MÉMOIRE

en vue de l'obtention du

DEA "Algorithmique, Robotique, Automatique, Vision, Image et Signal"

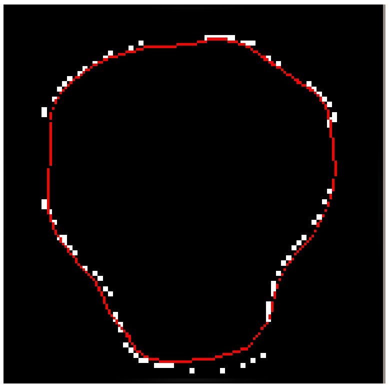
présenté par

Guillaume PICINBONO

sous la direction de

Hervé DELINGETTE

MODÈLE GÉNÉRIQUE DE CONTOUR DÉFORMABLE IMPLICITE POUR LA SEGMENTATION ET LA SIMULATION



Étude réalisée à l'Institut National de Recherche en Informatique
et en Automatique de Sophia Antipolis
-Juin 1997-

Table des matières

Introduction

Ce stage de D.E.A. a été réalisé au sein du projet Epidaure, à l'Institut National de Recherche en Informatique et Automatique (I.N.R.I.A.) de Sophia-Antipolis. Ce projet est spécialisé dans l'imagerie médicale, et ses activités se divisent en deux grandes catégories:

- le traitement des images médicales (débruitage, filtrage, segmentation, reconstruction, visualisation, recalage, ...) qui a pour finalité une aide au diagnostique.
- la simulation de chirurgie (cranio-faciale ou hépatique).

Dans toutes ces applications, on retrouve souvent l'utilisation de modèles déformables (*Snakes* ou modèle tétraèdrique à élasticité linéaire) qui sont de type paramétrique. Ces modèles, très performants au niveau du temps de calcul, ont cependant le défaut de ne pas pouvoir changer de topologie, ce qui pourrait s'avérer intéréssant pour la segmentation (pour segmenter plusieurs composantes connexes d'une même structure anatomique), et surtout pour la simulation chirurgicale, lors des étapes d'incision.
C'est pourquoi je me suis intéressé à l'élaboration et au développement d'un modèle déformable implicite, en essayant de conserver un champ d'application le plus large possible.

Le premier chapitre de ce rapport est consacré à une étude bibliographique des principaux modèles de surface implicites développés jusqu'à ce jour. Une fois cette étape terminée, un nouveau modèle a été imaginé, en accord avec le cahier des charges fixé à l'avance. Le chapitre 2 présente ce modèle, ainsi que tous les travaux effectués sur la courbure pour trouver la force interne adéquate. Dans le chapitre suivant, plusieurs forces externes sont proposées pour s'adapter à plusieurs types d'applications. Enfin, le chapitre 4 présente l'interface utilisateur développée au cours du stage.

Chapitre 1

Bibliographie sur les surfaces implicites

1.1 Introduction

Il existe deux manières de représenter des surfaces dans un espace à trois dimensions (ou des courbes en deux dimensions).

Premièrement, il y a la représentation paramétrique. Une surface dans l'espace est alors définie par une fonction $\mathcal{F}$ d'un ouvert $\mathcal{U}$ de $\mathbb{R}^2$ dans $\mathbb{R}^3$ de la forme :

$$\mathcal{F} : (\mathrm{u}, \mathrm{v}) \in \mathcal{U} \subset \mathbb{R}^2 \quad \longmapsto \quad [\mathcal{X}(\mathrm{u}, \mathrm{v}), \mathcal{Y}(\mathrm{u}, \mathrm{v}), \mathcal{Z}(\mathrm{u}, \mathrm{v})] \in \mathbb{R}^3 \tag{1.1}$$

Cette représentation est très utilisée en vision par ordinateur car elle permet la détermination de points sur la surface par un calcul direct (une simple substitution). Par contre la formulation de surfaces relativement complexes demande une formule analytique assez lourde. D'autre part, une surface paramétrique est figée au niveau de la topologie, ce qui peut poser de gros problèmes dans certaines applications telles que la simulation d'objets fortement déformables.

Deuxièmement, une surface peut être définie de manière implicite. Dans ce cas elle est considérée comme la surface isopotentielle (iso-surface) d'un champ scalaire défini sur $\mathbb{R}^3$ tout entier. Si on appelle $\mathcal{F}$ la fonction de champ :

$$\mathcal{F} : (\mathrm{x}, \mathrm{y}, \mathrm{z}) \in \mathbb{R}^3 \longmapsto \mathcal{F}(\mathrm{x}, \mathrm{y}, \mathrm{z}) \in \mathbb{R} \tag{1.2}$$

Alors la surface $\mathcal{S}$ est définie par :

$$\mathcal{S} = \{\mathrm{M}(\mathrm{x}, \mathrm{y}, \mathrm{z}) \in \mathbb{R}^3 / \mathcal{F}(\mathrm{M}) = 0\} \tag{1.3}$$

Une surface définie de manière implicite engendre les notions d'intérieur-extérieur (dans le cas des surfaces fermées), et de droite-gauche (pour les surfaces non-fermées), basées sur le signe de $\mathcal{F}$ dans ces régions. Il en découle deux propriétés essentielles: les surfaces implicites sont de topologie absolument quelconque (pour une fonction $\mathcal{F}$ donnée, la topologie peut changer suivant l'iso-valeur choisie), et ne s'auto-intersectent pas (la figure 1.1 en donne une justification en deux dimensions).

De plus, il n'est pas nécessaire de posséder une formulation analytique de la fonction de champ. En effet, la détermination de points sur la surface se fait par détection du passage

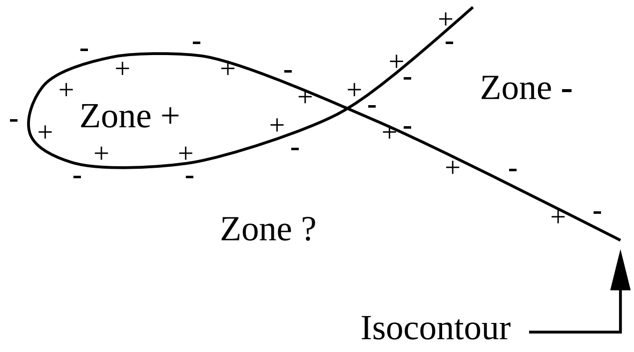


Fig. 1.1 – *Propriétés des surfaces implicites*

par zéro de $\mathcal{F}$ (ce qui constitue d'ailleurs une des difficultés majeures de cette représentation) grâce à des algorithmes qui fonctionnent tout aussi efficacement sur une version échantillonnée du champ scalaire. Mises à part les différences de définitions entre les deux modèles cités, ils impliquent des approches différentes lors des applications, en particulier dans le domaine qui nous intéresse pour ce travail: la simulation et l'utilisation d'objets déformables. En effet, alors que l'approche paramétrique va consister à confectionner un maillage de la surface, et à déformer ce maillage lors de la simulation, l'approche implicite permet d'appliquer une déformation sur l'ensemble du champ scalaire (en y appliquant par exemple une équation différentielle), puis éventuellement construire un maillage de la surface pour la visualiser.

Comme nous allons le voir, il existe de nombreuses manières de modéliser des surfaces implicites, suivant les domaines d'applications convoités par les auteurs. Les paragraphes suivants proposent une étude bibliographique menée sur une trentaine d'articles et ouvrages, visant à présenter les principales méthodes de modélisation de surfaces implicites, ainsi que les principes de déformation et de visualisation propre à chaque modèle.
La visualisation des surfaces implicites étant un problème épineux, de nombreuses méthodes ont été proposées. Certaines sont absolument indépendantes du modèle utilisé. Je les regrouperai en fin de chapitre. Par contre je vais consacrer le paragraphe "Principe de visualisation" de chaque modèle à la présentation de ceux propres à chaque modèle.

1.2 Surfaces implicites générées par un squelette

Pour obtenir la représentation implicite d'une surface, il faut créer un champ de potentiel scalaire dont on pourra extraire l'iso-surface correspondant à une iso-valeur donnée. Il existe plusieurs moyens d'y parvenir. Un des modèles les plus utilisés est celui que l'on pourrait nommer **Squelette+Potentiel**.

1.2.1 Principe de modélisation

Dans ce modèle, le champ scalaire $\mathcal{F}$ est engendré de la manière suivante : on définit un squelette $\mathcal{Q}$ à partir de plusieurs composantes élémentaires qui peuvent être des points, droites, courbes, figures géométriques, surfaces, volumes, etc... ([TG95],[Gas93],[BW90]), et on associe à chaque composante un champ qui est en général une fonction strictement décroissante de la distance du point à la composante du squelette. On regroupe souvent tous les modèles implicites basés sur la notion de distance sous l'appellation *Distance Surfaces*. La surface $\mathcal{S}$ est alors définie par:

$$\mathcal{S} = \{\mathrm{M(x,y,z)} \in \mathrm{I\!R}^3 / \mathcal{F}(\mathrm{M}) = \sum_{q_i} f_i(r_i) = c\} \tag{1.4}$$

Où r_i est la distance du point M à la composante q_i du squelette, et f_i le potentiel associé à cette composante.

C'est au niveau des fonctions f_i que de petites différences apparaissent entre les auteurs. Mais l'idée générale est souvent la même. Cette idée est exposée très clairement dans [TG95], où Nicolas Tsingos et Marie-Paule Gascuel s'intéressent surtout à la modélisation et la visualisation d'objets définis par des surfaces implicites. Ils définissent alors la fonction de potentiel à l'aide des trois paramètres mis en évidence sur la figure 1.2 ci-dessous:

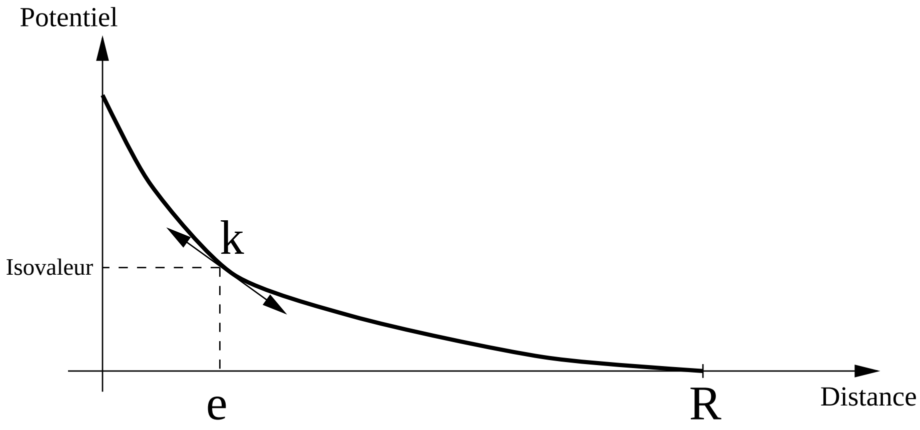


FIG. 1.2 – *Fonction potentiel et ses paramètres de contrôle*

- **R** est le *rayon d'influence*, au delà duquel le potentiel s'annule, ce qui réduit le temps de calcul.
- **e** est *l'épaisseur* de la surface autour du squelette lorsque ce dernier est isolé.
- **k** est la *raideur*, définie comme la pente de la fonction potentielle en **e**. Cette raideur contrôle le mélange des potentiels.

Le logiciel intéractif qu'ils présentent leur permet alors de construire un objet en assemblant des morceaux de squelette et en leur attribuant des potentiels adéquats.

Mais il existe une alternative aux *Distance Surfaces* particulièrement intéréssante au niveau des points de jonction entre les éléments du squelette (points justement où les *Distance Surfaces* présentent quelques défauts). Ce sont les *Convolution surfaces* introduites par Jules Bloomenthal et Ken Shoemake dans [BS91]. Le principe est d'opérer une convolution du squelette avec une gaussienne 3D. La fonction $\mathcal{F}$ est alors obtenue par:

$$\mathcal{F}(p) = \sum_{q_i} f_i(p) \tag{1.5}$$

Avec

$$f_i(p) = \int_{q_i} exp\left(\frac{-\|q_i - p\|^2}{2}\right) dq_i \tag{1.6}$$

Les propriétés de distributivité du produit de convolution garantissent alors que deux composantes disjointes du squelette juxtaposées engendreront la même surface que si elles ne formaient qu'une seule composante connexe. On contourne ainsi le problème de surépaisseur rencontré lors de la fusion de deux *Distance Surfaces* (les contributions des f_i s'ajoutent). La fléxibilité du modèle peut être augmentée en ajoutant une fonction de pondération lors de la convolution, ce qui permet de faire varier l'*épaisseur* de la suface le long du squelette, et ceci indépendamment dans les trois directions de l'espace.

