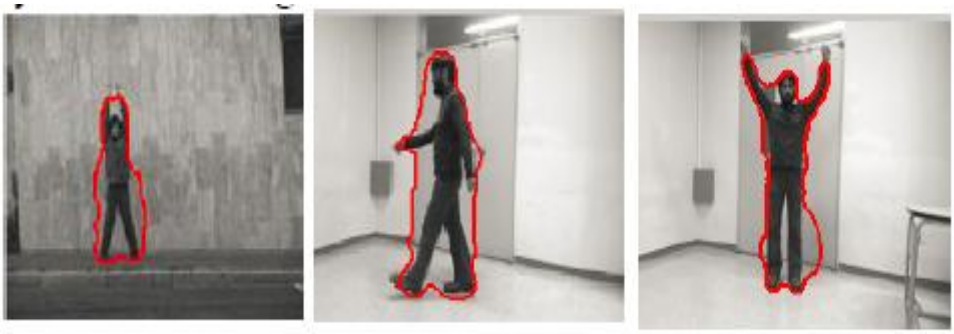


Fig. 2.8. *Illustration montrant l'incapacité des snake à segmenter des objet à forme concave.*

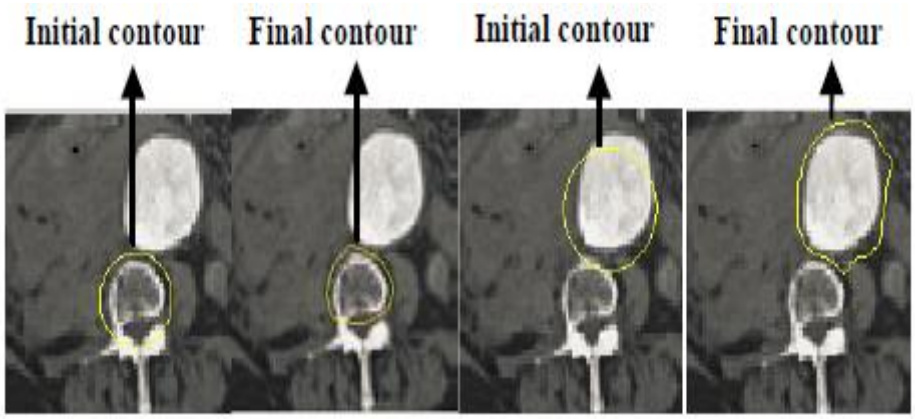


Fig. 2.9. *Incapacité des snakes à détecter plusieurs objets.*

De nombreux modèles ont alors été développés de manière à atténuer ou supprimer ces inconvénients.

Les *snakes*, de par leur discrétisation ont une tendance naturelle à se rétracter, la minimisation de l'énergie implique une minimisation de distance, et en cas d'une mauvaise initialisation il n'y aura pas d'attraction. Afin de pallier ces problèmes une énergie de contrainte appelée énergie ballons à été introduite par Laurent D. Cohen. [37]. La force ballon contribue à rendre le contour plus dynamique. Considérant le *snake* tel un ballon, elle va avoir un effet similaire à celui que procurerait la pression dans un ballon à mesure qu'on y introduirait de l'air.

Dans la grande majorité des implémentations des contours actifs, les auteurs utilisent directement le gradient de l'image comme force externe. Il s'agit certainement de la méthode la plus simple, mais le principal inconvénient de cette dernière est le caractère local de cette mesure [7]. En effet, si un tel critère permet de stopper l'évolution du contour actif sur les forts gradients, il n'est pas en mesure d'attirer la courbe vers une frontière lorsque celle-ci est située dans une zone très homogène, éloignée d'une zone de fort gradient [1]. Deux méthodes dérivées du gradient de l'image ont été proposées pour remédier à ces inconvénients. La

première consiste en une combinaison multi-échelles tandis que la seconde consiste en un flot vectoriel construit à partir du gradient de l'image. Le gradient multi-échelles est constitué d'une superposition linéaire de différents gradients de l'image. Le paramètre sigma définit l'espacement entre les points du gradient pour chacune des échelles ce qui modifie la résolution. Comme le montre la figure 2.10.a, un gradient calculé à petite échelle aura tendance à maximiser l'amplitude des vecteurs du gradient le long de la discontinuité. Par contre, ces vecteurs auront tendance à s'atténuer rapidement. Dans le cas d'un calcul de gradient à plus grande échelle (figure 2.10.b), l'amplitude du gradient est beaucoup moins grande, mais son effet existe à une plus grande distance de la discontinuité [7].

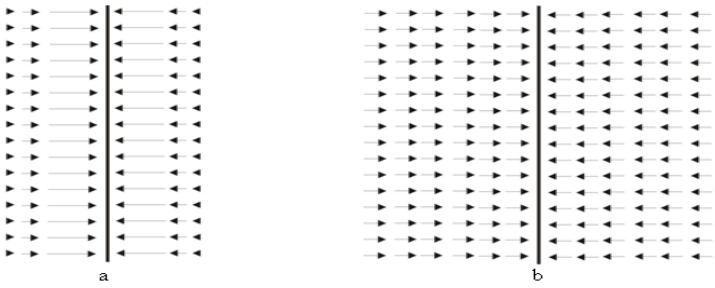


Fig. 2.10. *a) Exemple schématique de calcul du gradient à petite échelle pour une discontinuité et b) Exemple schématique de calcul du gradient à grande échelle pour une discontinuité.*

La superposition de ces différents gradients permet de combiner chacune des résolutions. Cette méthode permet à la fois d'augmenter l'amplitude du gradient aux abords de la discontinuité et aussi d'augmenter la distance pour laquelle cette discontinuité a un effet d'attraction sur le contour [7]. Dans [44], un nouveau terme d'attache aux données est proposé. Baptisé Gradient Vector Flow (GVF), celui-ci est basé sur la diffusion du gradient par champ vectoriel (figure 2.11). C'est un critère permettant au contour actif d'avoir une vision plus globale de la répartition du gradient d'intensité dans l'image.

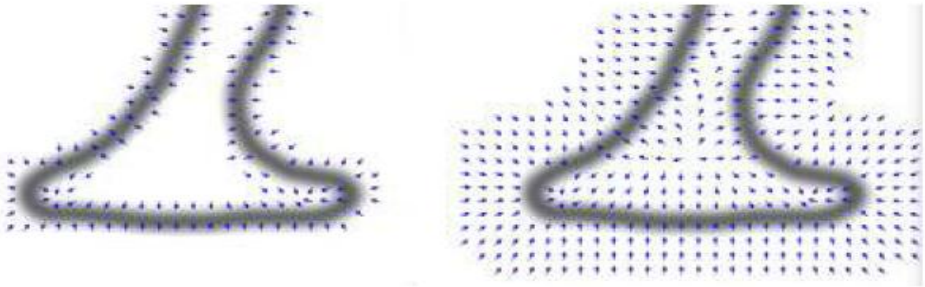


Fig. 2.11. *Illustration du champ de vecteur issu du gradient (image de gauche) et champ de vecteur diffusé par la méthode du GVF (image de droite).*

Une diffusion du gradient d'intensité est alors proposée de telle sorte qu'en chaque pixel de l'image soit accessible l'information des forts gradients présents dans son voisinage. La définition par diffusion permet aux forces du GVF de s'étendre loin de l'objet, elle permet donc au contour actif de retrouver le contour réel tout en étant placé plus loin qu'un contour actif traditionnel (figure 2.12). De plus, ce principe de diffusion permet également au contour actif de type GVF de rentrer dans les concavités, ce qui est un problème pour les contours actifs traditionnels (figure 2.12). Un autre avantage du GVF est qu'il permet de détecter un contour même si le tracé initial du contour actif traverse le contour réel, ce qui gêne la plus part du temps les *snakes* (figure 2.13) [4].

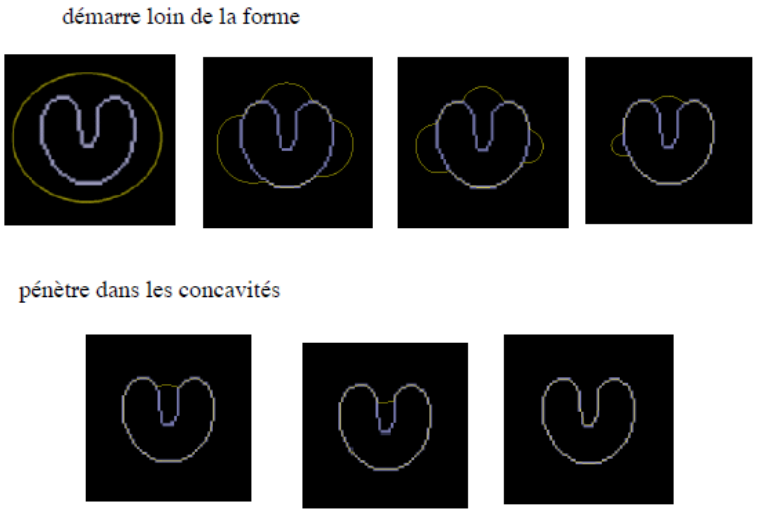


Fig. 2.12. *Initialisation loin de du contour recherché et capacité d'entourer les concavités.*

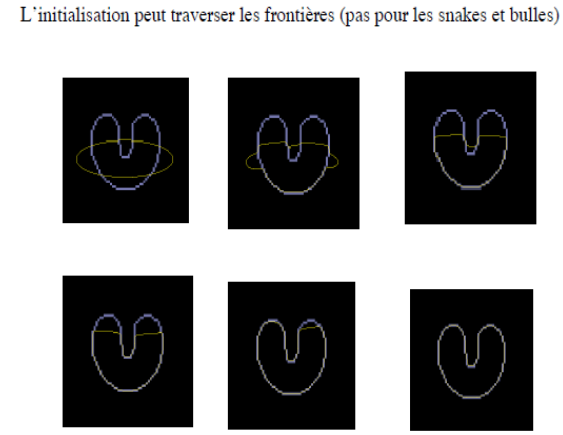


Figure 2.13. *Capacité du GVF initial à traverser les frontières de l'objet.*

Le GVF a été intégré avec succès dans plusieurs modèles de contours actifs. Xu et Prince l'ont initialement appliqué à un modèle représenté de manière paramétrique [44] mais celui-ci

a également été intégré à un contour actif géométrique, modèle représenté implicitement [45,1]. Cependant, le GVF comporte quelques inconvénients. En effet ce nouveau potentiel, d'un intérêt certain lorsque l'objet à segmenter est unique, peut poser problème dans le cas d'objets multiples dans des images réelles, la diffusion du gradient pouvant créer des interférences entre les zones d'influence des différents objets. Le temps de calcul de ce GVF peut aussi représenter un frein à cette méthode [22].

Nous venons de voir les modèles qui viennent à bout du problème de l'initialisation du contour et de sa capacité à entouré les objets à forme concaves. Parmi les autres problèmes qu'il est nécessaire de résoudre, il ya la détection de plusieurs objets dans une image. En effet, l'un des problèmes principaux des *snakes* est leur inaptitude à changer de topologie automatiquement. L'une des méthodes qui ont été proposées est la modélisation des contours actifs au moyen des courbes de niveau, qui permet la gestion directe des changements de leur topologie. Cependant, comme il a été vu au 1er chapitre, les contours actifs implicites nécessitent soient de grands temps de calcul, soient leur stabilité dans des situations encombrées est limitée, ce qui réduit leurs champs applicatifs. Ce qui a conduit à chercher des solutions au sein de la représentation paramétrique tout en conservant la rapidité d'exécution de ces modèles. Parmi les méthodes qui ont été développées, on peut trouver celles appliquées dans le suivi d'objets multiples.

2.9 La détection multiple dans les contours actifs paramétriques

Quelques rares tentatives de segmentation de plusieurs objets avec les contours actifs existent dans la littérature.

2.9.1 Les contours actifs multi topologique

Dans l'article [16], une solution au problème du suivi d'objets multiples dans une séquence vidéo couleur, acquise par une caméra mobile, est proposée. Afin de pouvoir gérer des objets multiples aux trajectoires croisées, une capacité à changer de topologie par un mécanisme de scission et fusion a été intégré au contour actif. Ce mécanisme permet respectivement de séparer un contour en deux lorsque deux objets proches sont suivis par un unique contour et s'éloignent l'un de l'autre et pour regrouper deux contours en un seul lorsque deux objets éloignés se rapprochent. Pour plus d'efficacité, cette étape de scission/fusion n'est effectuée sur chaque image qu'après convergence du ou des contours actifs. L'étape de scission permet de diviser le contour actif en plusieurs composants. Une fois

la convergence du contour actif obtenue, certains points du contour peuvent être piégés dans des zones d'énergie externe constante et ne pas se fixer correctement sur les bords des objets suivis, ils proposent donc de supprimer ces points et de scinder les contours actifs aux positions de ces points incorrects. A partir de la liste des points restants, chaque séquence de points successifs est utilisée pour créer un nouveau contour fermé (figure 2.14).

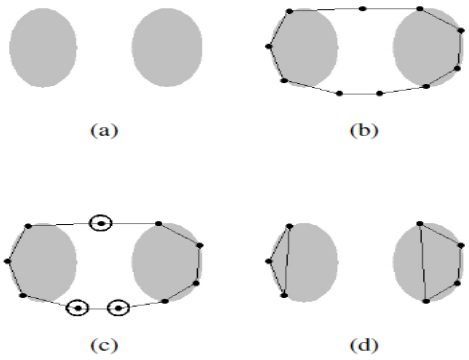


Fig. 2.14. *a) Processus de scission de deux objets ; b) associés à un unique contour; c) ce contour contient des points incorrects ; d) supprimés pour créer les contours finaux.*

L'étape de scission nécessite la définition d'une étape de fusion correspondante. La fusion de deux contours actifs a lieu si les objets suivis par ceux-ci se rapprochent, avant un phénomène d'occlusion. La figure 2.15 illustre les résultats obtenus par cette méthode.