1.2.2 Principe de déformation

Si le modèle **Squelette+Potentiel** représente un outil puissant de modélisation (assez intuitif d'ailleurs puisqu'on y retrouve l'idée d'un squelette rigide avec de la "matière molle" autour), il présente en plus d'intéréssantes possibilités pour simuler des déformations. On pense en effet tout de suite à l'ajout ou à la supression de composantes du squelette, à la modification de l'iso-valeur, et enfin, et surtout, à la modification des trois paramètres de la fonction potentielle (*épaisseur, raideur, rayon d'influence*). Mais l'intérêt de cette méthode ne s'arrête pas là. En effet les trois paramètres cités plus haut permettent d'attribuer des propriétés physiques à l'objet modélisé. Ainsi Marie-paule Gascuel dans [Gas93] utilise la raideur **k** pour simuler le comportement élastique (linéaire ou non-linéaire suivant la forme de f_i) de deux corps en mouvement qui entrent en collision. Ceci lui permet non seulement de calculer les déformations relatives des deux objets, mais aussi d'en déduire des forces de pression et le comportement dynamique de l'ensemble du système. Le principe de fonctionnement est le suivant:

- Deux objets sont en mouvement.
- On détecte la collision en déterminant la zone d'interpénétration des deux corps (utilisation des *Bounding Box* ([KB89])).
- On empêche l'interpénétration en déformant les objets en fonction de leurs propriétés physiques (*raideur*). On obtient ainsi une surface de contact.
- On déduit de cette surface de contact les forces de pression et éventuellement de torsion.
- On intègre les équations pour connaître les nouvelles composantes des mouvements.

C'est l'étape de déformation qui nous intéresse le plus dans ce paragraphe. Une fois la collision détectée, on ajoute à chaque f_i un terme de correction de champ g_{ji} calculé à partir du potentiel f_j du deuxième objet. C'est cette idée d'ajouter une composante de champ (de manière globale ou locale) qui présente le plus d'intérêt pour la simulation de mouvements de déformations.

L'équivalent dans le cas des *Convolution Surfaces* est de reconvoluer certaines composantes du squelette avec de nouvelles gaussiennes (cf. propriétés du produit de convolution).

Ceci permettra, en plus, des déformations plus puissantes comme l'étirement dans une direction privilégiée, ou la torsion.

1.2.3 Principe de visualisation

Dans [TG95], Nicolas Tsingos propose une version intéractive de la notion de **graine** introduite dans [BW90]. Le but est de calculer des points sur la surface. La méthode comporte deux étapes: l'initialisation des graines et leur migration.
Initialisation: (Figure 1.3) on fixe une boîte d'épaisseur **e** autour de chaque composante du squelette, un échantillonnage de la boîte nous donne les graines, et on projette ces graines orthogonalement sur le squelette pour en déduire les directions de migration. Au niveau des jonctions entre les morceaux du squelette, certaines graines sont désactivées lorsqu'elles ne sont pas dans une zone où le potentiel de leur squelette domine. *Migration:* (Figure 1.4)

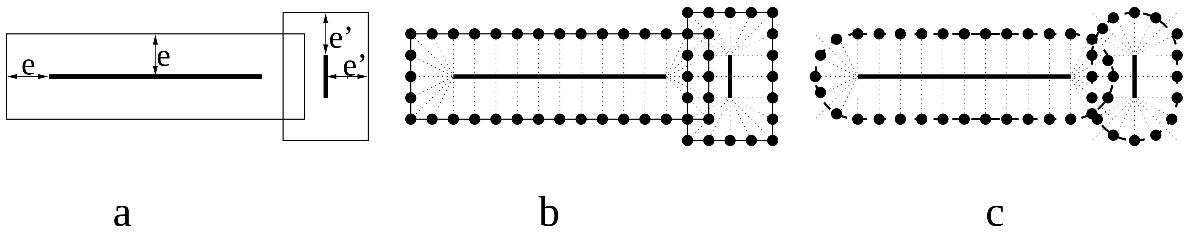


FIG. 1.3 – *Initialisation des graines*

les graines valides se déplacent selon leur direction de migration et trouvent leur place au point d'intersection avec la surface par dichotomie. Une fois sur la surface, les graines sont

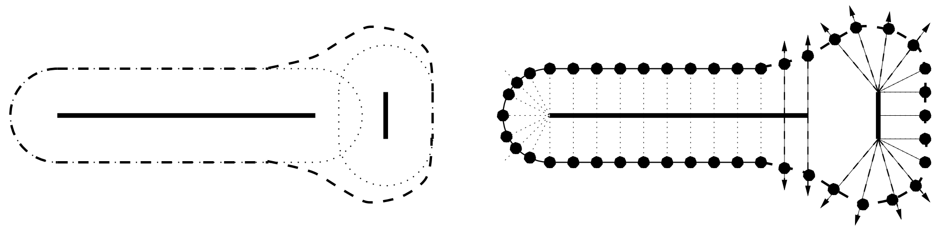


FIG. 1.4 – *Migration des graines*

utilisées pour une visualisation en écailles (petits triangles centrés sur la graine), ou une polygonisation.

1.3 Surfaces implicites générées par des particules

Si on veut un modèle qui se rapproche encore plus de la réalité physique des matériaux, on peut penser à modéliser un corps à partir d'atomes liés par des forces. Ce concept est applicable aux surfaces implicites, grâce aux **Systèmes de Particules** (*Particle Systems*) introduits tout d'abord par W.T.Reeves en 1983 (mais sans intéraction entre les particules), puis plus récemment par G. Miller et A. Pearce en 1989 et par D. Tonnesen en 1991.

1.3.1 Principe de modélisation

L'idée de départ est d'utiliser un ensemble de particules reliées par des forces d'*attraction-répulsion* pour engendrer un champ scalaire, et donc une surface implicite dont on pourra simuler les déformations en appliquant les lois de la mécanique et de la dynamique. Suivant l'application, on pourra choisir un modèle plus ou moins compliqué, c'est-à-dire qui prend en compte un nombre plus ou moins grand de propriétés de la matière.

Dans ce modèle, tout le travail réside dans la formulation des forces d'intéraction entre les particules. Le minimum est de définir une force d'attraction-répulsion ou force interne f_{ij} exercée par la particule i sur la particule j ([MD95],[DG96],[Mur91]). Cela permet de modéliser la répulsion à petite échelle, qui empêche les particules de fusionner (et la matière de s'effondrer), et l'attraction à grande échelle qui assure la cohésion du matériau. Mais en général les auteurs rajoutent d'autres forces comme les forces de pression ou de friction.

Dans [DG96] M. Desbrun et Marie-Paule Gascuel introduisent un formalisme un peu particulier dérivé des **Smoothed Particle Hydrodynamics** (SPH) introduites par J.J. Monaghan dans [Mon92]. Si on considère un ensemble de particules j, ayant une position r_j, une masse m_j, une vitesse v_j, une densité de masse ρ_j et un champ physique f_j, alors la valeur et la dérivée de ce champ physique en tout point de l'espace sont déterminées de la manière suivante:

$$f(r) = \sum_j m_j \frac{f_j}{\rho_j} W_h(r - r_j) \tag{1.7}$$

$$\nabla f(r) = \sum_j m_j \frac{f_j}{\rho_j} \nabla W_h(r - r_j) \tag{1.8}$$

La fonction W_h assure une dispersion homogène des particules en lissant leur fonction de répartition. Le champ physique *f(r)* prend alors la forme d'un produit de convolution, avec toutes les propriétés intéressantes que cela comporte, comme on peut déjà le noter dans la formule (8). Les forces (de pression et de viscosité en particulier) entre les particules sont ensuite calculées à partir du gradient du champ considéré. On retrouve ici un peu l'esprit des *Convolution surfaces* évoquées dans le modèle **Squelette+Potentiel** (cf. paragraphe précédent). C'est donc tout naturellement sur le noyau de convolution W_h que le plus gros du travail est fait afin d'obtenir un comportement physique proche de la réalité et une efficacité de calcul raisonnable.

En plus de la possibilité de construire un objet implicite avec des particules, il peut être intéressant de l'ajuster sur des données (images volumiques par exemple). Dans [Mur91] Shigeru Muraki propose un algorithme récursif permettant d'ajuster un système de particules (avec un nombre minimal de particules) sur un nuage de points. La méthode consiste tout d'abord à initialiser avec une seule particule et à optimiser sa position et son champ par un algorithme de minimisation d'une fonction d'énergie. Ensuite cette particule est scindée en deux et on recommence le même schéma sur les deux particules. On peut donc ainsi obtenir la représentation implicite d'un objet avec la précision que l'on désire (la précision est controlée par le seuil d'arrêt de l'algorithme de minimisation).

Une dernière variante des systèmes de particules est exposée dans la deuxième partie de l'article [LHdFVen] où sont utilisés des systèmes **Masses-Ressorts** pour simuler les forces internes entres les particules.

1.3.2 Principe de déformation

Dans [MD95] et [DG96] Mathieu Desbrun et Marie-Paule Gascuel traitent de manière assez complète les déformations des surfaces implicites modélisées par des systèmes de particules. Ce principe de représentation étant très proche de la réalité physique, on peut appliquer aux particules les lois de la dynamique. Ainsi, lorsqu'une force est appliquée, on en déduit l'accélération des particules ($F = m\gamma$), puis par double intégration les nouvelles positions des particules, ainsi que les forces de réaction engendrées par leurs déplacements (compression ou dilatation du matériau).

Mais il y a deux problèmes particuliers à gérer.

Le premier est celui de la déformation à volume constant, souvent souhaitable pour les matériaux incompressibles. Le modèle des particules y apporte une solution intéressante. Pour chaque particule on va définir un **Territoire** à l'intérieur duquel le champ généré par la particule est dominant. Le volume de ce territoire est calculé en utilisant des graines (cf paragraphe sur le modèle squelette+potentiel) qui migrent jusque sur les frontières des territoires (voir figure 1.5 ci-dessous). Ainsi, lors de la déformation, on pourra ajuster le

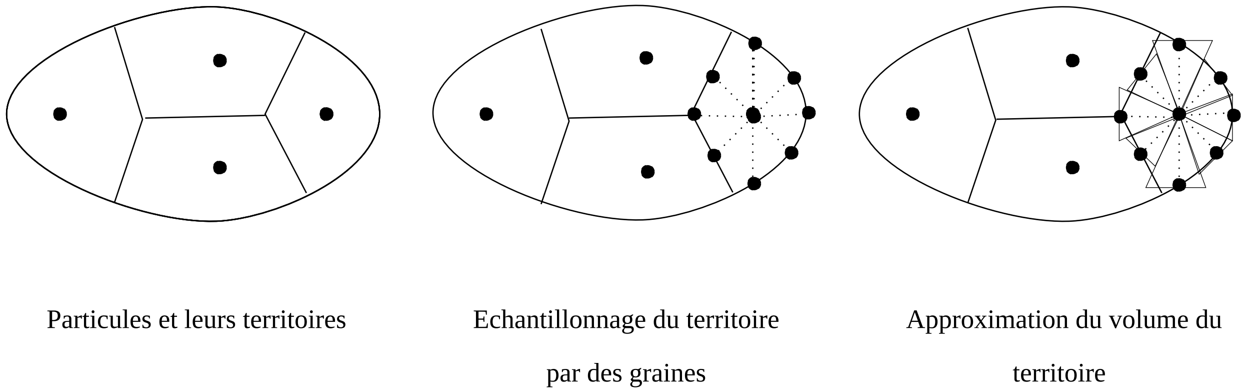


FIG. 1.5 – *Territoires des particules et les graines utilisées pour le calcul du volume*

champ de chaque graine pour que le volume de chaque territoire reste constant.