2.9.2 Contours Actifs Paramétriques Multi-Cibles

Dans l'article [17], est présentée l'extension de la méthode des contours actifs paramétriques multicaractéristiques pour permettre la détection et le suivi robuste, précis, et rapide de multiples objets et de leurs parties dans des scènes encombrées. Cette approche est capable de supporter la représentation des domaines ontologiques de type visio-spatiotemporels. En particulier, elle permet d'aider le raisonnement automatique sur des concepts et relations sémantiques inter et intra objets détectés, comme par exemple les relations spatiales directionnelles relatives de type horlogique, représentées au moyen de la logique descriptive. Ils se sont concentrés sur l'application de contours actifs multi-cibles (CAMC) comme sources d'information visuelles pour le domaine ontologique spatiale. Celui-ci comporte essentiellement trois types de relations spatiales : les relations topologiques [48], les relations de positions directionnelles relatives [46] et des relations de distance, taille et aire [47]. Un exemple de résultats obtenu à l'aide de cette méthode sont illustrés dans la figure 2.16.

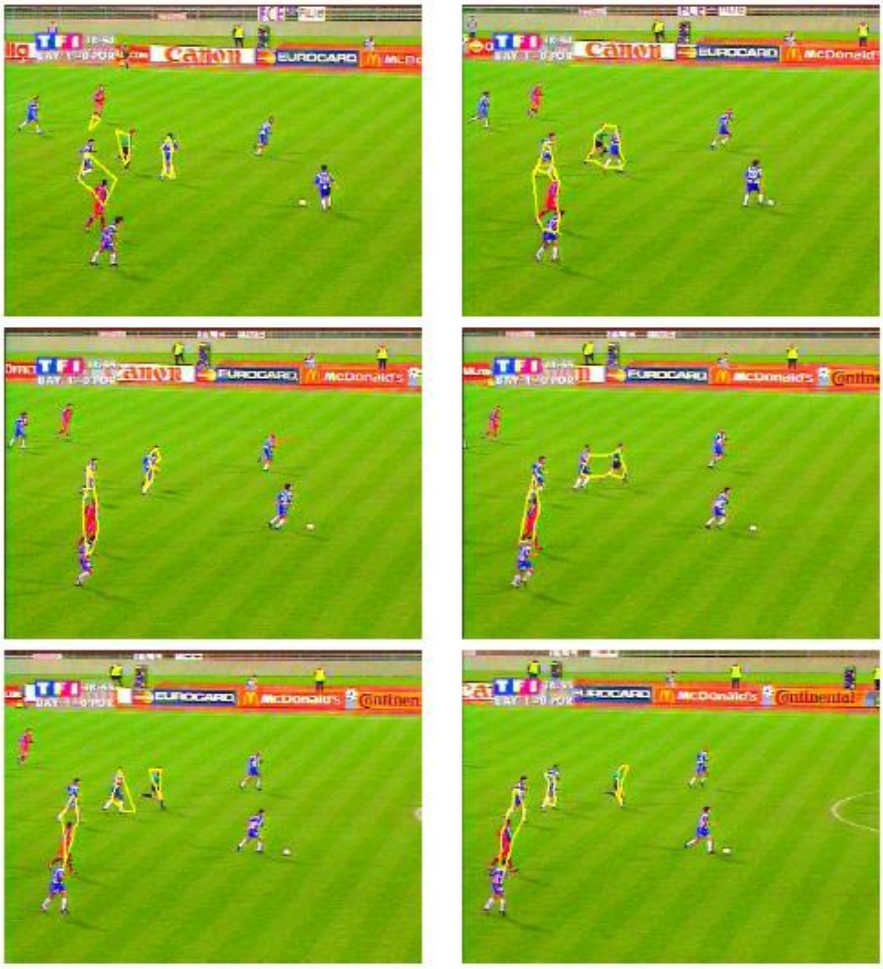
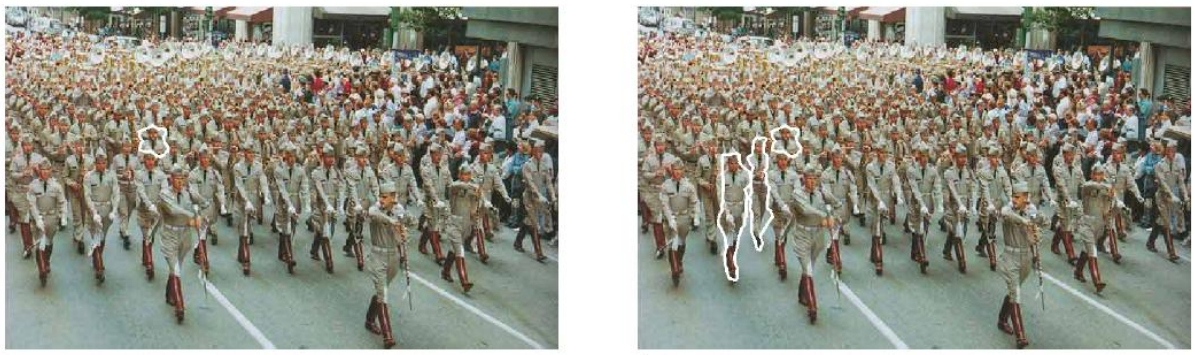


Fig. 2.15. *Intérêt des étapes de scission et fusion dans le cas d'objets proches et d'occlusions temporaires [16].*



(a) calcul du contour actif de l'objet de référence

(b) calcul des contours actifs multi-cibles des objets relatifs externes

Fig. 2.16. *Détection automatique d'objets externes en relation spatiales relatives "à 7 heures" (b) par rapport à l'objet de référence (a), au moyen des contours actifs multi-cibles [17].*

L'approche que nous avons adoptés dans notre mémoire, appelée contours actifs multi objets, s'inspire un peu de celle développée dans les contours multi topologique, mais principalement de [15]. Cette approche consiste en la réalisation de deux étapes : d'abord la détermination des points critiques et des paires de points critiques où serra, ensuite, effectuées

les deux processus de division et de raccordement des points, de manière à obtenir plusieurs contours à partir de l'initialisation d'un seul contour, afin d'entourer les différents objets de l'image. Ces deux étapes seront vues de manière plus exhaustive dans le chapitre 3.

2.10 Conclusion

Dans ce chapitre, nous avons commencer par montrer pourquoi la représentation paramétrique a été utilisée comme base pour la représentation des *snakes* qui est le 1er modèle des contours actifs, puis nous avons définis de manière exhaustive les *snakes,* en commençant par définir les différents éléments de la fonctionnelle d'énergie à minimiser, déterminer son équation d'évolution qui est basée sur une approche lagrangienne, et les différentes méthodes d'implémentation qui sont divisées en deux catégories : méthodes d'optimisation locales et globales. En ce qui concerne ces méthodes d'implémentation, nous avons pu constater que l'utilisation de l'algorithme Greedy pour minimiser l'énergie d'un contour actif, est devenue une alternative assez fréquente à l'approche variationnelle. En effet, nous avons vu que le Greedy permettait de pallier certains inconvénients du *snake* implémenté par la méthode vaiationnelle, tels que les points de contrôles qui ne sont pas toujours régulièrement espacés. De plus, de par son principe inspiré de l'approche gloutonne utilisée en optimisation, il permet de réduire la complexité de l'implémentation et de converger ainsi plus rapidement vers le minimum. Le Greedy est donc considéré comme étant le plus rapide des méthodes d'implémentation. Chose que nous allons essayer de prouver dans le chapitre 3.

Nous avons clôturé ce chapitre par une analyse des caractéristiques des *snakes,* et il en ressort que le *snake* possède beaucoup d'avantages, et l'un de ses principaux avantages est la rapidité d'exécution. Cependant, il possède aussi quelques inconvénients tels que le problème d'initialisation, et de la converge vers les zones concaves. L'autre principal inconvénient est l'incapacité des *snakes* à changer de topologie automatiquement. Ce qui nous a conduits à chercher une solution au sein de la représentation paramétrique tout en conservant la rapidité d'exécution de ces modèles. L'approche que nous avons adoptée appelée contours actifs multi-objets est développée dans le chapitre suivant.

3.1. Introduction

Dans ce chapitre, nous allons mettre en application certaines méthodes étudiées dans les chapitres précédents. En l'occurrence, celles des *snakes* "classique (ou Kass et *al.*) et Greedy" et celle de Chan et Vese. Nous introduiront une nouvelle forme de détection des contours actifs paramétriques qui s'adaptera au changement de topologie, celle-ci se décompose en plusieurs étapes. Des explications sur la méthode de multi-objets sont données dans ce chapitre avant de fournir les différents résultats obtenus. Les différents algorithmes que nous utiliserons ont étés implémenté sous MATLAB 7.8.

3.2. Implémentation des snakes « *Kass et al. et Greedy* » et du modèle de Chan et Vese

Cette première partie contient une brève introduction à la façon dont les contours actifs choisis sont mis en œuvre et comment sont ils gérés, avec une explication de certains détails d'implémentation.

L'utilisateur aura le choix de la méthode et aussi le type d'initialisation: manuelle (initialisation par points) ou automatique (ici l'initialisation étant un cercle dont le centre et le rayon seront introduit par l'utilisateur).

3.2.1. Algorithme de Greedy

Nous pouvons illustrer notre implémentation par l'organigramme de la figure 3.1, montrant un schéma de conception de la segmentation d'images par les contours actifs avec minimisation et déformation suivant l'algorithme de *Greedy*.

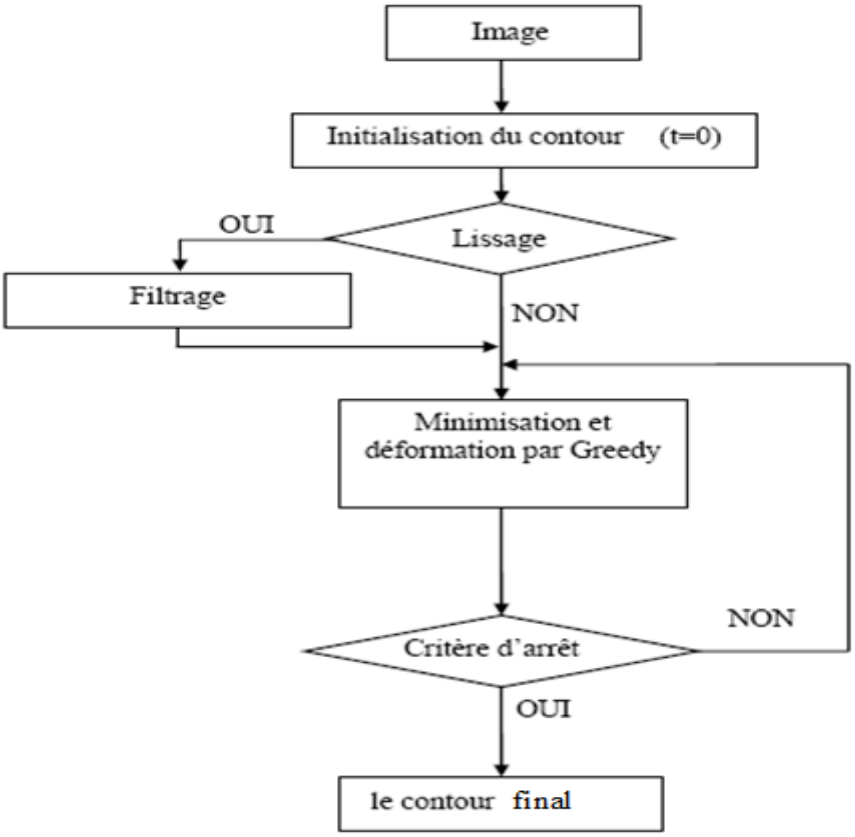


Fig.3.1. *Organigramme de la segmentation d'images par les contours actifs avec minimisation et déformation suivant l'algorithme de Greedy.*

- **Lissage de l'image :** avant tout traitement de détection de contours, il est nécessaire de lisser l'image. Cela conduit à réduire le bruit qui dégrade considérablement la qualité de l'image. Dans notre cas, nous avons utilisé, le filtre de Sobel.
- **Minimisation et déformation par l'algorithme de Greedy :** la méthode de Greedy consiste à faire évoluer le contour actif en minimisant la fonction d'énergie. Pour chaque point du contour actif, on choisit un nombre de voisins pour les quels on va calculer l'énergie. Le point se déplacera alors sur le voisin qui possède l'énergie la plus faible. On cherche, donc, l'ensemble *M* des points pour laquelle l'énergie est minimale. Donc on déplace un point unique pour constituer un nouveau contour actif à chaque itération. Tous les points sont traités successivement lors de chaque itération.
- **Critères d'arrêt :** Les critères d'arrêt sont très importants. En effet, dans certains cas, le contour actif s'arrête pendant un moment sur les contours de l'objet puis dépasse ceux-ci, attiré par des voisins dont l'énergie est plus faible .ou bien il fait des itérations supplémentaires inutiles. Dans le cas du Greedy l'évolution s'arrête quand les points contrôles du *snake* qui se déplacent ne dépassent pas un certains nombre déterminé par

l'utilisateur. Autrement dit, le *snake* est présumé avoir atteint l'énergie minimum quand la plupart des points ont arrêté de se déplacer. Dans notre cas, nous avons considéré que le nombre de points est de 3.

La figure 3.2 nous montre la procédure avec laquelle nous faisons évoluer les points du contour actif à chaque itération.

Remarque :

Afin d'éviter une trop grande dispersion des valeurs des énergies (certaines énergies varient entre 0 et 500 alors que d'autres entre -1 et 0), il est absolument nécessaire de normaliser, c'est-à-dire de diviser la valeur calculée par la plus grande valeur du voisinage. Cela permet ainsi, d'obtenir des variations comprises entre -1 et 1, pour toutes les énergies.

```
% n est le nombre total des points control du snake
% initialiser les paramètres Alpha, beta et gamma.
Faire % la boucle principale fait déplacer les points vers de nouveaux emplacements
    Pour i = 1: n % le premier et dernier point du snake sont le même
        Emin = infini
        Pour j = 1 : m % m est la taille du voisinage
            E(j) = alpha(i) Eela(j) + beta(i) Ecurv(j) + Eimg(j)
            Si E(j) < Emin % trouver les emplacements ayant une énergie minimale
            Emin = E (j)
            jmin = j
        % déplacement des points v(i) vers leurs nouveaux emplacements jmin
        Si jmin n'est pas l'emplacement courant le point se déplace encore
        % le processus ci-dessous détermine quand bêta est relaxé
        Pour i = 1: n % Calcul de la courbure exacte
            c (i) = || v (i) / || v (i) || - v (i+1) / || v (i+1)|| ||^2
        Pour i = 1: n %trouver ou beta doit être relaxé
            Si (c(i) > c(i-1) and c(i) > c(i+1)
            ET c (i) > seuil1
            Et mag(v(i)) > seuil2
                Alors beta(i) = 0 % Relaxer beta si toutes les conditions sont justes
Tant que ptsmoved > seuil3 % arrêt si seulement peu de points sont déplacés suivant le critère.
```

Fig.3.2. *Illustration de l'algorithme de Greedy.*

Remarque :

Dans nos tests, nous avons pris: *Le seuil 1=0.3, le seuil 2= 120 et le seuil 3= 3.*

L'algorithme de Greedy peut se présenté dune maniére trés simple, afin de mieux le comprendre (figure 3.3).

Début
Faire
| *Pour tous les points du snake*
| | ***Pour*** *tous les points du voisinage*
| | | *Calculer les énergies*
| | ***Fin Pour***
| | ***Pour*** *tous les points du voisinage*
| | | *Normalisation*
| | ***Fin Pour***
| | *Minimiser pour obtenir le nouveau point*
| ***Fin Pour***
Jusqu'au *critère d'arrêt*
Fin

Fig.3.3. *Algorithme simplifié de Greedy.*

3.2.2. Algorithme de Kass et *al.*

L'algorithme de Kass et *al.* est très connu. Le lecteur peut se référé à plusieurs articles dans la littérature ou tout simplement à l'article d'origine [35]. Toutefois, il y a lieu d'expliquer certains points :

- **Ré-échantillonnage actif de la courbe :** l'étape ré-échantillonnage, calcule d'abord la distance moyenne entre tous les points du *snake*. Le calcul de la distance moyenne nous permet d'enlever des points de contrôle si ces derniers sont très proches comparés à la distance moyenne, ou bien ajouter des points si la distance entre les points de contrôle est lointaine. Quand un nouveau point est inséré il sera donc positionné au milieu de la ligne reliant les deux points ayant une distance élevée comparé a la distance moyenne. Afin d'éviter trop de calculs inutiles, le ré-échantillonnage n'est pas exécuté après chaque itération mais après un certain nombre d'itérations (dans notre cas après 15 itérations).
- **Critère d'arrêt :** Dans l'article original de Kass et *al.* [35], aucune indication n'a été donnée quant au moment où l'évolution du *snake* doit être arrêtée. plusieurs recherches, dans ce contexte, ont déterminé plusieurs critères [6]. Cependant, comme c'est le cas pour Greedy, nous ne pouvons pas utiliser le nombre de points déplacées comme un critère d'arrêt du fait que la plupart des points contrôles continuent leur avancée même si le minimum d'énergie a été atteint. Cette avancée est cependant très faible. Afin d'éviter ce dépassement du minimum, il est conseillé d'arrêter l'algorithme quand le terme suivant est inferieur à un seuil donné :

$$\frac{\|v(s)^t - v(s)^{t-1}\|}{n} \quad (3.1)$$

$v(s)^t$ Contient les indices des points du *snake* au moment t et $v(s)^{t-1}$ contient les indices des points du *snake* au moment t-1. On divise par n, qui est le nombre total de points du *snake*, dans le but d'éviter des valeurs élevées de$\|v(s)^t - v(s)^{t-1}\|$. C'est ce critère d'arrêt que nous avons adopté.

Remarque :

Dans notre application, le seuil choisi est égale à 0.0008.

3.2.3 Algorithme de Chan et Vese

Nous n'allons pas détailler cet algorithme vu sa simplicité, Le lecteur peut se référer à la thèse [8] pour plus d'informations.

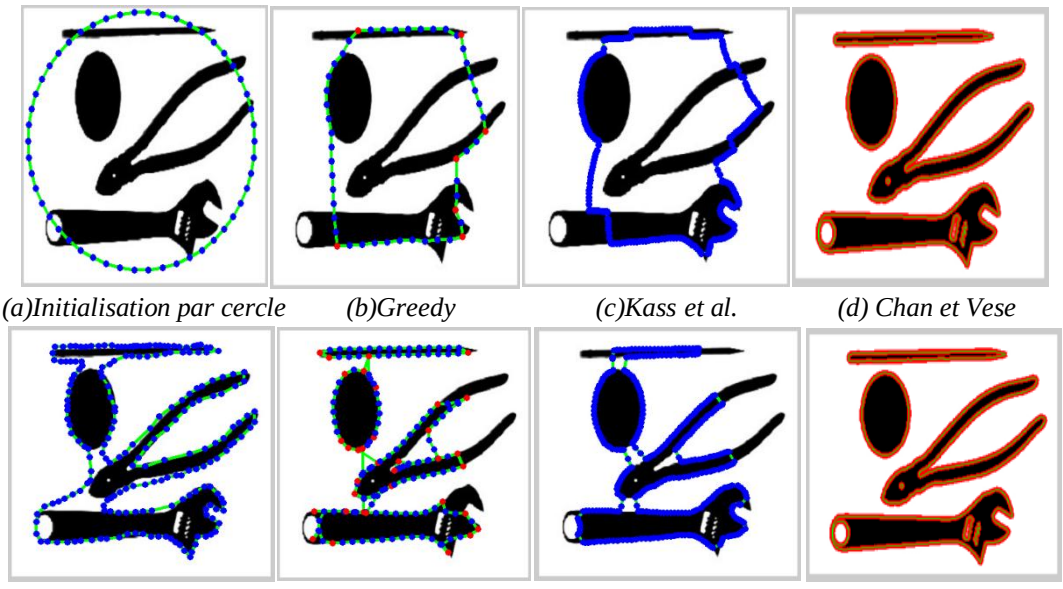
3.3 Tests et résultats

Pour les différents tests nous avons utilisé ces différents paramètres :

- ✓ Pour Kass et al : α=0.05, β=0.0005;
- ✓ Pour Greedy : α=1.2, β=1, γ=1.2, taille du voisinage: 5×5, le seuil d'arrêt est de 3

3.3.1 Test sur Image 1

Dans ce test, nous utilisons une image de taille 350×350, qui se nomme Image1.



(a)Initialisation par cercle *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese*

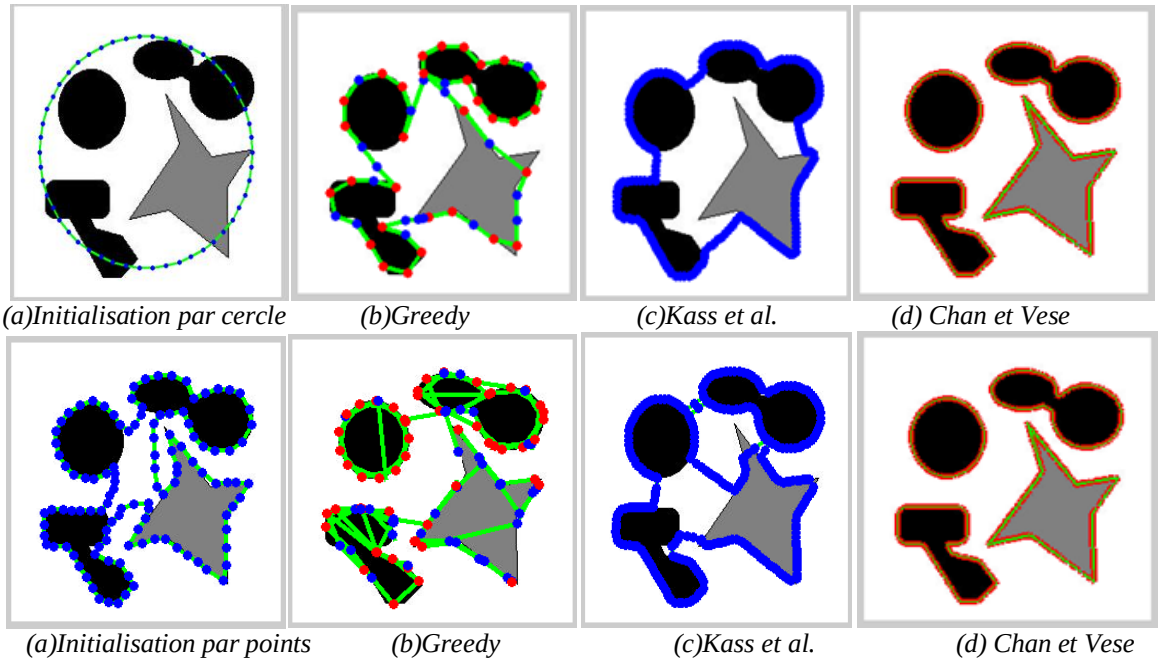
(a)Initialisation par points *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese*

Fig. 3.4. *Résultats de segmentation avec différentes méthodes et différentes initialisation.*

Les résultats obtenus lors des tests effectués sur l'image1 montrent une mauvaise segmentation quand à l'initialisation par cercle, cela est dû à la distance entre les objets du fond et le contour initial. Par contre lors de l'initialisation par points qui est assez proche des objets à détecter, le résultat obtenu est meilleur que son précédent. La segmentation par l'algorithme de Chan et Vese est parfaite et cela pour n'importe quelle initialisation.

3.3.2 Test sur Image 2

Dans ce test, nous utilisons une image de taille 256×256, qui se nomme Image2.



(a)Initialisation par cercle *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese*

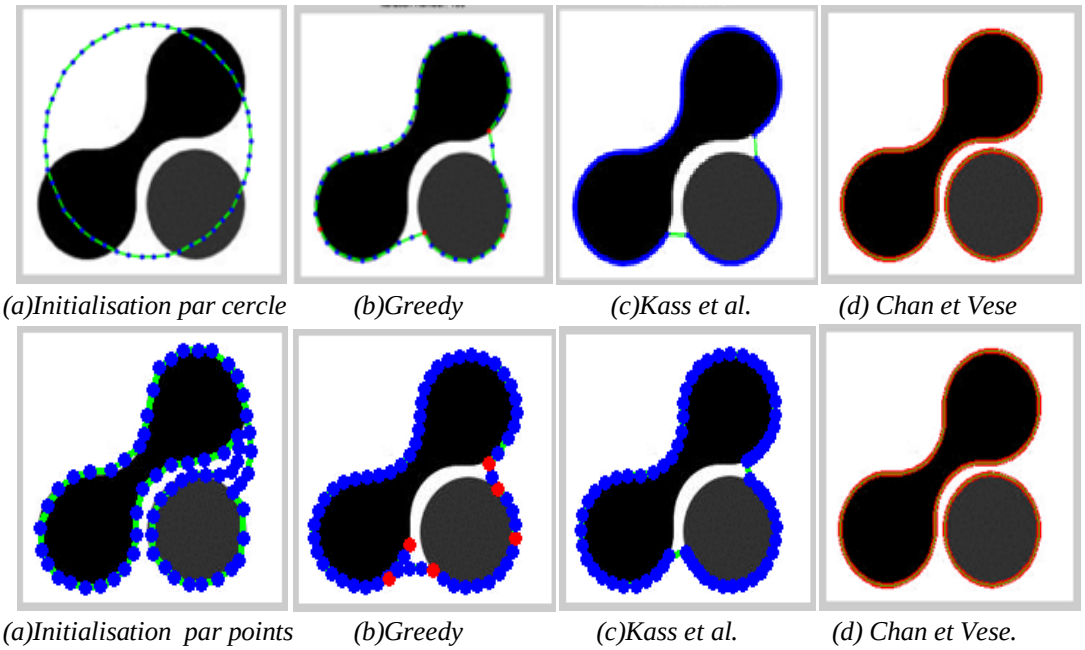
(a)Initialisation par points *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese*

Fig. 3.5. *Résultats de segmentation avec différentes méthodes et différentes initialisations.*

Nous remarquons dans cette image un meilleur résultat quant a l'initialisation par points, cela est dû au rapprochement du *snake* de la zone a détecter. Le résultat obtenu par le greedy est meilleur comparer a celui de Kass et *al.*, cela est dû à la capacité du Greedy à ce fondre dans les concavités. Mais la méthode de Chan et Vese donne un résultat meilleur que ses précédents.

3.3.3 Test sur Image 3

Dans ce test, nous utilisons une image de taille 172×158, qui se nomme Image3.



(a)Initialisation par cercle *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese*

(a)Initialisation par points *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese.*

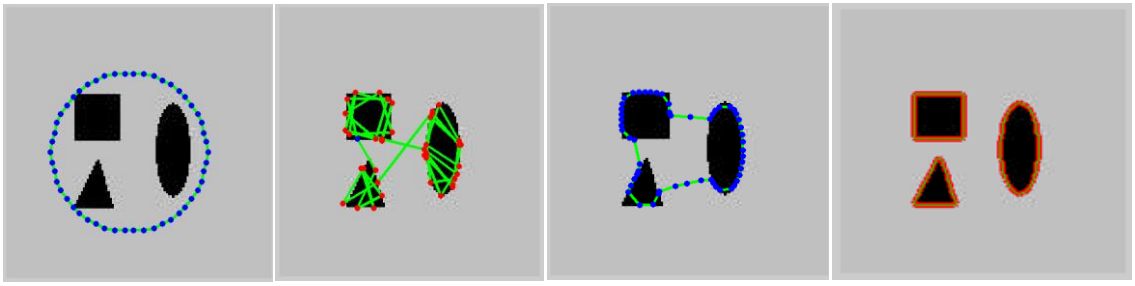
***Fig. 3.6.** Résultats de segmentation avec différentes méthodes et différentes initialisation.*

Dans cette image les résultats obtenus soit pour Kass et *al.* ou Greedy, sont similaires et encourageants lors de l'initialisation par cercle. Même chose pour l'initialisation par points les deux résultats sont pratiquement les mêmes, quoi qu'avec Greedy le *snake* tend à se faufiler dans la concavité. La remarque qu'on peut faire, est que l'algorithme de Chan et Vese ne trouve pas de problème quand à l'évolution dans les zones étroites.