Le second problème est celui de la fusion de deux corps qui rentrent en collision. On ne veut pas que les deux corps fusionnent dès qu'ils sont mis à proximité l'un de l'autre, à cause du mélange des champs comme on peut le voir sur la figure suivante. Ce problème est contourné par la gestion d'un graphe de connection entre les particules. Si deux particules dont les territoires entrent en collision sont déconnectées, on empêche les deux champs de se mélanger en déformant chacun des deux corps. On peut aussi définir un seuil sur la force de contact entre les deux corps, au delà duquel les particules mises en jeu sont connectées, permettant ainsi aux corps de fusionner.

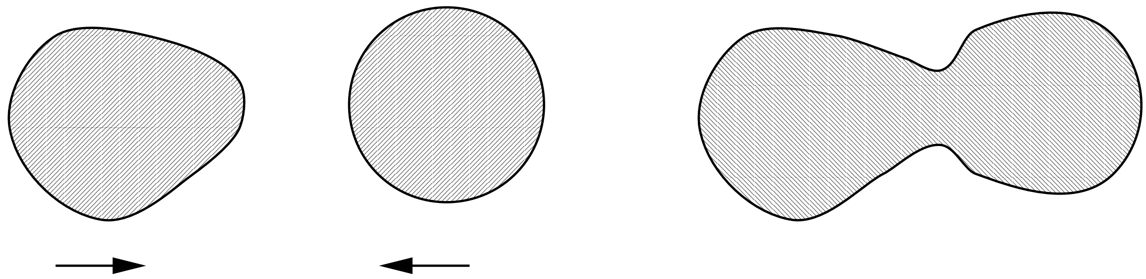


FIG. 1.6 – *Le problème de la fusion à distance*

1.3.3 Principe de visualisation

La notion de graine introduite au paragraphe 1.2.3 est applicable ici (en considérant les particules comme des squelettes réduits à des points).
D'autre part les algorithmes utilisés dans [LHdFVen] pour générer des systèmes masssesressorts engendrent une polygonisation de la surface implicite qui permet une visualisation efficace sur des machines graphiques.

1.4 Level set methods

Toute la théorie des **Level Set Methods** est développée dans le livre de J. A. Sethian [Set96], et une application inspirée des idées de R. Malladi est présentée dans l'article [RMV95]. L'idée fondamentale est la suivante (on se place en dimension deux pour plus de clarté). On dispose d'une courbe initiale $\gamma(0)$ que l'on veut déformer au cours du temps en une famille de courbes paramétrées $\gamma(t)$. Alors on considère $\gamma(0)$ comme la ligne de niveau zéro d'un certain champ scalaire $\Phi(X, o)$, et c'est cette fonction que l'on va déformer en une famille de fonctions $\Phi(X, t)$. Alors $\gamma(t)$ est la ligne de niveau zéro de $\Phi(X, t)$. Donc,

$$\gamma(t) = \{X/\Phi(X, t) = 0\} \tag{1.9}$$

Dans la plupart des applications, la fonction $\Phi(x, o)$ est initialisée comme la carte des distances de chaque point x à la courbe $\gamma(0)$.

$$\Phi(x,t) = \begin{cases} +d(X,\gamma(0)) & \text{si x est à l'extérieur de } \gamma(0) \\ -d(X,\gamma(0)) & \text{si x est à l'intérieur de } \gamma(0) \end{cases}$$

En déformant $\Phi(x, t)$, on déforme donc tous les isocontours. On obtient l'équation de déformation en dérivant $\Phi(x, t) = C$ (équation de la courbe de niveau C), ce qui donne:

$$\Phi_t + F|\nabla\Phi| = 0 \tag{1.10}$$

Où $F = X'(t).\vec{n}$ avec $\vec{n} = \frac{\nabla\Phi}{|\nabla\Phi|}$.

F est donc la composante normale du vecteur vitesse de chaque point d'un isocontour. Suivant les applications F peut être constant, ou proportionnel à la courbure, ou aux différentes

dérivées de la courbure. F peut enfin être une combinaison linéaire des différents termes cités précédemment.
Dans son livre, J. A. Sethian propose une étude très approffondie de la fonction F et des schémas numériques à utiliser pour éviter la formation de coins, de points de rebroussement et d'oscillations lors de l'évolution des courbes de niveau.

1.5 Surfaces implicites générées par une fonction Intérieur-Extérieur

Ce que je nomme fonction intérieur-extérieur est en fait une fonction qui ne peut prendre que deux valeurs, une à l'intérieur de l'objet et l'autre à l'extérieur. Si on travaille en niveaux de gris, ce sera par exemple 0 et 255. Pour éviter certains problèmes numériques, on va lisser cette fonction (par exemple en la convoluant avec une gaussienne) pour la rendre continue. Dans ce cas l'objet sera délimité par la surface de niveau 128.

Ce genre de représentation est utilisé par T. A. Galyean et J. F. Hughes dans [GH91] (avec 0.0 et 1.0 au lieu de 0 et 255), pour son logiciel de sculpture intéractive en trois dimensions. La sculpture se fait à l'aide de plusieurs instruments prédéterminés, par retrait de matière (c'est à dire par modification de la valeur de certains voxels). Pour se rapprocher encore plus de la réalité, il y a aussi la possibilité de rajouter de la matière.

1.6 Visualisation des surfaces implicites

Comme annoncé dans l'introduction, ce paragraphe est réservé à la présentation des méthodes de visualisation qui sont applicables à toutes les surfaces définies de manière implicite, quel que soit le modèle utilisé. Elles sont au nombre de trois, et comportent quelques points communs.

1.6.1 Le lancé de rayons ou *Ray tracing*

L'idée de base du lancé de rayons est de déterminer l'intersection d'une surface implicite avec une droite. Cette droite représente la direction de l'espace dans laquelle on regarde. Ainsi, en lançant des rayons dans toutes les directions, on obtient un ensemble de points sur la surface, qui permet de la visualiser.

Pour présenter cette méthode un peu plus précisement, je m'appuirai sur l'article de D. Kalra et A. H. Barr [KB89]. La plupart des méthodes de lancé de rayons imposent des hypothèses assez contraignantes sur les propriétés de la fonction de champ $\mathcal{F}$. L'intérêt du travail de Kalra et Barr est de n'imposer que l'existence (et la connaissance) des constantes de **Lipschitz** de $\mathcal{F}(\mathrm{X})$ et de $\mathcal{G}(\mathrm{t}) = \frac{\mathrm{d}\mathcal{F}(\alpha \mathrm{t}+\beta)}{\mathrm{dt}}$, où $\alpha t + \beta$ est la droite repèsentant le rayon.
On rappelle qu'une fonction f est dite $\mathcal{L}$-Lipschitzienne dans une région $\mathcal{R}$ si:

$$\exists \mathcal{L} \in \mathrm{I\!R} \mid \forall \mathrm{x}_1 \in \mathcal{R}, \mathrm{x}_2 \in \mathcal{R}, \|\mathrm{f}(\mathrm{x}_1) - \mathrm{f}(\mathrm{x}_2)\| < \mathcal{L}\|\mathrm{x}_1 - \mathrm{x}_2\| \tag{1.11}$$

A partir du moment où l'on dispose de $\mathcal{L}$ (constante de Lipschitz de $\mathcal{F}(\mathrm{X})$) et de $\mathcal{M}$ (constante de Lipschitz de $\mathcal{G}(\mathrm{t})$), on peut appliquer l'algorithme.

La première partie de l'algorithme consiste à creer les *Bounding Boxes* c'est à dire un ensemble de boîtes qui contiennent la surface. Pour cela, on prend au départ une boîte qui contient la surface toute entière (toute l'image par exemple), et on la découpe en sous-boîtes. Pour chaque sous-boîte:

- Si les sommets de la boîte sont de signes différents, alors la surface coupe la boîte et on la garde.
- Si les sommets sont du même signe, et que $\mathcal{F}(\mathrm{X}_0) > \mathcal{L}\mathrm{d}$ (où X_0 est le centre de la boîte et d le rayon de la sphère circonscrite à la boîte), alors la surface ne coupe pas la boîte et on la jette.
- Si les sommets de la boîte sont de signes différents et que $\mathcal{F}(\mathrm{X}_0) < \mathcal{L}\mathrm{d}$, il faut redécouper la boîte pour lever l'ambiguïté.

Cette première étape permet d'augmenter l'efficacité du lancé de rayon proprement dit.

La deuxième partie se déroule selon le principe suivant: un rayon donné va intersecter une ou plusieurs boîtes déterminées à la première étape. On prend la plus proche. Le rayon la coupe en t_1 et t_2.

- Si $t_1.t_2 < 0$ et $|\mathcal{G}(\mathrm{t_m})| > \mathcal{M}\mathrm{d}$ (t_m est le milieu de $[t_1 t_2]$), alors il y a un seul point d'intersection, déterminé par un algorithme de Newton.
- Si $t_1.t_2 > 0$ et $|\mathcal{G}(\mathrm{t_m})| > \mathcal{M}\mathrm{d}$, il n'y a pas de point d'intersection et on passe à la boîte suivante.
- Si $|\mathcal{G}(\mathrm{t_m})| < \mathcal{M}\mathrm{d}$, on divise l'intervalle $[t_1, t_2]$ en deux et on recommence avec chaque sous-intervalle.

Il faut remarquer que l'algorithme est d'autant plus efficace que les constantes de Lipschitz sont précises et locales. Il ne faut donc pas hésiter à les calculer pour chaque boîte.

1.6.2 *Marching Cubes*

Le *Marching cubes algorithm* est présenté par W. E. Lorensen et H. E. Cline dans [LC87]. On peut le résumer de la manière suivante: les huit sommets de chaque voxel se voient attribuer un signe suivant qu'ils se trouvent à l'intérieur ou à l'extérieur de l'objet. Ceci fournit 256 possibilités que l'on peut réduire à 15 cas en considérant les symétries. Pour chaque cas, une triangulation générale est proposée et adaptée par interpolation linéaire. Une des caractéristiques intéressantes de cette algorithme est de pouvoir suivre la surface d'un voxel vers certains de ses voisins. De nombreuses méthodes existent pour trouver un premier voxel sur la surface: recherche systématique, tirage au sort, ...

Mais le gros problème de ce type d'algorithme est qu'il faut orienter les surfaces et surtout propager l'orientation d'un voxel à l'autre pour assurer de bonnes propriétés topologiques (ces problèmes ont été mis en évidence pour la première fois dans [D̈88]). Une grosse partie du travail de J-P. Thirion et A. Gourdon ([TG93]) a été de developper une théorie leur permettant de démontrer que les isosurfaces (ou les lignes, intersections de deux isosurfaces) générées par leur algorithme (*Marching lines*) possèdent toutes les propriétés topologiques convoitées.

1.6.3 Polygonisation

Le principe de la polygonisation est d'approximer une surface par un ensemble de polygones dont les sommets sont des points de la surface. Ce principe se rapproche de celui présenté dans le chapitre précédent. Mais la méthode peut être raffinée, comme c'est le cas dans les articles de L. Velho ([Velen]) et de J. Bloomenthal ([Blo94]). Une fois l'espace décomposé en voxels, on selectionne ceux qui intersectent la surface. Alors chacun de ces voxels est à son tour divisé en six tétraèdres (ou *3-simplex*) de la manière suivante:

- Choisir une des diagonales du cube et la tracer.
- Tracer la projection orthogonale de cette diagonale sur chaque face.
- Chaque tétraèdre est alors construit comme le montre la figure suivante:

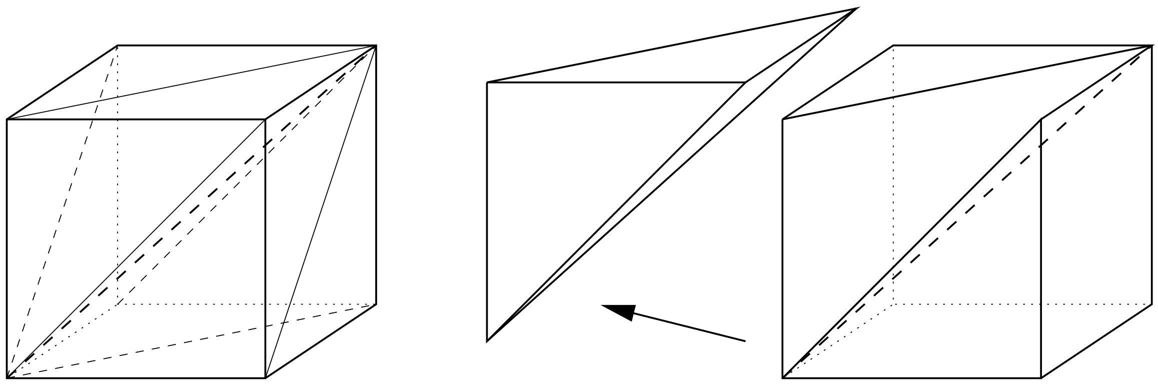


FIG. 1.7 – *Décomposition d'un cube en tétraèdres*

L'intersection d'un tétraèdre et d'un plan peut être soit un triangle, soit un quadrilatère. Dans ce dernier cas, il est coupé en deux suivant la plus petite des diagonales. On obtient ainsi une triangulation de la surface. Dans [Velen], L. Velho se sert de cette triangulation comme point de départ pour un algorithme de raffinement adaptatif. En effet, une polygonisation est une bonne approximation d'une surface lorsque la surface est assez plate. Par contre, dans des zones à forte courbure, la précision s'avère souvent insuffisante. Ainsi, l'algorithme de raffinement calcule la variation de la courbure le long des arêtes de chaque triangle du maillage, et lorsque cette variation dépasse un certain seuil au moins le long d'une des trois arêtes, un point est rajouté au milieu de chaque arête du triangle . Si ces points ne sont pas sur la surface, ils y sont projetés orthogonalement. Le maillage est ensuite remis à jours en reliant les points milieux entre eux. Un triangle est ainsi décomposé en quatre (voir Figure 1.8). L'algorithme est ainsi appliqué recursivement sur le nouveau maillage.

1.7 Conclusion

Les domaines d'application des surfaces implicites sont vastes. Elles sont aujourd'hui utilisées en segmentation, en reconnaissance de forme, en animation, ainsi que pour la mo-

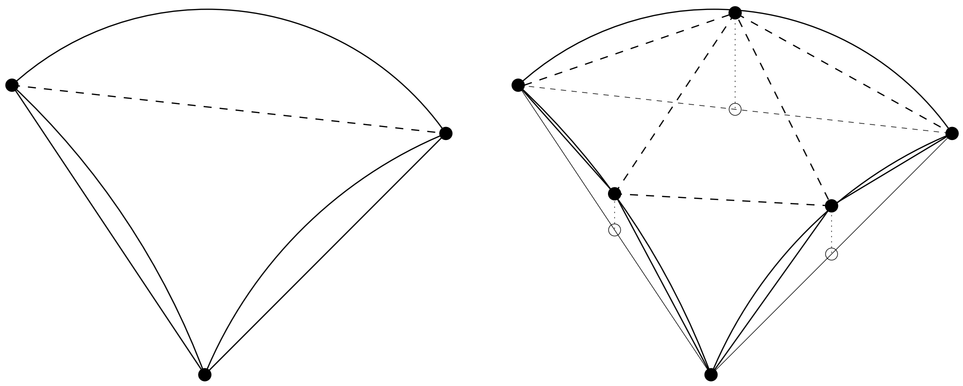


FIG. 1.8 – *Raffinement du maillage dans les zones à forte courbure*

délisation, la simulation et la déformation des matériaux. Elles constituent un outil très puissant en vision par ordinateur de par leurs propriétés topologiques. Cela reste un sujet de recherche très ouvert et de nouvelles méthodes de modélisation, de déformation et de visualisation ne manqueront pas de voir le jour rapidement.

Chapitre 2

La force interne: travaux sur la courbure

2.1 Introduction

L'élaboration d'un modèle déformable pour des applications en ségmentation ou en simulation nécéssite la définition de plusieurs "ingrédients". Il faut choisir le principe de modélisation (à base de surfaces implicites, bien sûr ...), le principe de déformation (forces internes et externes) et celui de visualisation. Le but de ce paragraphe est d'étudier le comportement du modèle choisi lors de l'application de différentes forces internes.

Il est à noter que tous les travaux exposés dans ce paragraphe ont été effectués en dimension deux, pour des raisons de place mémoire et de rapidité de calcul.

2.2 Le modèle implicite utilisé

2.2.1 Une image binaire lissée

Le modèle implicite utilisé est celui d'une fonction binaire (intérieur-extérieur), légèrement lissée par un filtrage gaussien, dont les valeurs varient continûment entre 0 et 255. Le contour auquel on s'intérèsse est la ligne de niveau 128 de cette image (voir figures 2.1 et 2.2).

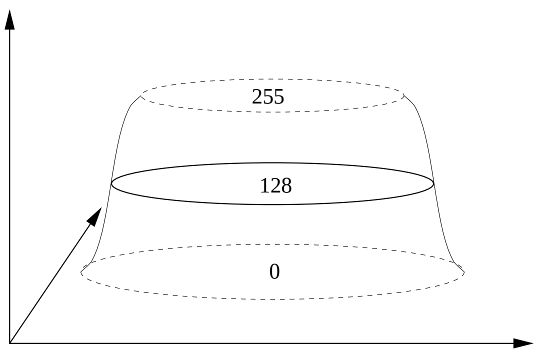


FIG. 2.1 – *Représentation d'un cercle avec le modèle implicite choisi*

FIG. 2.2 – *Plusieurs exemples d'images avec leurs isocontours correspondants*

2.2.2 L'utilisation des *Level Set Methods*

Comme nous l'avons vue dans le paragraphe 1.4, la formule $\Phi_t + F|\nabla\Phi| = 0$ peut être considérée comme l'équation fondamentale des *Level Set Methods*. F correspond à la composante normale de la vitesse de chaque point de l'image. Par exemple en prenant $F = \pm 1$, on a alors une image qui se déforme sous la seule influence de son gradient et qui va donc se contracter ou se dilater suivant le signe de F. Ceci est représenté sur la figure 2.3.

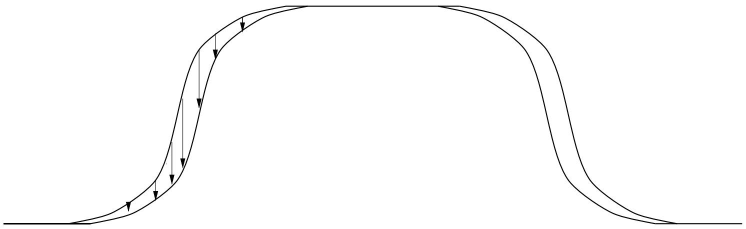


FIG. 2.3 – *Déformation d'une image sous la force de son gradient*

Mais ce mode de déformation pose des problèmes de stabilité dûs à la formation de **coins**, comme le montre la figure 2.4.

Ce genre de problème peut être évité en rajoutant un terme qui va lisser les zones à forte courbure ($\mathcal{K}$) et donc interdire la formation des coins. F est alors de la forme $F = 1 - \varepsilon\mathcal{K}$. On gère ainsi la "douceur" de l'image en modifiant le paramétre ε.

2.3 Modes de déformation de l'image

Les fonctions F présentées dans le paragraphe précédent ont en commun le fait de provoquer un mouvement permanent de l'isocontour, quelle que soit la forme initiale de celui-ci.

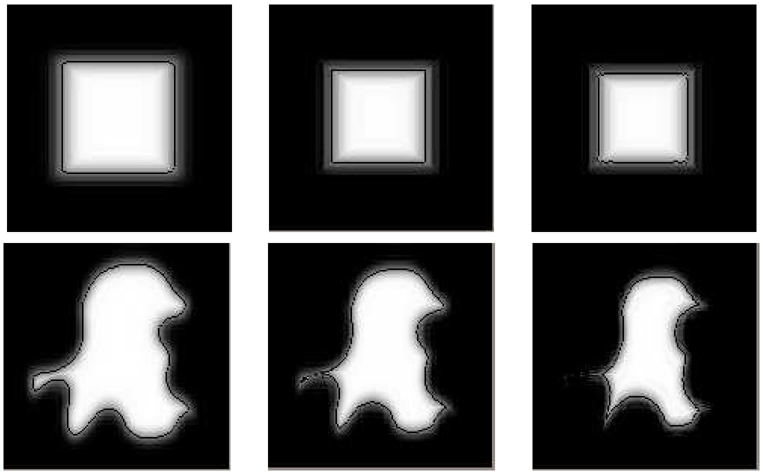


FIG. 2.4 – *Formation de coins lors de l'évolution sous la force du gradient*

Pour une application en segmentation on peut y trouver des avantages (voir [RMV95]), mais il faut néanmoins développer une force supplémentaire pour "stopper" ce mouvement. Par contre en simulation, de telles fonctions F sont inutilisables. Voici donc les différentes solutions proposées pour remédier au problème.

2.3.1 La courbure

En prenant $F = -\mathcal{K}$, et en le reportant dans l'équation (1.10), on trouve l'équation différentielle suivante:

$$\Phi^{t+\delta t} = \Phi^t + \delta t.\mathcal{K}|\nabla\Phi| \tag{2.1}$$

$\mathcal{K}$ est la courbure de la ligne de niveau passant par le point considéré. Elle se calcule à partir des dérivées partielles du premier et second ordre de l'image à l'aide de la formule suivante (le calcul complet est présenté dans l'annexe A):

$$\mathcal{K} = \nabla.\frac{\nabla\Phi}{|\nabla\Phi|} = \frac{\Phi_{xx}\Phi_y^2 - 2\Phi_x\Phi_y\Phi_{xy} + \Phi_{yy}\Phi_x^2}{(\Phi_x^2 + \Phi_y^2)^{3/2}} \tag{2.2}$$

Où les dérivées partielles sont calculées par la méthode des différences finies **centrées**, en utilisant les masques ci-dessous:

$$\begin{gathered}\Phi_x = \tfrac{1}{2}\begin{pmatrix} -1 & 0 & 1 \end{pmatrix} \qquad \Phi_y = \tfrac{1}{2}\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \\ \Phi_{xy} = \tfrac{1}{4}\begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{pmatrix} \qquad \Phi_{xx} = \begin{pmatrix} 1 & -2 & 1 \end{pmatrix} \qquad \Phi_{yy} = \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}\end{gathered} \tag{2.3}$$

Les résultats de la déformation étaient prévisibles: les fortes oscillations sont attenuées en premier, et toute courbe fermée évolue vers un cercle qui lui-même se contracte pour disparaître en un point. Ces résultats ont été démontrés par Grayson dans [Gra87]. De telles déformations sont présentées sur la figure 2.5.

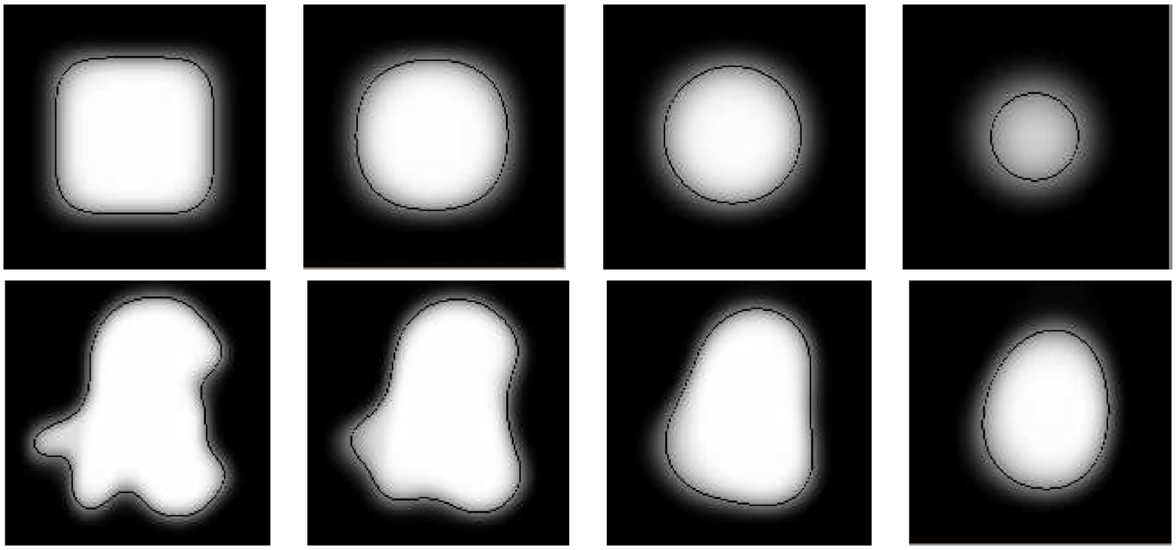


FIG. 2.5 – *Images déformées en utilisant la courbure*

Le problème du mouvement permanent de l'isocontour n'est donc pas réglé.