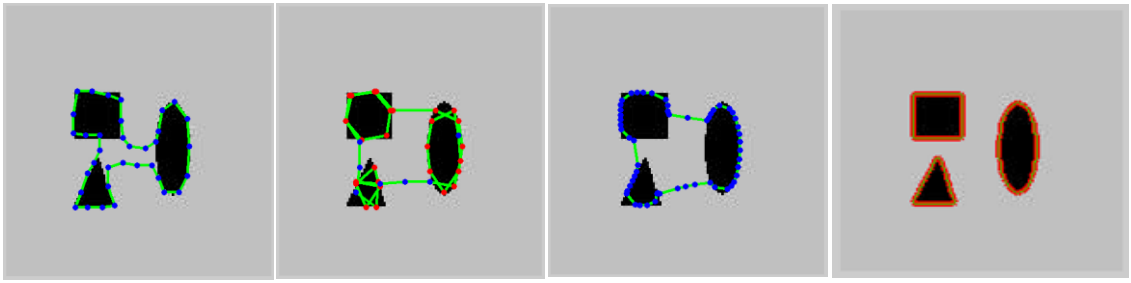
3.3.4 Test sur Image 4

Dans ce test, nous utilisons une image de taille 300×300, qui se nomme Image4.

Le Greedy entoure parfaitement les trois objets de l'image comparant à Kass et *al*. Le chevauchement que nous constatons dans l'image (b) est dû à la taille du voisinage qu'on a choisi. La détection est plus fine avec Greedy que celle avec Kass et *al*. La méthode de Chan et Vese reste toujours meilleur.



(a)Initialisation par cercle *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese*

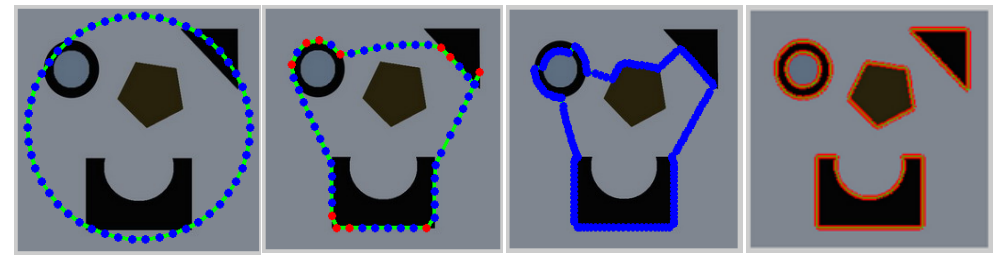


(a)Initialisation par points *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese.*

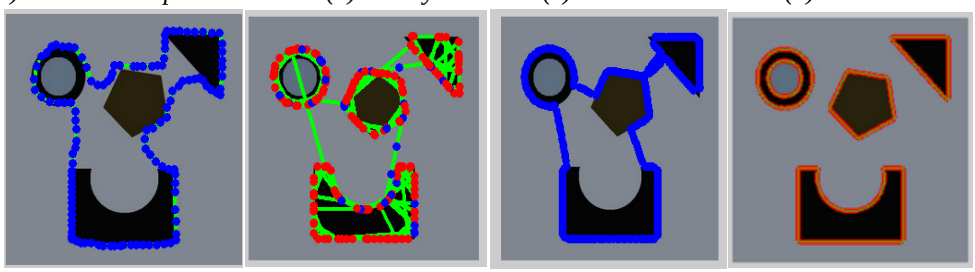
Fig. 3.7. *Résultats de segmentation avec différentes méthodes et différentes initialisations.*

3.3.5 Test sur Image 5

Dans ce test, nous utilisons une image de taille 400×400, qui se nomme Image5.



(a)Initialisation par cercle *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese*



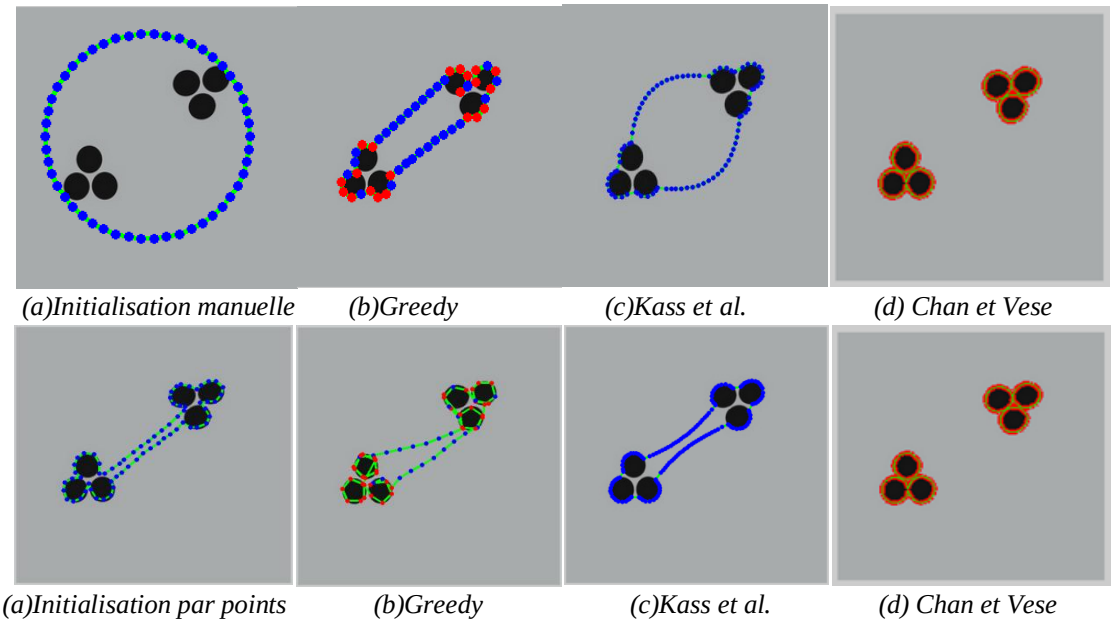
(a)Initialisation par points *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese.*

Fig. 3.8. *Résultats de segmentation avec différentes méthodes et différentes initialisations.*

Dans cette image, l'initialisation par cercle nous a donné un mauvais résultat soit pour Greedy ou Kass. Par contre l'initialisation par point, qui est proche des objets à détecter donne un meilleur résultat surtout pour la méthode de Greedy. La segmentation par Chan et vese est la plus fine.

3.3.6 Test sur Image 6

Dans ce test, nous utilisons une image de taille 256×256, qui se nomme Image 7.



(a)Initialisation manuelle *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese*

(a)Initialisation par points *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese*

Fig. 3.9. *Résultats de segmentation avec différentes méthodes et différentes initialisations.*

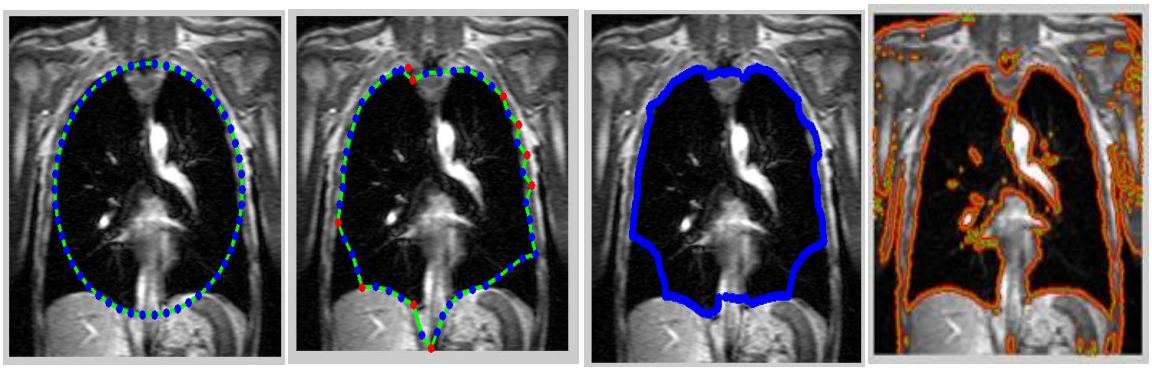
La segmentation de cette image montre un meilleur résultat pour Greedy et un résultat encourageant pour Kass et *al.* du fait qu'avec Greedy la détection s'est faite même dans les coins les plus étroits. Même chose pour l'initialisation par points, la détection par l'algorithme de Greedy est plus performante. Mais la détection par l'algorithme de Chan et vese reste meilleure.

3.3.7 Test sur Image 7

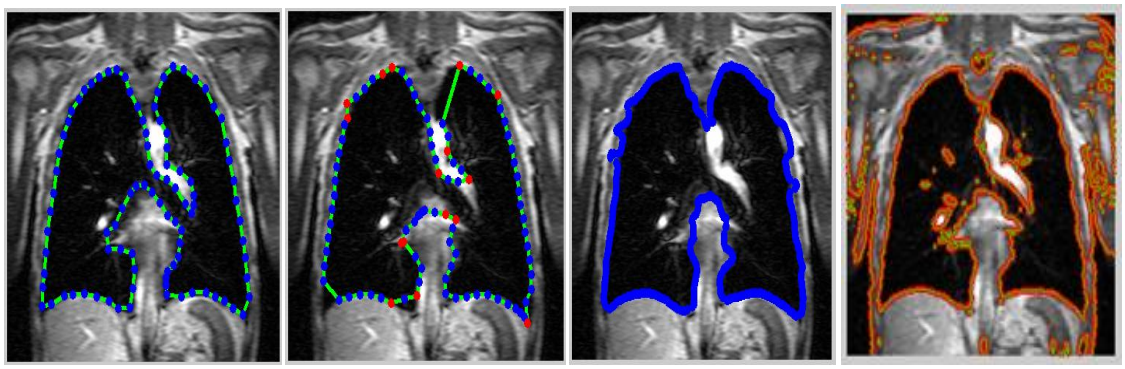
Dans ce test, nous utilisons une image réelle de taille 354×331, qui se nomme Image 8. Pour cette image médicale les deux résultats sont du moins satisfaisants pour l'initialisation par cercle, cela est dû à la détection d'un seul objet. La segmentation est d'autant meilleure avec l'initialisation par cercle, quoi que le résultat avec le Greedy snake est plus bon. L'algorithme de Chan et vese est nettement meilleur quand a la détection des zones les plus fines.

Remarque :

L'interprétation des résultats n'a pas tenu compte du temps convergence ou de nombre d'itérations.



(a)Initialisation par cercle *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese*



(a)Initialisation par points *(b)Greedy* *(c)Kass et al.* *(d) Chan et Vese*

Fig. 3.10. *Résultats de segmentation avec différentes méthodes et différentes initialisations.*

3.3.5 Interprétation des résultats

Avant de donner une interprétation des différents résultats, nous avons dressé deux tableaux. Le tableau 3.1 regroupe tous les résultats obtenus lors des tests suivant les algorithmes de Kass et *al.* et Greedy et cela en utilisant deux initialisations différentes. Tandis que le tableau 3.2 renferme tous les résultats numériques obtenus lors de l'exécution de la méthode Chan et Vese.

En comparant les résultats obtenus du tableau 3.1, nous constatons que le temps de calcul est plus important lors de l'initialisation par points, aussi la convergence est plus rapide avec l'initialisation par cercle. Mais la chose que nous pouvons dire est que la qualité de détection des contours est bien plus fine lorsque l'initialisation par points est effectué, cela revient à l'initialisation proche des contours désirés.