2.3.2 La dérivée de la courbure

Dans le but d'obtenir un contour dont l'évolution se stabilise, j'ai ensuite considéré la dérivée de la courbure prise le long de chaque isocontour. En se référant aux travaux de Monga, Lengagne et Deriche ([OMD94]), on montre que la courbure peut s'écrire sous la forme:

$$\mathcal{K} = \frac{-t^T H t}{||\vec{g}||} \tag{2.4}$$

Où $\vec{g} = \begin{vmatrix} \Phi_x \\ \Phi_y \end{vmatrix}$ est le gradient de l'image, $\vec{t}$ est le vecteur tangent au contour et H est la matrice du Hessien de l'image: $H = \begin{bmatrix} \Phi_{xx} & \Phi_{xy} \\ \Phi_{xy} & \Phi_{yy} \end{bmatrix}$. Alors en dérivant cette formule dans la direction du contour, Monga, Lengagne et Deriche ont montré que:

$$\frac{d\mathcal{K}}{ds} = \frac{3\left(t^T H t\right)\left(g^T H t\right) - ||\vec{g}||^2 t^T \begin{pmatrix} t^T H_x t \\ t^T H_y t \end{pmatrix}}{||\vec{g}||^3} \tag{2.5}$$

Où

$$H_x = \begin{bmatrix} \Phi_{xxx} & \Phi_{xxy} \\ \Phi_{xxy} & \Phi_{xyy} \end{bmatrix} \quad et \quad H_y = \begin{bmatrix} \Phi_{xxy} & \Phi_{xyy} \\ \Phi_{xyy} & \Phi_{yyy} \end{bmatrix}$$

Alors il ne reste plus qu'à faire quelques opérations sur les matrices à l'aide de Maple (logiciel de calcul formel), pour trouver finalement, en fonction des dérivées partielles de l'image, la formule suivante:

$$\frac{d\mathcal{K}}{ds} = \frac{1}{\left(\Phi_x^2 + \Phi_y^2\right)^3} \begin{bmatrix} -\Phi_{xxx}\Phi_y^3\left(\Phi_x^2+\Phi_y^2\right) + \Phi_{yyy}\Phi_x^3\left(\Phi_x^2+\Phi_y^2\right) \\ +3\Phi_{xxy}\Phi_x\Phi_y^2\left(\Phi_x^2+\Phi_y^2\right) - 3\Phi_{xyy}\Phi_x^2\Phi_y\left(\Phi_x^2+\Phi_y^2\right) \\ +3\Phi_{xx}\Phi_{yy}\Phi_x\Phi_y\left(\Phi_x^2-\Phi_y^2\right) + 6\Phi_{xy}^2\Phi_x\Phi_y\left(\Phi_x^2-\Phi_y^2\right) \\ +3\Phi_{xx}\Phi_y^3\left(\Phi_x\Phi_{xx}+\Phi_y\Phi_{xy}\right) - 3\Phi_{yy}\Phi_x^3\left(\Phi_y\Phi_{yy}+\Phi_x\Phi_{xy}\right) \\ +9\Phi_x^2\Phi_y^2\Phi_{xy}\left(\Phi_{yy}-\Phi_{xx}\right) \end{bmatrix} \tag{2.6}$$

Les masques utilisés pour les différences finies sont les mêmes que pour la courbure, avec en plus les dérivées du troisième ordre (les calculs complets de ces masques sont regroupés dans l'annexe B):

$$\Phi_{xxx} = \tfrac{1}{2}\begin{pmatrix} -1 & 2 & 0 & -2 & 1 \end{pmatrix} \qquad \Phi_{yyy} = \tfrac{1}{2}\begin{pmatrix} -1 \\ 2 \\ 0 \\ -2 \\ 1 \end{pmatrix} \tag{2.7}$$

$$\Phi_{xxy} = \tfrac{1}{2}\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & -2 & 1 \end{pmatrix} \qquad \Phi_{xyy} = \tfrac{1}{2}\begin{pmatrix} -1 & 0 & 1 \\ 2 & 0 & -2 \\ -1 & 0 & 1 \end{pmatrix}$$

Plusieurs problèmes apparaissent à ce niveau. Tout d'abord, le calcul de la dérivée de la courbure correspond à des **dérivées d'ordre trois de l'image**, avec toutes les instabilités numériques que cela implique. Ce problème a été pour une grande part résolu en calculant la dérivée de la courbure sur une version filtrée de l'image considérée; le reste des calculs étant fait sur l'image d'origine, afin de ne pas perdre d'informations. Pour cela j'ai utilisé des filtres gaussiens discrets, dont les coefficients sont ceux de la l^{ieme} ligne du triangle de Pascal (l est la longueur du filtre). Pratiquement, selon le pas de temps utilisé, des filtres de longueur variant de 3 à 13 ont été nécéssaires à la stabilisation du processus.

Une fois les instabilités numériques écartées, les premiers essais ont été effectués sur un cercle. La courbure d'un cercle étant par définition constante le long de celui-ci, on pouvait s'attendre à ce que le contour n'évolue pas. Ceci fut confirmé lors de la simulation, permettant au passage de mettre à l'épreuve la stabilité numérique de l'algorithme.
Pourtant sur des images autres que le cercle, les résultats n'étaient pas significatifs, ce qui m'a poussé à examiner de plus près le comportement théorique de ce genre de déformation. Voici ce que j'ai remarqué (ces explications s'appuient sur la figure 2.6):

- Considérons une oscillation de notre contour. La courbure est maximale en haut de l'oscillation et minimale (négative) en bas. En ces points la dérivée de la courbure est nulle et donc ils ne bougeront pas (figure 2.6-(a)). On voit déjà apparaitre le problème: comment résorber une oscillation sans en modifier les points extrémaux?

- Entre le minimum et le maximum, la courbure ne fait qu'augmenter, et donc sa dérivée est positive. De même, de l'autre coté de l'oscillation, elle est négative (figure 2.6-(b)).
- On en déduit alors la déformation de cette oscillation, qui va être translatée, mais surtout **amplifiée** (figure 2.6-(c)).

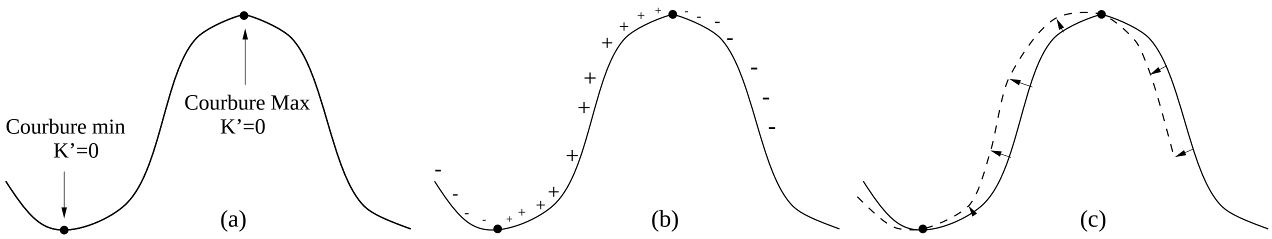


FIG. 2.6 – *Les problèmes de l'évolution sous l'action de la dérivée de la coubure*

Voici ce que produit ce genre de phénomène sur un cercle, volontairement légèrement perturbé. Les oscillations se propagent autour du cercle en s'amplifiant, comme on peut le voir sur la figure 2.7.

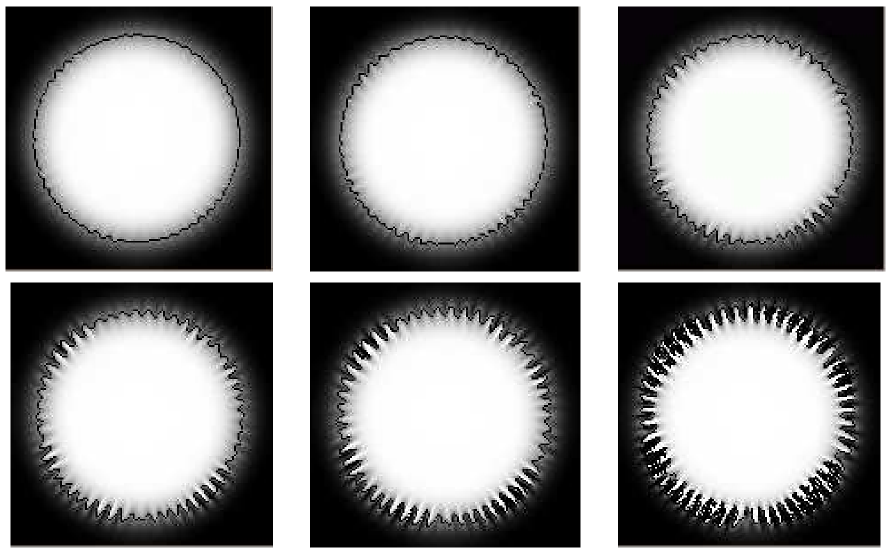


FIG. 2.7 – *Images déformées en utilisant la dérivée de la courbure*

Il faut donc se tourner vers un autre mode de déformation.

2.3.3 La courbure moyenne sur un intervalle: notion d'échelle

Pour rendre la courbure constante, on peut aussi faire tendre la courbure vers la courbure moyenne de l'image calculée sur un certain voisinage. C'est ce qu'on fait en choisissant pour la force d'évolution la forme suivante:

$$\begin{aligned} \mathcal{F} &= \left[\frac{1}{(2u_0+1)(2v_0+1)} \int_{u-u_0}^{u+u_0} \int_{v-v_0}^{v+v_0} \mathcal{K}(x,y)dxdy \right] - \mathcal{K}(u,v) \\ &= \mathcal{K}_{moy}(u,v) - \mathcal{K}(u,v) \end{aligned} \tag{2.8}$$

Comme dans le cas précédent, la courbure est calculée sur une image lissée, mais dans ce cas, un filtre de longueur trois suffit.
Les paramètres u_0 et v_0 déterminent la taille du voisinage considéré, et donc **l'échelle** à laquelle les oscillations vont être lissées. Sur les figures 2.9 et 2.8 sont présentés les résultats obtenus par cette méthode sur une image comportant des oscillations à plusieurs échelles.

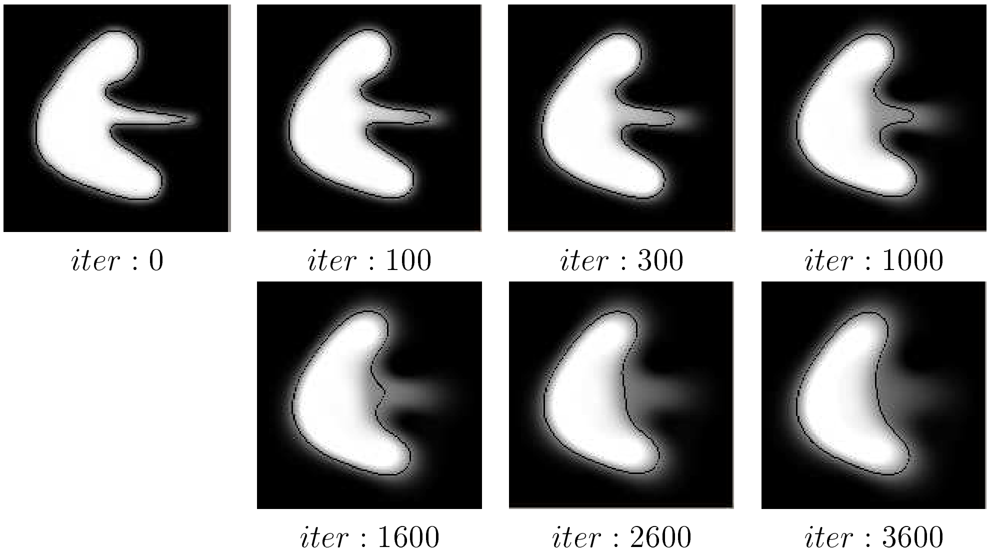


FIG. 2.8 – *Images déformées en utilisant la courbure moyenne avec un voisinage (7 x 7)*

On remarque que, dans le premier cas, certaines oscillations à moyenne échelle sont conservées alors que dans le dernier cas elles sont lissées. On pourra ainsi choisir la taille des oscillations que l'on désire conserver.
En regardant de plus près la signification mathématique d'une telle force de déformation, on arrive à la conclusion suivante: le calcul de $\mathcal{K}_{moy}$ n'est autre que l'application sur l'image du masque $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$. Alors $\mathcal{K}_{moy} - \mathcal{K}$ peut être identifié au masque $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, qui n'est autre que celui de la dérivée seconde de la courbure de l'image.
La figure 2.10 montre comment le calcul est effectué sur l'image à l'aide du masque.

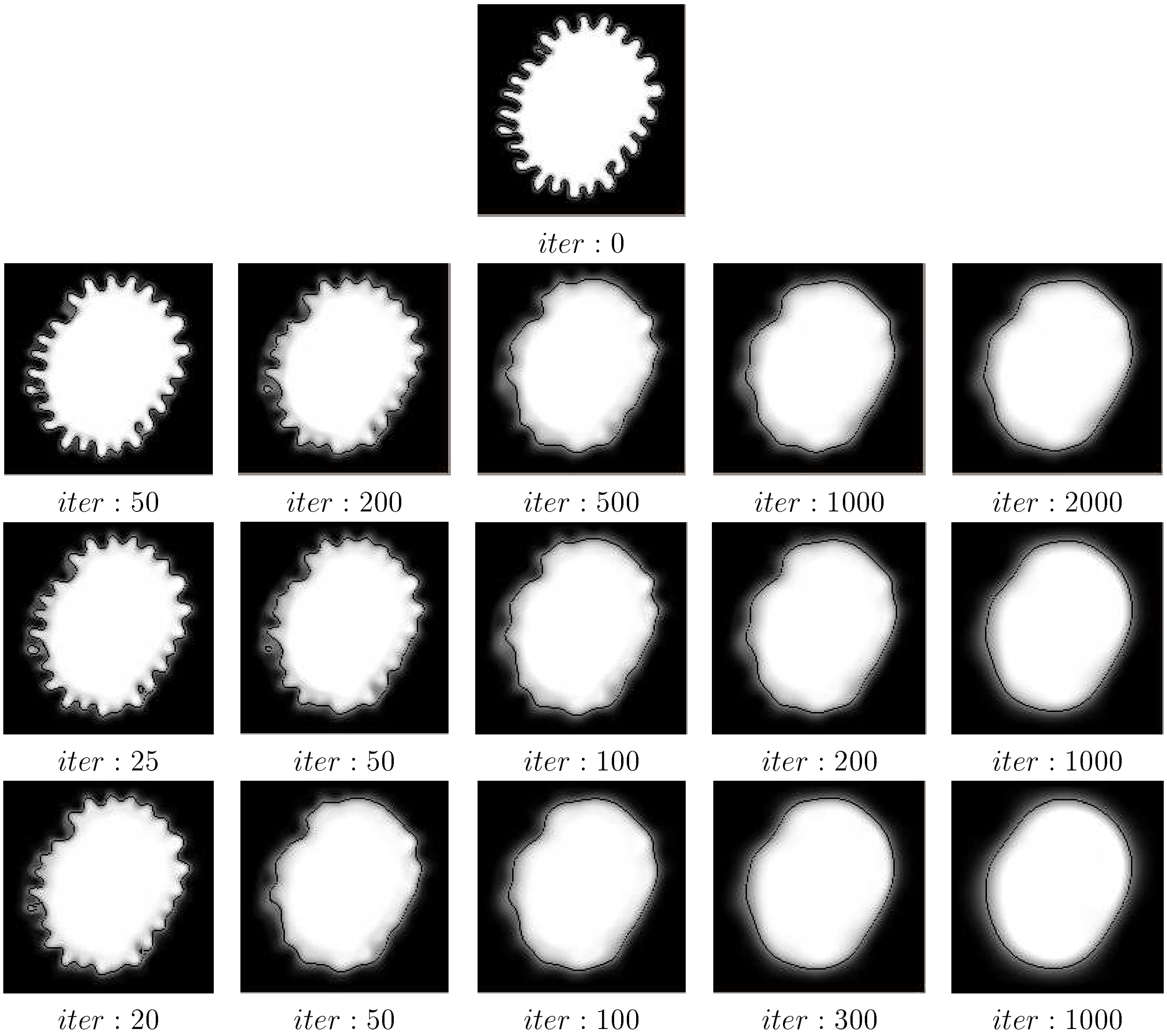


FIG. 2.9 – *Images déformées en utilisant la courbure moyenne, pour des voisinages de (3x3) pour la première ligne, (7x7) pour la seconde et (11x11) pour la troisième*

Comme on peut le voir, les résultats de cette méthode sont bons. Cependant, il faut remarquer que les masques utilisés sont **isotropes**, et que par conséquent un tel filtrage entraîne une perte d'information par lissage. Nous voudrions une force interne qui lisse l'image dans la direction des lignes de niveau, tout en ne changeant rien dans celle du gradient. Ceci nous amène à parler de diffusion **anisotrope**.

FIG. 2.10 – *Application du masque déduit de* $\mathcal{F} = \mathcal{K}_{moy}(u,v) - \mathcal{K}(u,v)$

2.3.4 La diffusion anisotrope

La diffusion anisotrope est dérivée des techniques de filtrage par convolution avec des fonctions gaussiennes. Mais dans ce cas particulier, le traîtement n'est pas le même dans toutes les directions, l'idée étant de lisser l'image uniquement dans la direction normale au gradient. Dans ce but, Perona et Malik [PM90] ont introduit l'équation de diffusion suivante:

$$\frac{\partial u}{\partial t} = div(g(|\nabla u|).\nabla u) \tag{2.9}$$

La convergence de cette équation est assurée pour les fonctions g qui ont certaines propriétés [DF95]. Pour montrer cette convergence, l'équation précédente est exprimée comme la minimisation de la fonction d'énergie suivante:

$$E(u) = \lambda \int_{\Omega} \Psi(|\nabla u|) d\Omega \tag{2.10}$$

Où $\frac{\Psi'(|\nabla u|)}{|\nabla u|} = g(|\nabla u|)$

L'équation 2.9 est une descente de gradient le long de la surface d'énergie $E(u)$. En remplaçant g par sa valeur en fonction de Ψ dans l'équation 2.9, on trouve la formule:

$$\frac{\partial u}{\partial t} = \Psi"(|\nabla u|) u_{\xi\xi} + \frac{\Psi'(|\nabla u|)}{|\nabla u|} u_{\eta\eta} \tag{2.11}$$

Où $\xi = \frac{\nabla u}{|\nabla u|}$ est le vecteur unitaire dans la direction du gradient, et η le vecteur orthogonal au gradient.

Grâce à la collaboration de Karl Krissian (en première année de thèse dans le projet Epidaure), qui a programmé cet algorithme ([KMA96]), j'ai pu appliquer une diffusion anisotrope sur mes images, et comparer ces résultats avec les miens.

Les résultats sont bons, et on a reussi à enlever les petites oscillations en conservant les moyennes. Par contre, on observe un effet d'étalement des contours de l'image qui n'est pas souhaitable par exemple lors d'une application en segmentation: on verrait l'image se lisser au cours du temps, allant même jusqu'à effacer certaines parties de l'objet simulé.

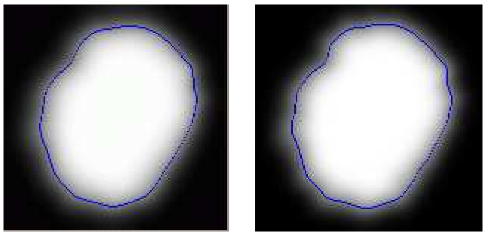


FIG. 2.11 – *Diffusion anisotropique*

Il faut cependant retenir de cette expérience l'idée d'anisotropie du filtre qui doit être utilisé: il faut utiliser un filtre qui calcule la dérivée seconde de la courbure dans le direction perpendiculaire au gradient. Ceci nous mène tout naturellement vers le paragraphe suivant.

2.3.5 La dérivée seconde de la courbure le long du contour

Le but de ce paragraphe est d'étudier le comportement d'une image déformée sous l'action de la dérivée seconde de sa courbure prise dans une direction privilégiée: celle des lignes de niveau.
Si on se reporte au paragraphe consacré à la dérivée première de la courbure, on s'aperçoit qu'une des méthodes envisageables serait de dériver une nouvelle fois la formule 2.6. La lourdeur et l'instabilité numérique d'une telle équation m'ont incité à opérer différemment. Plusieurs solutions ont été implémentées.

La première, qui est aussi la plus simple, consiste dans un premier temps à calculer la courbure en tous les points de l'image où elle existe, à partir de l'image légèrement lissée. Un masque de dérivée seconde monodimensionnel est alors appliqué sur cette image de courbure, dans la direction perpendiculaire au gradient de l'image (c'est à dire parallèle à la ligne de niveau passant par le point considéré). Cette direction est calculée sur l'image elle-même, comme on peut le voir sur la figure 2.12.

La formule de $\mathcal{F}$ est alors:

$$\mathcal{F} = \mathcal{K}_{\eta\eta} = \Delta\mathcal{K} - \mathcal{K}_{\xi\xi} \tag{2.12}$$

Où

$$\mathcal{K}_{\xi\xi} = \nabla(\nabla\mathcal{K}.\xi).\xi = \nabla(|\nabla\mathcal{K}|).\frac{\nabla\mathcal{K}}{|\nabla\mathcal{K}|} \tag{2.13}$$

ξ et η sont les vecteurs unitaires dans la direction du gradient et du contour.

Les résultats obtenus par cette méthode sont intéréssants (voir figure 2.13). En particulier on note un très faible étalement de l'image, même au bout d'un grand nombre d'itérations (à comparer avec les résultats de la diffusion anisotrope: figure 2.11).
Le seul inconvénient est la faible extension spatiale du filtre (trois pixels seulement), ce qui entraîne un très bon lissage des hautes fréquences (on peut envisager une application pour le débruitage des isosurfaces), mais par contre une très faible efficacité sur les plus larges oscillations.