	Initialisation de cercle				Initialisation de points			
	Greedy		Kass		Greedy		Kass	
	Temps (s)	Itérations	Temps (s)	Itération	Temps (s)	Itérations	Temps (s)	Itération
Image1 (350×350)	**2.9578**	**148**	**2.0736**	**1600**	**17.1586**	**284**	**1.9374**	**606**
Image2 (256×256)	**3.3323**	**158**	**1.4**	**1200**	**14.3541**	**400**	**1.7916**	**808**
Image3 (172×158)	**2.6352**	**135**	**3.9668**	**721**	**6.5181**	**284**	**0.7404**	**981**
Image4 (300×300)	**6.8191**	**357**	**0.9633**	**800**	**1.6798**	**106**	**0.8246**	**890**
Image5 (400×400)	**2.7224**	**141**	**1.0385**	**1010**	**30.7254**	**488**	**2.0572**	**1000**
Image6 (256×256)	**2.8544**	**142**	**0.8195**	**789**	**5.3243**	**218**	**1.2623**	**1010**
Image7 (354×331)	**2.2138**	**99**	**1.7902**	**1700**	**4.7218**	**136**	**4.4848**	**960**

Tab.3.1. *Le nombre d'itération et le temps de convergence des snakes Greedy et Kass sur plusieurs images.*

	CHAN ET VESE	
	Temps (s)	Itérations
Image 1	**107.0477**	**1744**
Image 2	**36.5760**	**728**
Image 3	**6.9309**	**205**
Image 4	**39.5204**	**2000**
Image 5	**74.5659**	**1600**
Image 6	**29.0954**	**700**
Image 7	**165.5507**	**3000**

Tab.3.2. *Le nombre d'itération et le temps de convergence du modéle de Chan et Vese sur plusieurs images.*

La comparaison des résultats à la fois du Kass et *al.* et du Greedy, nous amène a conclure que le Kass et *al.* s*nake* donne de meilleurs résultats globaux. cela est dû à plusieurs raisons. L'étape de ré-échantillonnage, qui fait augmenter les points du *snake*, ce qui nous permet d'avoir des contours plus fin. Toutefois, si nous essayons d'augmenter le nombre de points du *snake* dans le Greedy, nous aurons un risque de chevauchement entre les points, cela conduit à des effets indésirables entre autre augmentation de la distance entre les points (faible rigidité). Le problème que nous avons constaté est la façon dont l'initialisation du contour se fait, aussi le problème qui revient souvent est le fait que les paramètres du *snake*

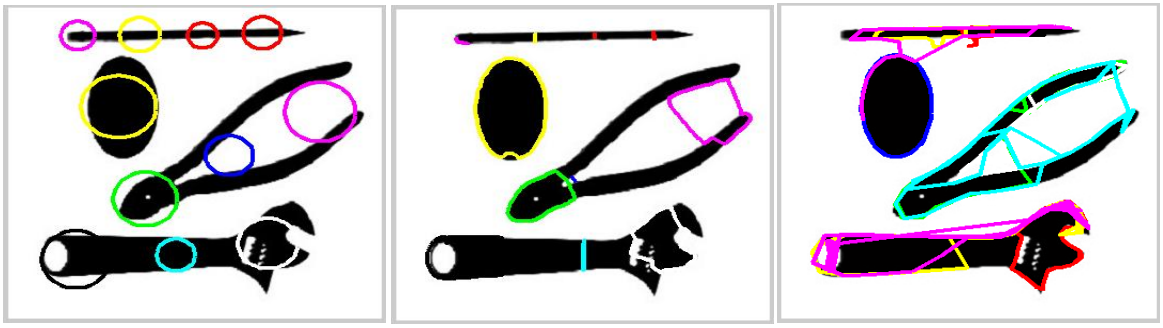
doivent être souvent ajustés pour chaque nouveau type d'image. Trouver les valeurs appropriées pour les paramètres de réglage s'appuie bien sûr sur des essais et des erreurs.

Dans la phase d'exécution du Chan et Vese nous avons constatés une très bonne segmentation. En effet ce modèle peut détecter plusieurs objets avec une simple initialisation et à n'importe quelle endroit dans l'image. Cette méthode allie efficacité et simplicité, notamment, dans son implémentation et surtout pour son extension aux images en couleur. Cependant, cette méthode présente un certain nombre d'inconvénients, mais le plus important est le temps d'exécution qui est assez lent, notamment pour les grandes images. En comparant la méthode des Chan et Vese à celle des *snakes* (Kass et *al.et Greedy)*, nous remarquons que la convergence et le temps d'exécution et plus important pour la méthode de Chan et Vese.

3.4. Introduction aux multi-objets

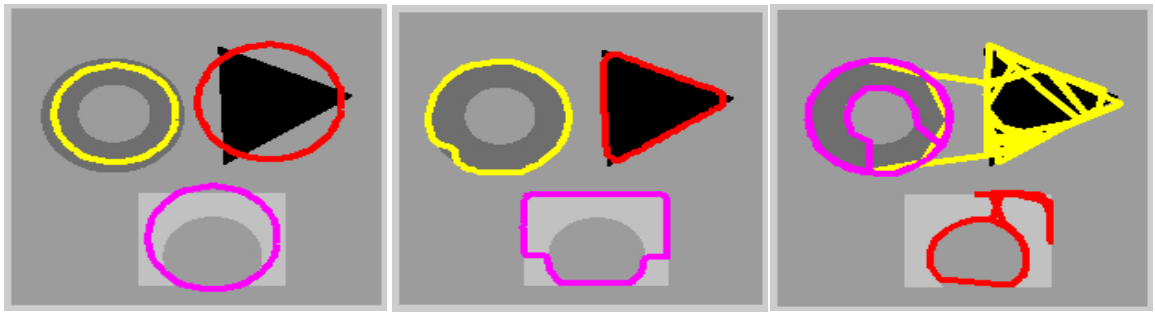
un constat que nous pouvons faire de la section précédant est que le *snake* (Greedy ou Kass et *al.)*, présente un avantage majeur quand au temps de convergence, seulement il faut essayer de résoudre le problème de la détection de plusieurs objets dans la même image.

Pour faire face à ce problème, l'idée la plus simple est d'initialiser plusieurs contours sur les images à multi-objets (figure 3.10, 3.11, 3.12, 3.13, 3.14 et 3.15).



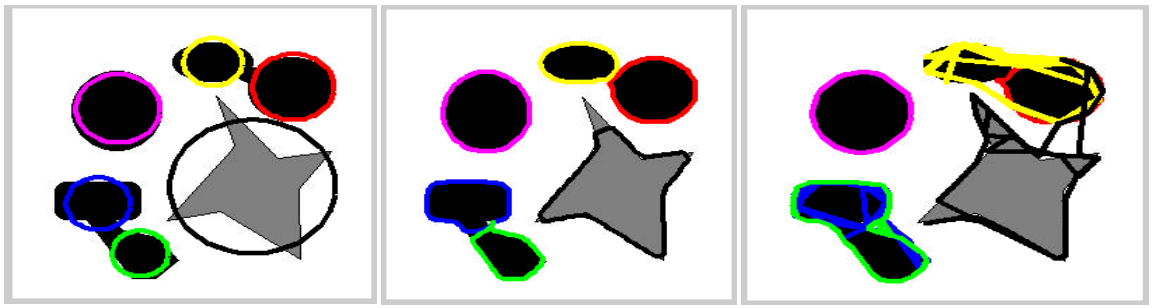
(a)Initialisation manuelle *(b) Kass et al.* *(c) Greedy*

Fig. 3.10. *Résultats de segmentation multi-objets sur l'image1.*



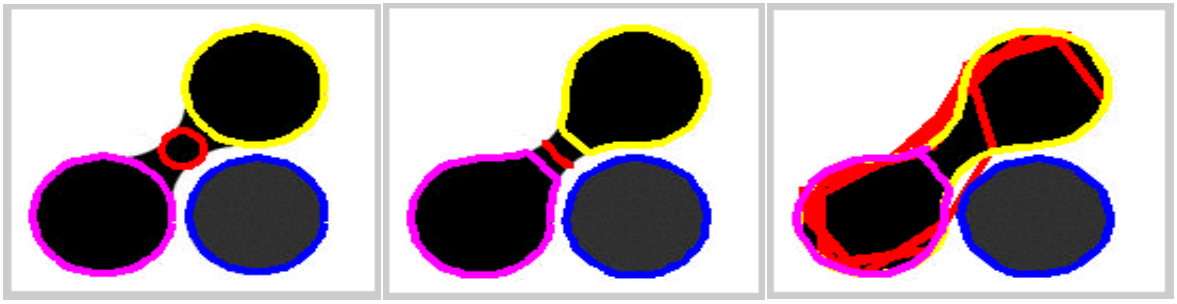
(a)Initialisation manuelle *(b) Kass et al.* *(c) Greedy*

Fig. 3.11. *Résultats de segmentation multi-objets sur l'image 2.*



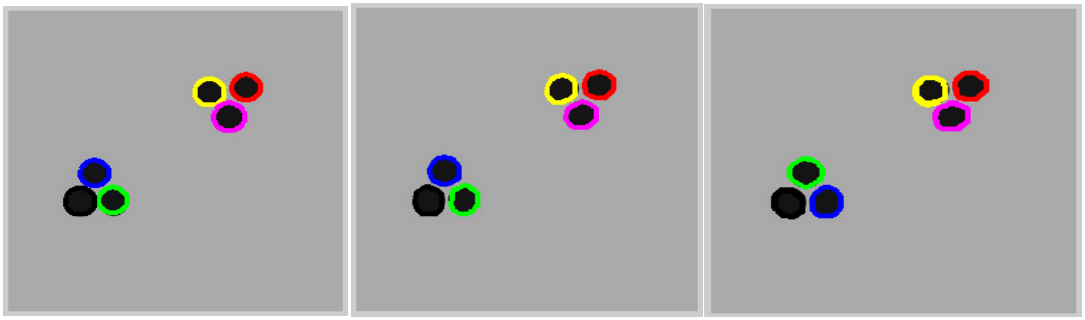
(a)Initialisation manuelle *(b) Kass et al.* *(c) Greedy*

Fig. 3.12. *Résultats de segmentation multi-objets sur l'image 3.*



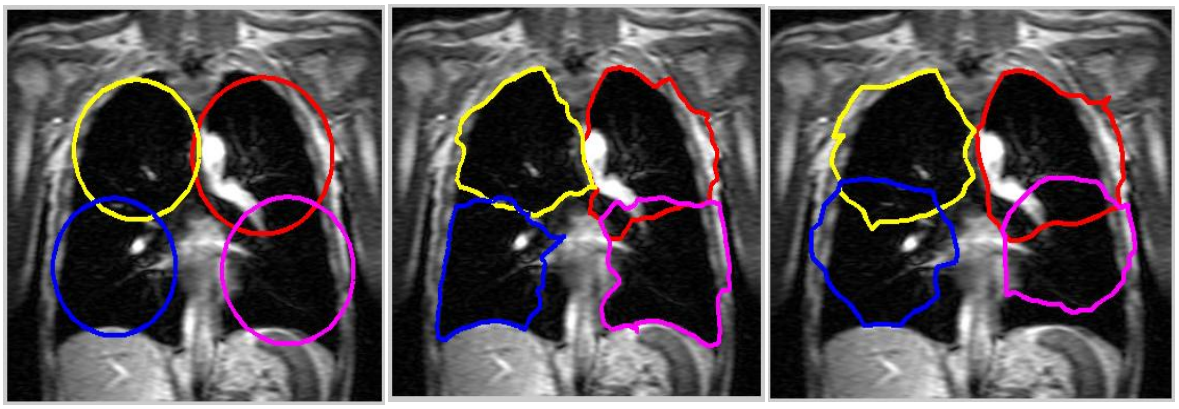
(a)Initialisation manuelle *(b)Kass et al.* *(c) Greedy*

Fig. 3.13. *Résultats de segmentation multi-objets sur l'image 4.*



(a)Initialisation manuelle *(b) Kass et al.* *(c)Greedy*

Fig. 3.14. *Résultats de segmentation multi-objets sur l'image 5.*



(a)Initialisation *(b) Kass et al.* *(c) Greedy*

Fig. 3.15. *Résultats de segmentation multi-objets sur l'image 6.*

Dans la figure 3.10, nous remarquons une mauvaise segmentation avec l'algorithme de Kass. Cela est dû à l'étroitesse des zones de détection. Le résultat obtenu avec l'algorithme de Greedy est meilleur que son précédant, quoi que le problème qui ce pose est celui de la fusion des différents contours initialisés. Dans le deuxième test (figure 3.11), nous remarquons que la segmentation par l'algorithme de Greedy donne un résultat moins bon qu'avec Kass. Cela est dû à la taille du voisinage que nous avons initialisé. Même chose pour le résultat de la (figure 3.12).tandis que, dans la (figure.3.13) le kass donne un meilleur résultat que celui du Greedy. La détection de contour par l'algorithme de Greedy montre une redondance des points des *snakes* initialisés. Pour la figure 3.14, nous obtenons le même résultat soit pour Greedy ou Kass. Dans la (figure 3.15.) le Kass converge mieux que le Greedy. Une étape de fusion et de suppression des contours redondants est nécessaire dans ce cas.

Remarque :
L'interprétation des images ne tient pas compte du temps convergence.

Le tableau 3.3, donne les différentes valeurs concernant le nombre de contours initialisés et le temps de convergence.

	Greedy		**Kass**	
	Nombre de cercles initialisés	**Temps(s)**	**Nombre de cercles initialisés**	**Temps(s)**
Image 1	11	35.8893	11	29.1339
Image 2	3	11.0761	3	8.7470
Image 3	6	16.4271	6	11.3806
Image 4	4	8.9420	4	0.8917
Image 5	6	16.3186	6	16.2828
Image 6	4	3.0409	4	35.0012

Tab.3.3. *Le temps de convergence des snakes Greedy et Kass sur les différentes images avec différents nombres de contours initiaux.*

Le temps de calcul est meilleur avec l'algorithme de Kass que celui de Greedy sauf pour l'image 5 et l'image 6. L'image 5 donne le même résultat pour l'algorithme de Kass et celui de Greedy. Pour l'image 6 le temps est plus important pour Kass.

L'initialisation de plusieurs contours reste une solution prometteuse pour les contours actifs paramétriques. Néanmoins le problème qui ce pose est de trouver une certaine formule de fusion des contours. Pour cela, une autre solution pour les multi-objets est très importante. Ainsi, nous avons essayé de développer une approche qui, à partir d'un seul contour initial, permet de détecter plusieurs objets dans une image. Cette approche est constituée de plusieurs

étapes. Ces parties viendront après avoir exécuté le *snake* (Greedy ou Kass et *al.*) avec initialisation sous forme d'un seul contour.

3.4.1. Détermination des points critiques et des paires de points critiques

Cette étape consiste à déterminer la zone du contour où la division doit être effectuée pour former des sous contours. Pour cela, les contours sont, d'abord, classés en un segment contour et un segment de non-contour (figure 3.16), qui sont déterminés par le calcul de l'énergie de l'image le long du contour. Si l'énergie de l'image en un point est inferieure à un seuil prédéfini, alors ce point est considéré comme un point de non-contour, sinon comme un point de contour.

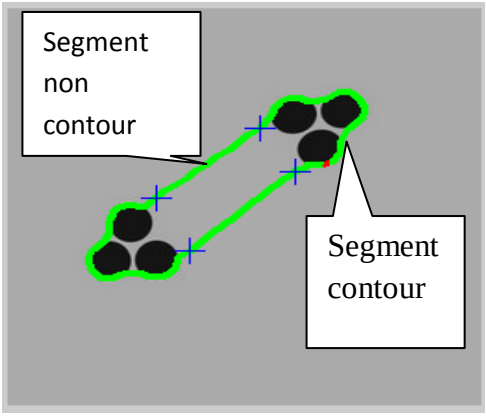


Fig. 3.16. *Séparation du contour actif en 2 segments.*

Le point critique est défini comme étant la frontière entre le segment contour et le segment de non-contour (figure 3.17). La détermination d'un point de contour est illustrée comme suit:

$$if\ (E(v_i) > Es, E(v_{i-1}) > Es\ et\ E(v_{i+1}) < Es)$$
$$\text{Ou} \qquad (3.2)$$
$$if\ (E(v_i) > Es, E(v_{i-1}) < Es\ et\ E(v_{i+1}) > Es)$$

Alors le point v_i est marqué comme un point critique, où $E(v_i)$ est l'énergie de l'image dans le voisinage de v_i , et Es est un seuil prédéfini. Un certain nombre de points critiques sont alors identifiés et notés comme une séquence :

$$C = \{c_i = (x, y) | \, i = 0,1,2, \ldots n-1\}, \qquad (3.3)$$

Où c_i est le point critique, et n est le nombre de points critiques qui sera égale à zéro ou à un chiffre pair pour un contour fermé. Au cours du processus d'itération, chacun des points critiques sera vérifié dans l'ordre. Des paires de points critiques seront alors marqués comme

points à connecter où les opérations de séparations et de raccordement seront effectuées afin de former deux contours (figure 3.17). La procédure d'identification de ces paires est effectuée comme suit :

Etape 1 : mettre $i = 0$ *comme point de départ du point critique* c_i .
Etape 2 : mettre $j = i + 1$ *comme étant un autre point critique*c_j.
Etape 3 : calcul de la distance d_{ij} *entre les deux points critiques* c_i *et* c_j.
Etape 4 : si d_{ij} *est inferieur à un seuil,* c_i *et* c_j *sont marqués comme paire de points à relier.*
Etape 5 : si $j < n$, *alors* $j = j + 2$ *et retour à l'étape 3, sinon aller à l'étape 6.*
Etape 6 : si $i < n$, *alors* $i = i + 1$ *et retour à l'étape 2, sinon aller à l'étape 7.*
Etape 7 : fin.

Remarque :

Dans notre travail nous avons choisi de prendre comme seuil (étape 4), la distance euclidienne minimale entre deux points critiques.

- *Si* $d_{ij} <$ *seuil, on aura des paires dont les points critiques seront proches.*
- *Sinon, on aura des paires dont les points critiques sont espacés.*

Ce choix sera effectué selon l'image.

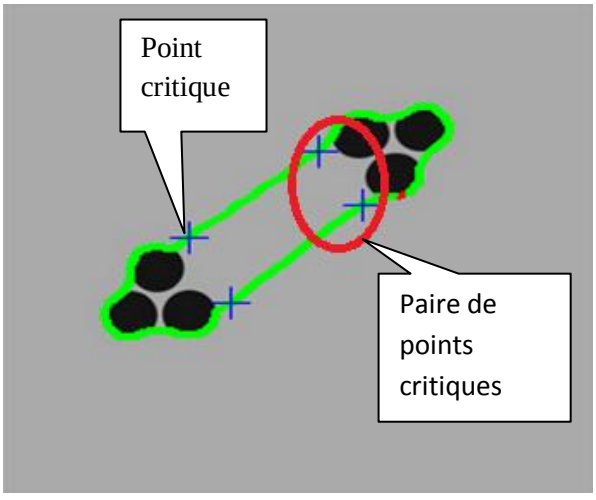


Fig.3.17. *Détection des points critiques et des paires de points critiques*

3.4.2. Processus de séparation et de raccordement

Après avoir obtenu les paires de points critiques qui sont proches les uns des autres, l'opération de séparation et de raccordement est effectuée (figure 3.18 et 3.19).

Soit la paire des points à connecter défini comme suit :

$$S = \{s_k = (v_i, v_j) | \; i < j < n, \; k < m\} \tag{3.4}$$

Où s_k est la k nième paire, n est le nombre de points du *snake*, et m est le nombre de paires connectées. La transition du segment contour au segment non contour s'effectue au point v_i, tandis que la transition du segment non contour au segment contour s'effectue au point v_j . Cela nous indique le sens d'orientation du contour. La procédure de séparation des points du segment contour du segment non contour, et celle de raccordement des points critiques d'une paire, est effectué comme suit :

Etape 1 : mettre $k = 0$.
Etape 2 : séparation : le contour est divisé entre le point v_i *et* v_{i+1} *et entre le point* v_j *et* v_{j-1}.
Etape 3 : raccordement des points : le contour entre les points v_i *et* v_j, *et le contour entre* v_{i+1} *et* v_{j-1} *sont connectés. On obtient, alors, après ce processus de raccordement deux contours.*
Etape 4 : si $k < m$, *alors* $k = k + 1$ *et retour à l'étape 2, sinon aller à l'étape 5.*
Etape 5 : fin.

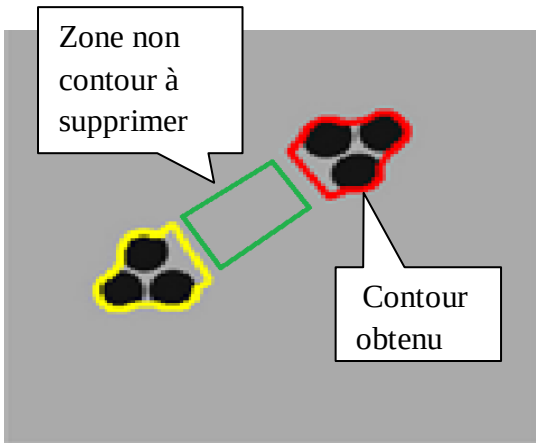


Fig.3.18. *Etape transitoire entre la suppression de zone contour et détection d'objets.*

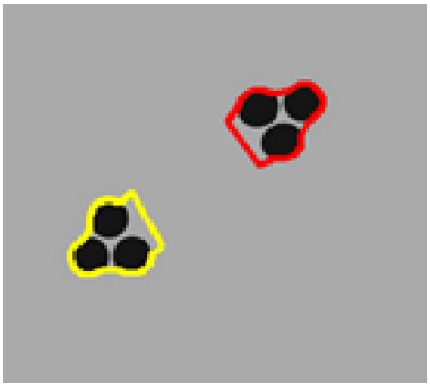


Fig.3.19. *Illustration de la séparation d'objets*

Remarque:

Après séparation d'objets nous avons pensés a faire une continuité de convergence a l'aide des algorithmes de Kass ou de Greedy , afin d'obtenir un meilleur résultat. La figure 3.20 nous montre une détection d'objet après continuité de convergence.

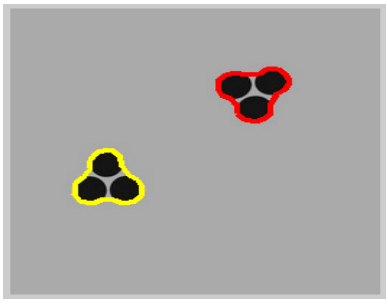
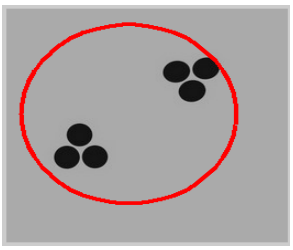


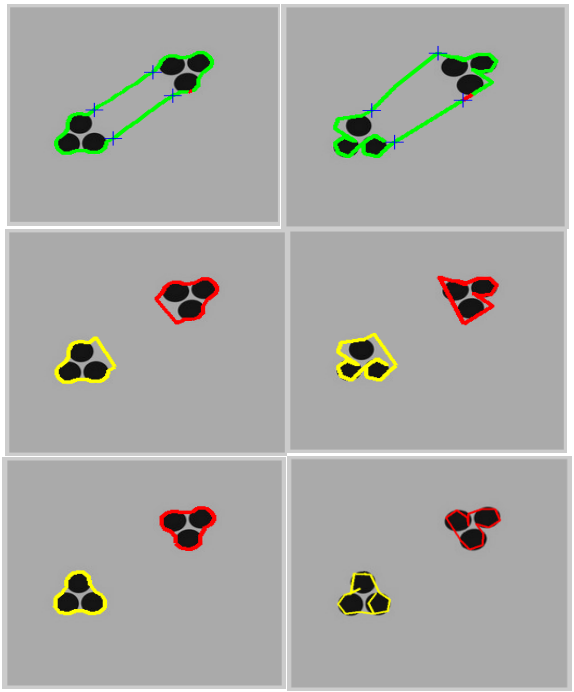
Fig.3.20. *Résultat obtenu après continuité de convergence.*

3.4.3. Tests et résultats

Dans cette partie, nous montrerons des résultats d'application des différentes étapes décrites dans la section précédente (figures 3.21, 3.22, 3.23, 3.24, 3.25 et 3.26).



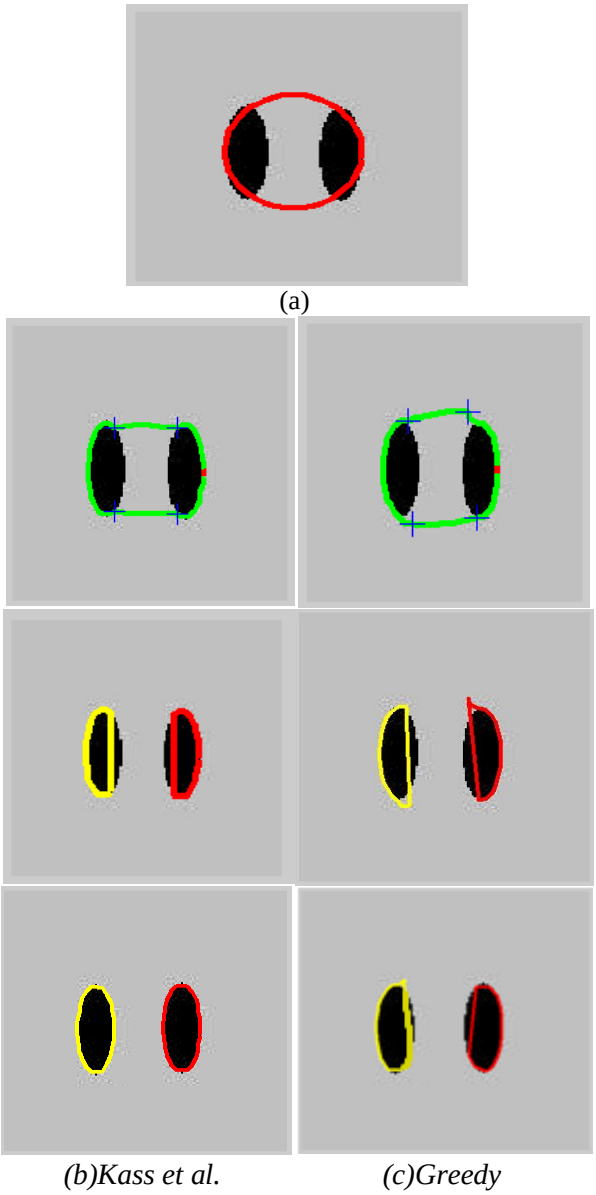
(a)Initialisation



(b)Kass et al. *(c)Greedy*

Fig. 3.21. *Résultats de segmentation multi-objets: la deuxième ligne représente la détection des points critique, la troisième ligne la séparation et la dernière ligne la continuité de convergence*

Le test effectué par l'algorithme de Kass montre une bonne segmentation d'image. néanmoins, le *snake* n'a pas évolué dans les concavités étroites. Tandis que celui de Greedy montre une petite amélioration après continuité de convergence quand à l'évolution vers les concavités. Pour ce test la valeur du seuil distance est de 100.



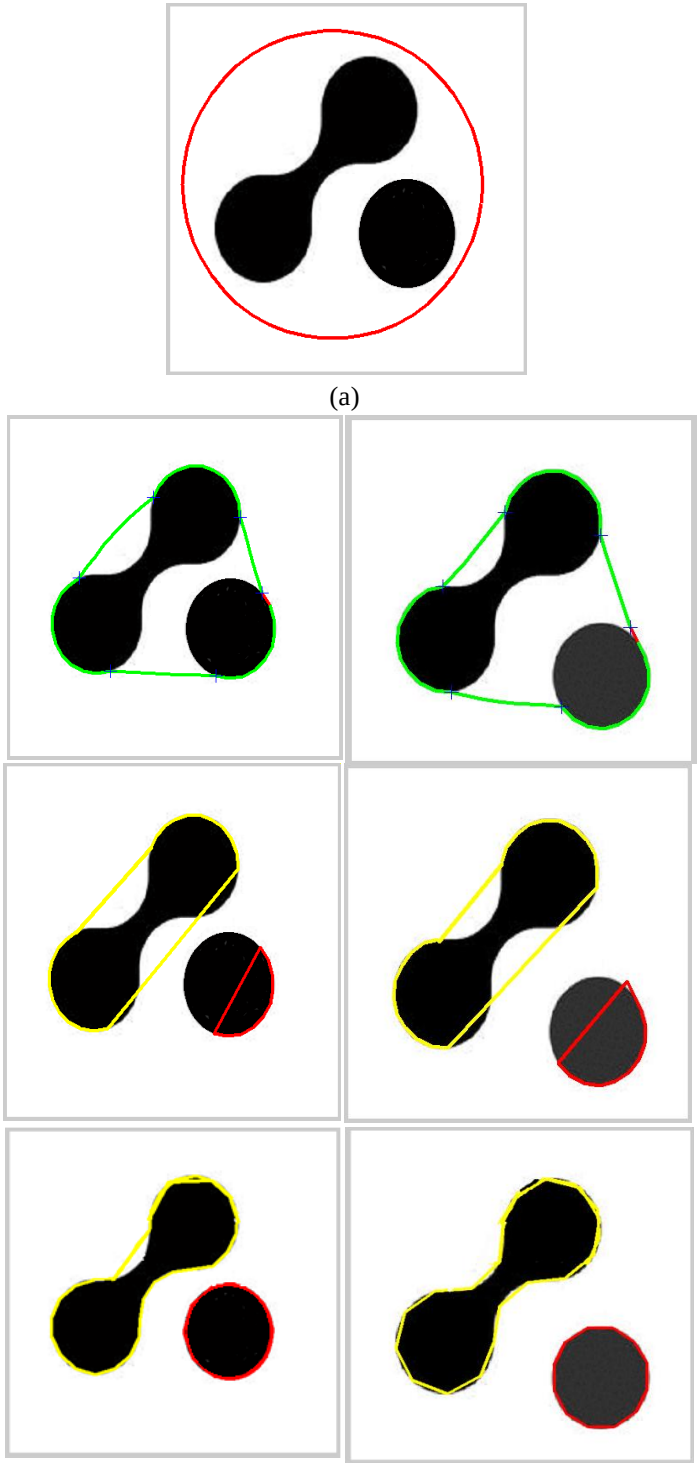
(a)

(b)Kass et al. *(c)Greedy*

Fig. 3.22. *Résultats de segmentation multi-objets: la deuxième ligne représente la détection des points critique, la troisième ligne la séparation et la dernière ligne la continuité de convergence.*

Ce test nous montre une parfaite détection de contour et cela bien sûr après continuité de convergence. La détection des contours par l'algorithme de Greedy montre une détection

moins fine que celle de Kass, mais du moins acceptable. Pour ce test la valeur du seuil distance est de 100.