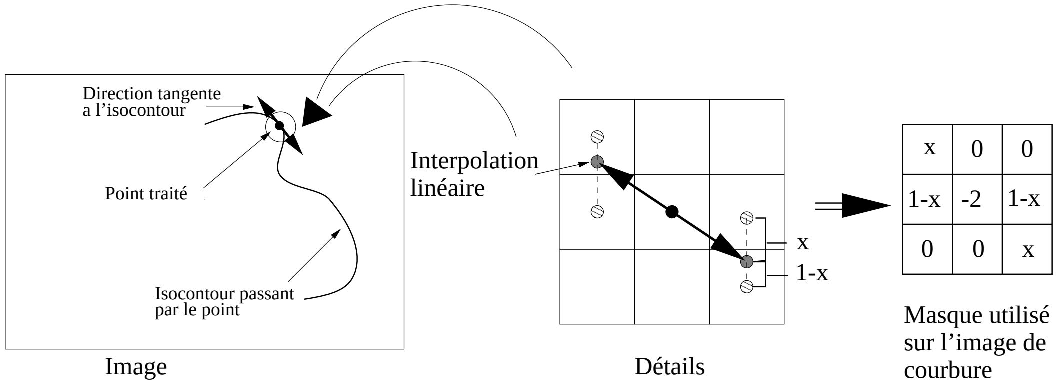


FIG. 2.12 – *Détermination de la direction du vecteur tangent à l'isocontour en vue du calcul de la dérivée seconde directionnelle de la courbure de l'image*

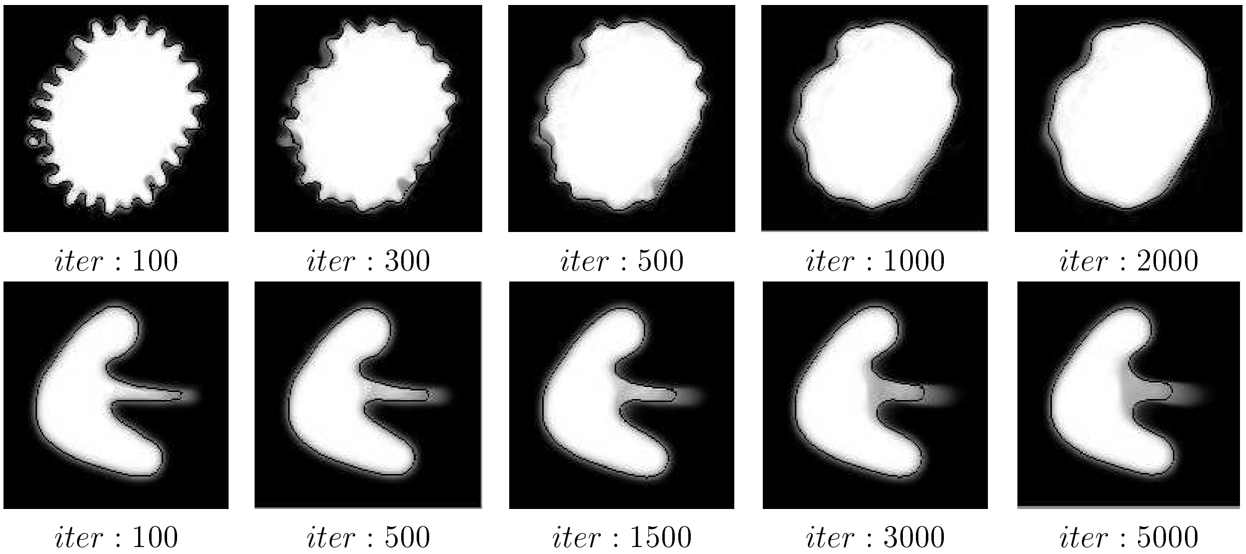


FIG. 2.13 – *Images déformées en utilisant la dérivée seconde de la courbure*

Le nouveau problème à surmonter est donc d'étendre la méthode à un voisinage de taille supérieure. On ne peut malheureusement pas simplement augmenter la taille du filtre de dérivée et l'appliquer comme précédemment. On désire calculer la derivée seconde de la courbure le long du contour. Dans le cas d'un filtre de taille trois, le contour est correctement approximé par la tangente. Mais ce n'est plus vrai pour des filtres plus longs, qui vont engendrer des erreurs aux points de forte courbure, qui sont justement ceux qui nous intérèssent le plus (figure 2.14).

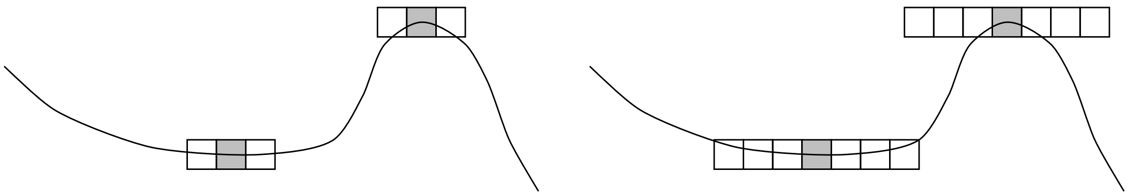


FIG. 2.14 – *Approximation du contour par la tangente*

Si on veut augmenter la taille du filtre, il faut s'arranger pour placer celui-ci réellement le long du contour, c'est à dire, pour chaque point à traiter, connaître un certain nombre de ces "voisins de ligne de niveau". Ceci sous-entend la détermination de toutes les lignes de niveau de l'image, et donc un temps de calcul assez long.
Une première amélioration est directement dérivée du modèle implicite choisi. Notre image étant binaire lissée, les points à 0 et 255 ne doivent pas être remis à jour. On se restreint donc à traiter les points de l'image où la norme du gradient est différente de zéro, ou même supérieure à un certain seuil pour éviter les instabilités liées à l'incertitude numérique sur la direction du gradient. On peut donc penser à ne calculer qu'un nombre réduit d'isocontours.

Pour la deuxième méthode mise en œuvre, ce raisonnement a été poussé à l'extrême, en choisissant de n'appliquer la déformation que sur les points appartenant à l'isocontour qui nous intérèsse: l'isocontour 128. L'idée est de faire suivre le mouvement aux autres points de manière simple et surtout efficace. Les premiers essais ont été effectués avec un trés léger lissage des points extérieurs à l'isocontour 128, afin de conserver la notion de "binaire lissée" au cours de la déformation. Dans ce cas la force interne vaut:

$$\begin{aligned} \mathcal{F} &= \left[\frac{1}{2u_0+1} \int_{u-u_0}^{u+u_0} \mathcal{K}(s)ds \right] - \mathcal{K}(u) \\ &= \mathcal{K}_{moy}(u, u_0) - \mathcal{K}(u) \end{aligned} \tag{2.14}$$

Où s représente l'isocontour 128.
La figure 2.15 montre comment est appliqué le masque le masque sur les points de l'isocontour 128 de l'image.

Les résultats présentés sur la figure 2.16 ont été obtenus par cette méthode. Sur la première ligne, les calculs sont faits avec un filtre de longueur totale 11 (5 points de chaque coté du point traité), jusqu'à la convergence numérique (symbolisée par "inf"), puis avec des filtres de longueur supérieure. Sur la deuxième ligne, les oscillations de l'image initiale étant larges, j'ai directement utilisé un filtre de grande taille.

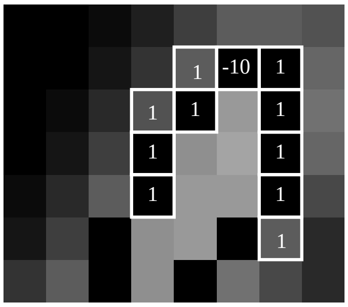


FIG. 2.15 – *Calcul de la dérivée seconde de la courbure le long de l'isocontour 128 pour un voisinage de taille 11*

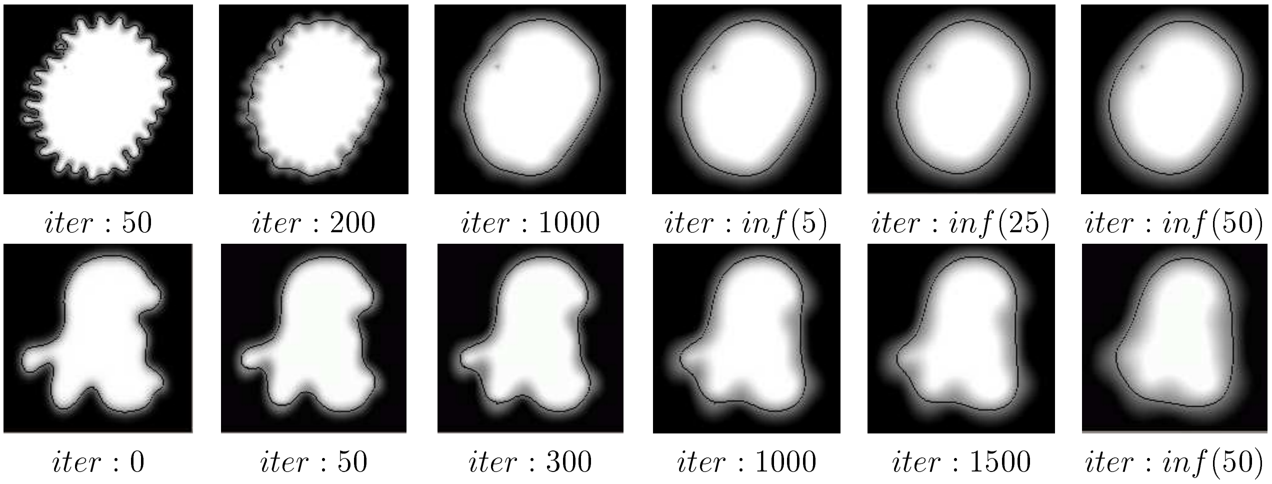


FIG. 2.16 – *Images déformées en utilisant la dérivée seconde de la courbure le long de l'isocontour 128*

Une remarques doit être faite concernant ces résultats. le gain en nombre d'itérations ne semble pas évident. En effet les instabilités numériques obligent à garder un pas de temps très faible. Mais c'est au niveau du temps de calcul qu'on gagne énormement. Contrairement aux autres méthodes, les calculs couteux (dérivée 4^{ieme} de l'image) ne sont faits que sur les points de l'isocontour 128. On gagne ainsi une dimension!

Mais il y a un meilleur moyen d'obliger tous les points de l'image à suivre l'évolution de l'isocontour. La solution que je propose est la suivante. Pour un point de l'isocontour, on calcule son déplacement, que l'on propage ensuite vers tous les points situés sur la droite perpendiculaire au contour passant par le point. Cette propagation est obtenue en pondérant le déplacement du point de l'isocontour par la norme du gradient calculée au point courant (voir figure 2.17). On se rapproche ainsi de la notion de filtrage anisotrope, qui lisse l'image le long du contour, mais la conserve inchangée dans la direction du gradient.

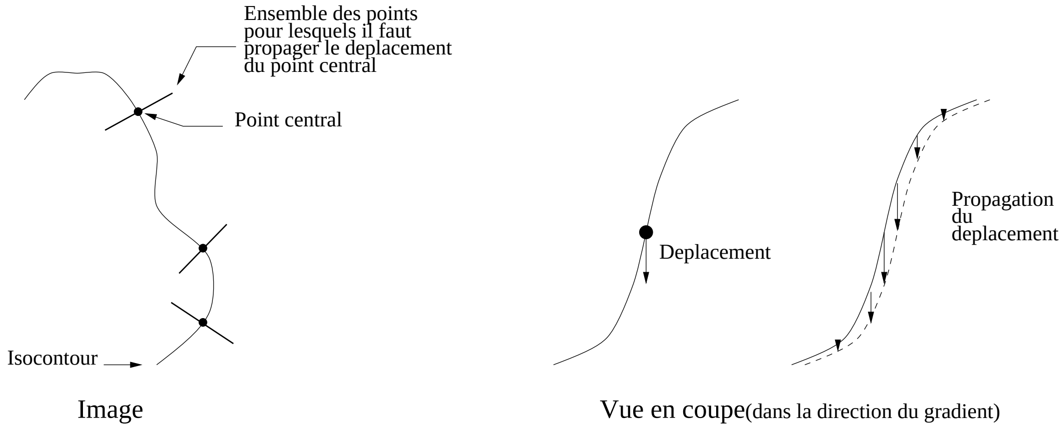


FIG. 2.17 – *Conservation de l'image dans la direction du gradient*

En plus de conserver la forme du modèle implicite, cette technique a un autre avantage, qui est d'augmenter la vitesse de déplacement. En effet dans le cas où seul le point sur l'isocontour est modifié, celui-ci ne se déplacera au mieux que d'un pixel, alors que cette nouvelle méthode, lui permet de se déplacer d'une distance d représentée sur la figure 2.18.

Si $f(x)$ est la courbe en trait plein, alors la courbe en pointillés est $f(x-d)$. En appliquant un développement de Taylor à l'ordre 1, on trouve $f(x-d) = f(x) - df'(x) + o(x^2)$. En identifiant avec l'équation (1.10), on retrouve que d vaut $\delta t\mathcal{F}$.