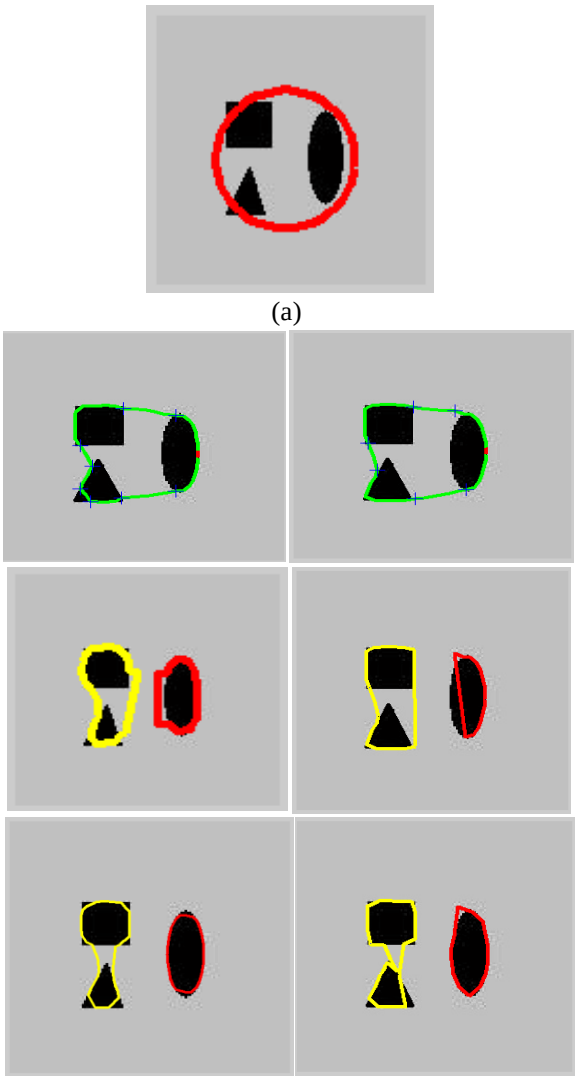
(a)

(b)Kass et al. *(c)Greedy*

Fig. 3.23. *Résultats de segmentation multi-objets: la deuxième ligne représente la détection des points critique, la troisième ligne la séparation et la dernière ligne la continuité de convergence.*

La détection par l'algorithme de Kass est acceptable après convergence. La non détection de toute la partie supérieure est dû aussi à l'incapacité de Kass à détecter les concavités. Parfaite détection pour l'algorithme de Greedy, comparé a son précédant. La seule chose qui est remarquable, est que les contours ne sont pas réellement circulaires. Cela est dû à la relaxation des paramètres beta.

Remarque : *pour la relaxation du paramètre beta, ce référer au chapitre 2.*



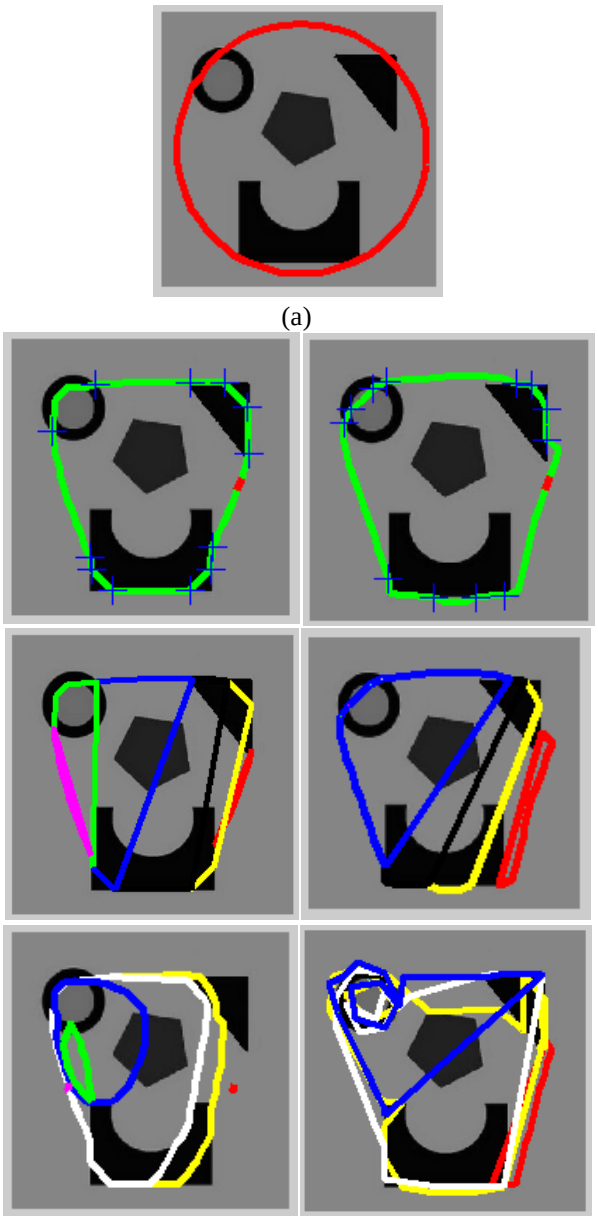
(a)

(b)Kass et al. *(c)Greedy*

Fig. 3.24. *Résultats de segmentation multi-objets: la deuxième ligne représente la détection des points critique, la troisième ligne la séparation et la dernière ligne la continuité de convergence.*

Les tests effectués dans la figure 3.24, donne un mauvais résultat quant à la détection des objets se situant à gauche de l'image. Cela est illustré par la par la non séparation de ces derniers. La fonction distance que nous avons mis en œuvre est dans l'incapacité de faire face

à ce problème. Donc cela doit être amélioré pour détecter a la fois les distances proches et lointaines sans trouver de confusion.



(a)

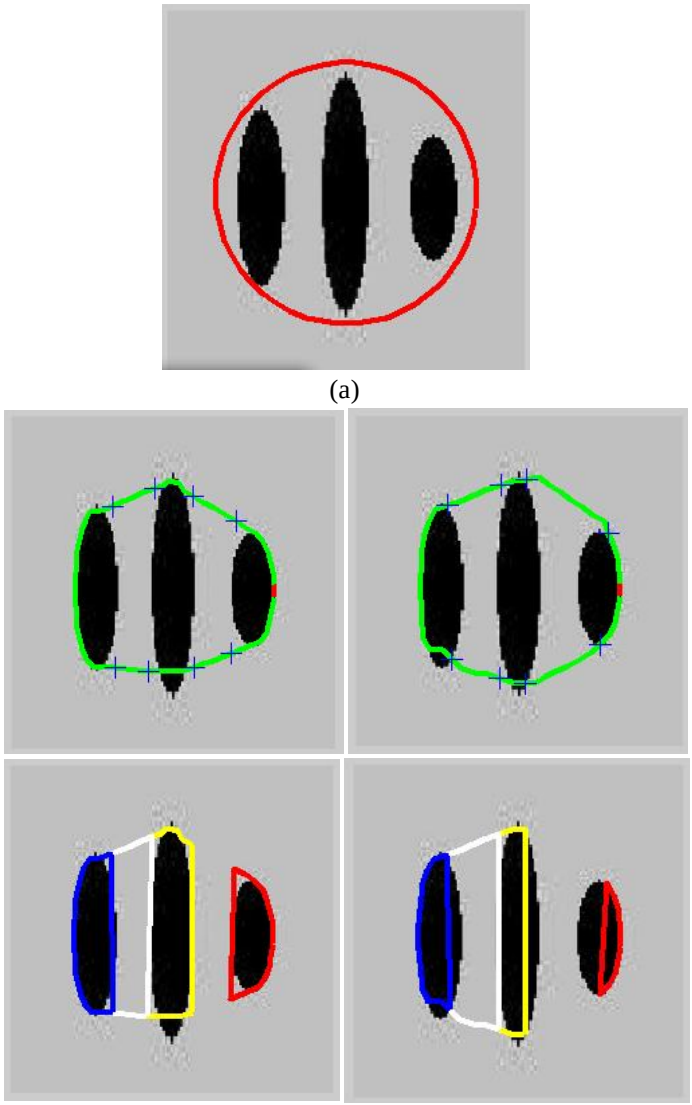
(b)Kass et al. *(c)Greedy*

Fig. 3.26. *Résultats de segmentation multi-objets: la deuxième ligne représente la détection des points critique, la troisième ligne la séparation et la dernière ligne la continuité de convergence.*

Du fait de la détection de plusieurs points critiques, notre algorithme présente des handicapes quant à la sélection des paires. Cela est dû aux différentes distances calculées entre tous les points critiques. La fonction distance que nous avons mis en œuvre est dans

l'incapacité de faire face à ce problème, cela nous servira de perspective pour la continuité de notre travail.

Pour la figure 3.27, la continuité de convergence n'est pas prise en considération.



(a)

(b)Kass et al. *(c)Greedy*

Fig. 3.26. *Résultats de segmentation multi-objets: la deuxième ligne représente la détection des points critique, la troisième ligne la séparation.*

Après avoir fait le test de la (figure 3.26), on remarque l'apparition d'une zone non contour en blanc. Cela est dû l'apparition de plusieurs points critiques, donc, confusion entre les distances des points. Comme on la déjà cité l'idée est de l'améliorer la distance pour qu'elle arrive à distinguer les petites et grande distance. Aussi essayer de trouver une nouvelle fonction d'énergie que celle du gradient qui est très simple pour ce genre de détection.

Le tableau 3.4 englobe tous les résultats obtenus durant l'application.

	Greedy			Kass		
	Temps (s)	**Points Critiques**	**Es**	**Temps (s)**	**Points critiques**	**Es**
Image1 (256×256)	**4.1108**	**4**	**10**	**12.8742**	**4**	**10**
Image2 (400×400)	**2.3790**	**4**	**10**	**1.5783**	**4**	**1**
Image3 (172×158)	**5.1619**	**6**	**10**	**3.8937**	**6**	**10**
Image4 (300×300)	**2.9852**	**6**	**7**	**4.9412**	**8**	**7**
Image5 (400×400)	**7.3599**	**12**	**10**	**2.6750**	**12**	**10**
Image6 (400×400)	**1.7549**	**8**	**7**	**1.4318**	**8**	**7**

Tab.3.4. *Le temps de convergence des snakes Greedy et Kass sur les différentes images avec différents nombres de points critique et le seuil d'énergie.*

Pour l'image 1 nous constatons que le nombre de points critiques est le même pour les deux méthodes, mais la chose qui nous a interpelé est le temps de calcul qui est élevé pour Kass comparé à Greedy. Cela est dû au développement des *snakes* dans les concavités ce qui est facile pour Greedy. Pour les images (2, 3,5 et 6) le temps de calcule est plus rapide pour l'algorithme de Kass cela est dû à la fonction de ré-échantillonnage, qui garde les points du *snake* distants, donc facilité du déplacement. L'image 4 donne un temps meilleur pour Greedy du fait que le nombre de points critique est inferieur à celui de Kass.

3.5. Conclusion

Lors de ce chapitre nous avons fait une multitude de tests. En premier lieu, nous nous somme intéressé a la segmentation avec les *snakes* (Greedy ou Kass et *al.)* et par la méthode des Chan et Vese. Nous avons remarqué, pour la première, qu'elle est excellente en question de temps de convergence comparé à celle de Chan et Vese. Néanmoins, deux problème on été constaté, entre autre le problème d'initialisation qui doit se faire prêt des objets à détecter, puis celui du changement de topologie au quel nous nous somme intéresser par la suite. Cependant, pour la méthode de Chan et Vese, le bilan avec le quel nous sortant est la bonne segmentation des différentes images utilisées, et cela quel que soit sa taille ou complexité, mais l'un des points faible de cette méthode est le temps de convergence qui est tres lent.

En deuxième lieux, nous nous somme attaqué au problème de détection de plusieurs objets avec les contours actifs paramétriques. Pour pallier au problème de changement de topologie, nous avons eu l'idée d'initialisé plusieurs contours autour des objets du fond, mais le problème qui se pose est la non fusion des contours initiaux. Ensuite, nous avons pensé à l'initialisation à partir d'un seul contour. Ceci nous a contraints de rajouter des étapes supplémentaires pour arriver à notre fin. Les résultats étaient encourageants, mais nous avons pu constater certaines lacunes.

L'objectif de notre travail était de faire une étude approfondie sur les contours actifs en segmentation d'images. Pour cela, une recherche bibliographique dans ce domaine a été effectuée. Nous avons développé séparément, les contours actifs implicites et paramétriques, tout en citant les avantages et les inconvénients. L'utilisation de l'approche *level set* dans le premier modèle des contours actifs, a rendu cette dernière plus simple à implémenter, à changer de topologie et à s'étendre facilement à des images en couleur. Mais l'inconvénient notable de cette méthode est le temps de calcul nécessaire pour sa convergence. En effet, la convergence de l'algorithme nécessite plus d'itérations, car tous les pixels de l'image interviennent, ce qui accroit considérablement la taille du système à résoudre, d'autant que la taille des images augmente au cours de ces dernières années.

Nous avons, ensuite donc, exposé la méthode des contours actifs paramétriques, en détaillant, en particulier, les algorithmes de Kass et Greedy. Ces dernières présentent beaucoup d'avantages. Cependant, lors de leur étude, nous avons constaté certains problèmes. Nous pouvons citer en particulier la sensibilité du modèle à plusieurs paramètres (α, β, γ, etc.). Un mauvais choix de ces paramètres induit, le plus souvent, un comportement erroné du modèle. Il est ainsi, difficile de définir des règles universelles sur le choix de ces paramètres et de prévoir le comportement du modèle pour un ensemble de valeurs donné. Une autre difficulté réside dans son incapacité à détecter plusieurs objets a la fois et, donc, de changer de topologie. Le modèle doit aussi être initialisé proche du contour de l'objet. Au terme de cette étude, l'idée nous vint d'incorporer une novelle méthode qu'on a nommé méthodes des contours actifs multi-objets. Cette méthode a été inspirée du modèle adaptative du *snake* qui est déjà bénéfique question temps de convergence.

Pour cela nous avons procédé par étapes, la première consistait à l'initialisation de plusieurs contours, cela suivant le nombre d'objets à détecter. Le déroulement de cette méthode nous a amené à dresser un bilan suivant les deux algorithmes utilisés (Greedy, Kass et *al.).* Résultats encourageants pour la méthode de Kass et *al.* Pour Greedy, nous avons constaté un problème de redondance des contours finaux, aussi un chevauchement de points qui est dû justement à la taille du voisinage que nous avons initialisé (5×5). Le problème notable qui nous a interpelés durant ce test, est la non fusion des contours initialisés. Ce qui reste une perspective pour les recherches à venir. Ensuite, nous avons pensé à l'initialisation à partir d'un seul contour. Ceci nous a contraints de rajouter des étapes supplémentaires pour arriver à notre fin, entre autre détection des points critiques, puis leur réorganisation en couple. Enfin l'étape de séparation et de reconstruction sous forme de sous contours, cela en

éliminant les zones contour des zones non contour, tout ça sera suivi par une continuité de convergence. Les résultats obtenus étaient encourageants, néanmoins nous avons pu constater certaines lacunes, entre autre l'incapacité à coupler les points critiques suivant les objets à détecter, en la présence de plusieurs de ces points.

Cela nous a amené à dresser certaines perspectives pour une, éventuelle, amélioration de cette méthode, afin de pouvoir la tester sur des images réelles :

- développement d'une fonction distance pour arriver à détecter les bons couples.
- amélioration de la fonction énergie, autre que celle du gradient qui est très simples.
- apporter une autre condition sur les points en comparant justement leurs énergies.

Enfin, nous espérons que ce modeste travail pourra servir à d'autres finalités et sera bénéfique pour les futures promotions.

[1] **Julien Olivier**, "Méthode d'accélération et approches supervisées pour les contours actifs, application à la segmentation d'image 2D, 3D et texturées", thèse de doctorat, Université François Rabelais de Tours, France, 2009.

[2] **Frédéric PRECIOSO**, "Contours actifs paramétriques pour la segmentation d'images et vidéos", thèse de doctorat, Université de NICE - SOPHIA ANTIPOLIS, France, 2004.

[3] **Jean-Jaques Rousselle**, "Les contours actifs, une méthode de segmentation, Application à l'imagerie médicale", thèse de doctorat, Université François Rabelais de Tours, France, 2003.

[4] **HUGUES BELANGER**, "Réseau de KOHONEN pour la détection de contours d'objets dans une image à niveau de gris", rapport de recherche, projet d'application, école de technologie supérieure, université du Québec, Canada, 1998.

[5] **Antoine COUTANT**, "La méthode des contours actifs en traitement des images", Mémoire pour l'examen probatoire en Calcul Scientifique, Conservatoire National des Arts et Métiers Paris, France, 2005.

[6] **Nikolas Petteri Tiilikainen**, "A Comparative Study of Active Contour Snakes", rapport de recherche, Copenhagen University (DIKU), Angleterre, 2007.

[7] **Geoffroy Rivet-Sabourin**, **"**Méthode de segmentation par modèle déformable 2D", rapport de recherche, université Laval, Québec, Canada, 2009.

[8] **CHILALI Ouardia**, "Classification automatique des données utilisant les modèles déformables". Mémoire de magister, université Mouloud Mammeri, Algérie, 2006.

[9] **Hamidatou née Hamadi Fatma Zohra**, "Techniques de segmentation d'images par contours actifs et contribution au mouvement dans des séquences d'images", Mémoire de magister, Ecole Nationale Polytechnique, USD de Blida, Algérie, 2007.

[10] **Jean-Philippe Pons**, "Contributions Méthodologiques et Appliquées à la Méthode des Modèles Déformables", thèse de doctorat, Ecole nationale des Ponts et Chaussées, Marne-la-Vallée, France, 2005.

[11] **Chenyang Xu**, **Dzung L. Pham**, **Jerry L. Prince**, "Image Segmentation Using Deformable Models", téléchargé du site : *www.iacl.ece.jhu.edu/pubs/p119b.pdf.*

[12] **Vincent Barra - Christophe Tilmant**, "TP6 - Segmentation d'images par contours actifs implicites (level set)", téléchargé du site : *www.isima.fr/v**barra**/IMG/pdf/TP6.pdf.*

[13] **Julien MILLE**, "Modèles déformables pour la segmentation et le suivi en imagerie 2D et 3D", thèse de doctorat, Université François Rabelais de Tours, France, 2007.

[14] **Raimana Teina**, "Les modèles déformables et Level Sets (courbes de niveaux)", cours sur les level set, université Pierre et Marie CURIE, France, 2006-2007.

[15] **Wai-Pak Choi**, **Kin-Man Lam*, Wan-Chi Siu**, "An adaptive active contour model for highly irregular boundaries", Pattern Recognition 34 (2001) 323-331, 2001.

[16] **Sébastien LEFÈVRE**, **Nicole VINCENT**, "Contours actifs pour le suivi d'objet en temps-réel: multi-topologies et multi-résolutions", téléchargé du site : *documents.irevues.inist.fr/bitstream/handle/2042/.../A376_55866.pdf.*

[17] **Joanna Isabelle Olszewska**, "Contours Actifs Paramétriques Multi-Cibles pour le Support de la Représentation de Domaines Ontologiques", article du RFIA 2012 (Reconnaissance des Formes et Intelligence Artificielle), Lyon, France, 2012.

[18] **Yagi, Y**., **Kawasaki, Y., Yachida, M**. et **Brady, M.**, "Active contour road model for smart vehicle"**,** proceedings of the International Conference on Pattern Recognition (ICPR00), 2000 .

[19] **Bossart, P.-L., David, D., Dinten, J.-M. et Chassery**, **J.-M.,** "Détection de contours réguliers dans des images bruitées et texturées : une approche par contours actifs multiéchelle", Traitement du Signal, 14(2):209–225, 1996.

[20] **Safont, L.-V. et Marroquin, E.-M.**, "3D reconstruction of third proximal femur (31-) with active contours", In proceedings of the 10th International Conference on Image Analysis and Processing (ICIAP99), Washington, DC, USA, 1999.

[21] **Sethian, J.**, "A review of the theory, algorithms, and applications of level set method for propagating surfaces", In Proceedings of the National Academy of Science,1995.

[22] **Delemas, P**, "Extraction des contours de lèvres d'un visage parlant par contours actifs", thèse de doctorat, Institut Polytechnique de Grenoble, France, 2000.

[23] Contours par modèles élastiques, contours actifs (snakes). Téléchargé du site *www.ceremade.dauphine.fr/~cohen/MVA/*

[24] **V. Caselles, F. Catte, T. Coll**, and **F. Dibos**, "A geometric model for active contours", Numerishe Mathematik, 66:1–31, 1993.

[25] **R. Malladi**, **J. A. Sethian,** and **B. C. Vemuri**, "Shape modeling with front propagation: a level set approach", IEEE Trans, 1995.

[26] **Haker, S., Angenent, S., Tennenbaum**, **A.** et **Kikinis, R.,** "Nondistorting flattening maps and the 3D visualisation of colon CT images", IEEE Transactions on Medical Imaging, 2000.

[27] **G. Sapiro**, "Geometric partial differential equations and image analysis", Cambridge University Press, UK, 2001.

[28] **Roach, M.** et **Mason, J.**, "Salience distance transform. Graphical Models and Image Processing", 1995.

[29] **Chan, T.** et **Vese, L.**, "Active contours without edges", IEEE Transactions on Image Processin, 2001.

[30] **Mumford, D.** et **Shah, J.**, "Optimal approximation by piecewise smooth functions and associated variational problems". Communications on Pure and Applied Mathematics, 1989.

[31] **Solem, J. E., Overgaard, N. C.** et **Heyden, A.**, "Initialization techniques for segmentation with the Chan-Vese model", In proceedings of the 18th International Conference on Pattern Recognition, Washington, DC, USA. IEEE Computer Society, 2006.

[32] **Sethian, J. A.** "A fast marching level set method for monotonically advancing fronts", In Proceedings of the National Academy of Science, 1996.

[33] **Adalsteinsson, D.** et **Sethian, J. A.**, "A fast level set method for propagating interfaces", Journal of Computational Physics,1995.

[34] **Paragios, N**. et **Deriche, R.**, "Geodesic active contours and level sets for the detection and tracking of moving objects", International Journal of Computer Vision, 2000.

[35] **Kass, M., Witkin**, A. et **Terzopoulos, D.**, "Snakes : active contour models", International Journal of Computer Vision, 1988.

[36] **Tikhonov, A.** et **Arsénine, V.**, "Méthodes de résolution de problèmes mal posés". Moscou, URSS, 1976.

[37] **Cohen, L. D. Cohen, I**., "Finite element methods for active contours models and balloons for 2D and 3D Image", Third International Conference on Computer Vision, Osaka, Japan, 1990.

[38] **Williams, D. J**. et **Shah, M.**, "A fast algorithm for active contours and curvature estimation. Computer Vision, Graphics, and Image Processing: Image Understanding", 1992.

[39] **Lam, C. L.** et **Yuen, S. Y.**, "An unbiased active contour algorithm for object tracking", Pattern Recognition Letters, 1998.

[40] **Amini, A. A., Weymouth, T. E.** et **Rain, R. J.**, "Using dynamic programming for solving variational problems in vision", IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990.

[41] **Shamir, A.**, "Segmentation and shape extraction of 3d boundary meshes", In State-of-the-Art Report, Proceedings of Eurographics, 2006.

[42] **Holland, J.H.**, "Adaptative in natural and artificial systems", University of Michigan Press, 1975.

[43] **Muhammad Hameed Siddiqi**; **Sungyoung Lee; Young-Koo Lee**, "Object Segmentation by Comparison of Active Contour Snake and Level Set in Biomedical Applications", téléchargé du site: *doi.ieeecomputersociety.org/10.../BIBM.2011.61*

[44] **Xu, C. et Prince, J.**, "Snakes, shapes, and gradient vector flow", IEEE Transactions on Image Processing, 1998.

[45] **Paragios, N., Mellina-Gottardo, O**. et **Ramesh, V.**, "Gradient vector flow fast geometric active contours", IEEE Transactions on Pattern Analysis and Machine Intelligence 2004.

[46] **J. I. Olszewska and T. L. McCluskey**, "Ontologycoupled active contours for dynamic video scene understanding", in INES'11, June 2011, pp. 369-374, 2011.

[47] **J. I. Olszewska**, "Spatio-Temporal Visual Ontology", in VL'11, 2011.

[48] **D. A. Randell, Z. Cui,** and **A. G. Cohn**, "A spatial logic based on regions and connection", in Proceedings of the Int. Conf. on Know. Repr. and Reas.,1992.

[49] **Ge Xingfei, Tian Jie.** "An automatic active contour model for multiple objects", *AILAB*, Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China.

[50] **Nikos Paragios**, "Level Set Methods in Medical Image Analysis: Segmentation", CERTIS Ecole Nationale des Ponts et Chaussées Paris, France.

[51] **Adalsteinsson, D.** et **Sethian, J. A.**, "A fast level set method for propagating interfaces", Journal of Computational Physics,1995.

[52] **Shi, Y. et Karl, W.**, "A fast level set method without solving PDEs. In proceedings of the", in proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.

[53] **Ariane HERBULOT,** "Mesures statistiques non-paramétriques pour la segmentation d'images et de vidéos et minimisation par contours actifs". Thèse de doctorat, Université de Nice - Sophia Antipolis, 2007.

[54] **Dr. Xavier BRESSON**, **Prof. Jean-Philippe THIRAN,** " Reconnaissance des formes, Cours 3: Contours actifs", Swiss Federal Institute of Technology, Lausanne.

[55] **Michel Roux**, "Segmentation des images, Détection de contours", cours sur les contours actifs, Telecom Paris Tech.

[56] **Olivier Bernard**, "Segmentation d'images par la méthode des Level-set", centre national de la recherche scientifique, Inserm Lyon, 2012.

[57] **Khalifa Nawres, Malek Amel, Hamrouni Kamel,** "Segmentation d'images par contours actifs : Application à la détection du ventricule gauche dans les images de scintigraphie cardiaque", Ecole Nationale d'Ingénieurs de Tunis, 2005.