Malheureusement, la mise en œuvre de ce procédé présente quelques problèmes. Si, pour chaque point de l'isocontour, on propage le déplacement vers tous les points sur la droite perpendiculaire à l'isocontour, certains vont être remis à jour plusieurs fois, en particulier aux points de forte courbure. Il faut donc opérer en sens inverse, c'est à dire pour chaque point de l'image, trouver le point le plus proche sur l'isocontour 128, et en déduire le déplacement qu'il faut lui appliquer. Mais ce genre de recherche est couteuse en temps de calcul. Les résultats sont aussi bons qu'avec le lissage, mais on a en plus une conservation de toutes les caractéristiques initiales du modèle. Mais en attendant un algorithme plus efficace pour la recherche du point le plus proche sur l'isocontour, j'ai laissé de côté cette technique, trop

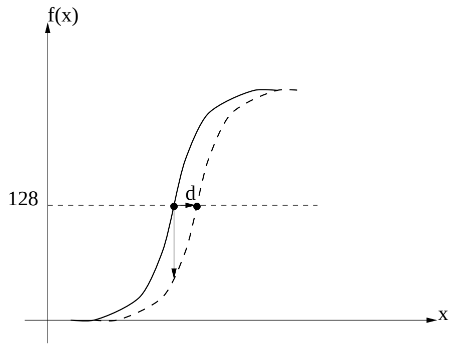


FIG. 2.18 – *Relation entre déplacement des pixels et déplacement du contour*

longue pour la phase d'expérimentation du modèle. Dans toute la suite des travaux la force interne utilisée sera donc:

- La dérivée seconde de la courbure dans le direction de l'isocontour pour les points de celui-ci.
- Un faible lissage pour les autres points de l'image où le gradient est supérieur à un certain seuil.

Chapitre 3

Forces externes

3.1 Introduction

Pour qu'un modèle déformable soit complet, il faut lui associer une ou plusieurs **forces externes**, c'est à dire définir comment un objet étranger peut intéragir et imposer des contraintes sur l'isocontour (dimension 2) ou l'isosurface (dimension 3). Si on vise une application en segmentation, on a besoin que le contour soit attiré par exemple par le gradient de l'image à ségmenter. Pour la simulation, on doit pouvoir contraindre le contour à passer par un point, ou à avoir un point fixe, ou encore à revenir vers une position d'équilibre après une déformation (c'est à dire avoir une forme de référence).

3.2 Intéragir avec un point

On peut imaginer plusieurs sortes de contraintes infligées à un isocontour par un point. En voici plusieurs exemples, avec leurs applications.

3.2.1 Faire passer l'isocontour par un point

Dans ce premier paragraphe, je vais me rapprocher d'une notion développée par Jules Bloomenthal et Ken Shoemake dans [BS91]: les *Convolution surface*. En effet, notre modèle "Binaire-Lissée" se prête extrêmement bien à ce genre de manipulation. On ne parle pas ici de déformation ou de contraintes, mais plutôt de rajouter ou enlever un morceau de l'isocontour. Si on considère un isocontour de référence, supposons qu'on veuille le faire passer par un point bien précis. Alors, on détermine le point le plus proche sur l'isocontour. Ces deux points délimitent un segment. Mathématiquement, ce segment est representé par une fonction de $\mathbb{R}^2$ dans $\mathbb{R}$ qui vaut une fonction porte dans la direction du segment et un dirac dans la direction perpendiculaire (on trouve parfois cette fonction sous le nom de **mur de Dirac**). En convoluant cette fonction avec une gaussienne, dont on peut choisir la largeur à mi hauteur, on obtient une fonction aux propriétés intéréssantes (voir figure 3.1):

- Les deux points situés aux extrémités ont pour valeur 128 (donc l'isocontour passe par ces deux points).

- La forme de la fonction est de type "binaire lissée", d'où un parfait mélange avec la fonction potentiel de départ (voir article sur les *Convolution Surfaces*).

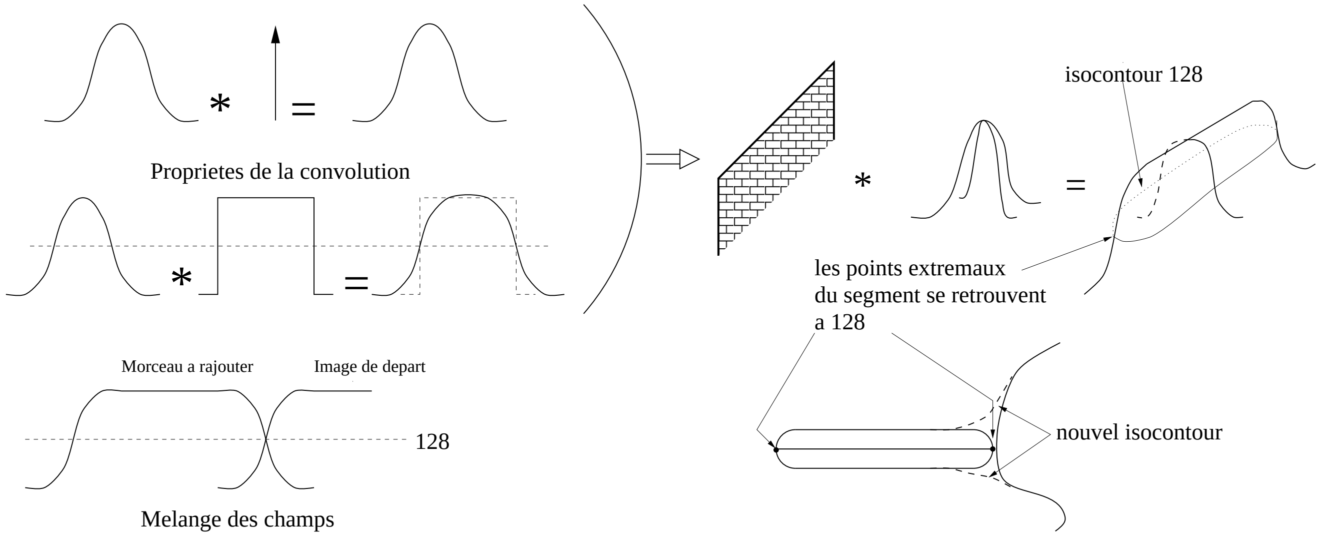


FIG. 3.1 – *Convolution d'un mur de Dirac par une gaussienne*

Deux exemples de cette technique sont proposés ci-dessous.

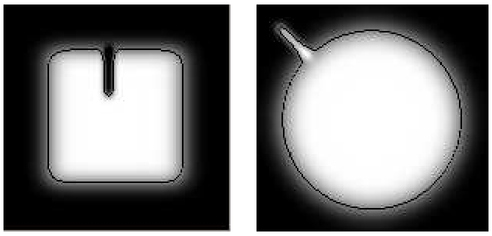


FIG. 3.2 – *Utilisation de la convolution sur notre modèle*

3.2.2 Attirer le contour sur un point

La technique de convolution proposée au paragraphe précédent présente un intérêt pour la construction de formes géométriques plus ou moins complexes dans le cadre d'applications comme les logiciels de dessin ou de sculpture (voir article [GH91]). Dans le cadre de la simulation ou de la segmentation, il est préférable de modéliser l'attraction de l'isocontour par un point. Cette force externe va tout naturellement se comporter exactement comme la force de rappel interne. Un point fixé par l'utilisateur va donc modifier son plus proche voisin sur l'isocontour. L'équation du modèle devient alors:

$$\Phi^{t+\delta t} = \Phi^t - \delta t \left(\mathcal{F}_{int} + \mathcal{F}_{ext}\right) |\nabla \Phi| \tag{3.1}$$

Où la force interne est donnée par la formule 2.14, et la force externe par la formule:

$$\mathcal{F}_{ext} = \|\overrightarrow{MM'}\| \tag{3.2}$$

M étant le point "attracteur" et M' son plus proche voisin sur l'isocontour 128.

Dans un premier temps, observons le comportement du modèle sous la double action de la force interne et de la force externe. Pour cela, prenons un cercle que l'on soumet à l'attraction d'un point. On peut voir les résultats sur la figure 3.3. La longueur du voisinage utilisé pour la force interne vaut 11 points pour la première ligne, 31 pour la seconde et 81 pour la dernière. Cette expérience met clairement en évidence la relation étroite existant entre la longueur du voisinage et le comportement dynamique du modèle (en simulation on peut utiliser cette longueur de voisinage pour caractériser la raideur des matériaux).

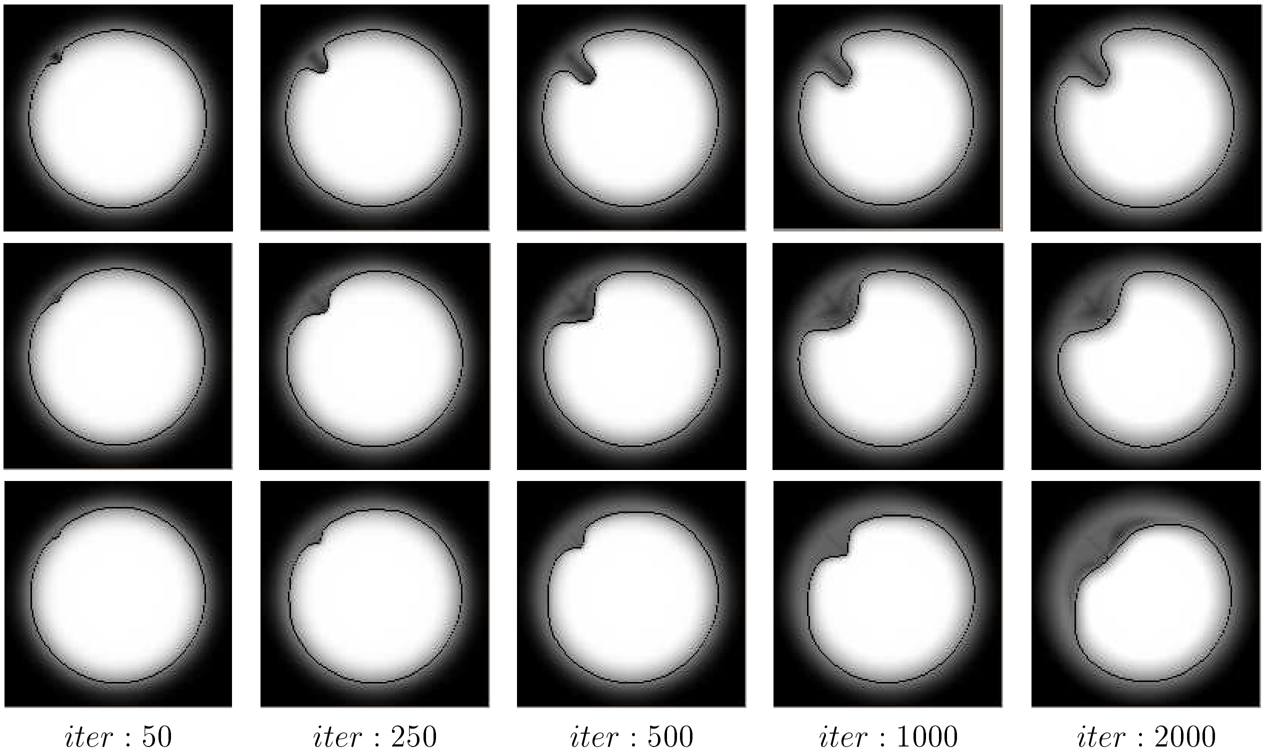


FIG. 3.3 – *Contour attiré par un point pour différentes longueurs du voisinage*

Une variante, figure 3.4, est de se servir de la technique de convolution vue au paragraphe précédent pour attirer d'un seul coup le contour jusqu'au point, ensuite de garder ce point fixe grâce à la force externe et de faire evoluer l'ensemble jusqu'à l'équilibre.

On peut aussi attirer avec plusieurs points, comme on peut le voir sur la figure 3.5. Une application possible de cet exemple serait, en segmentation, de procéder à une pre-

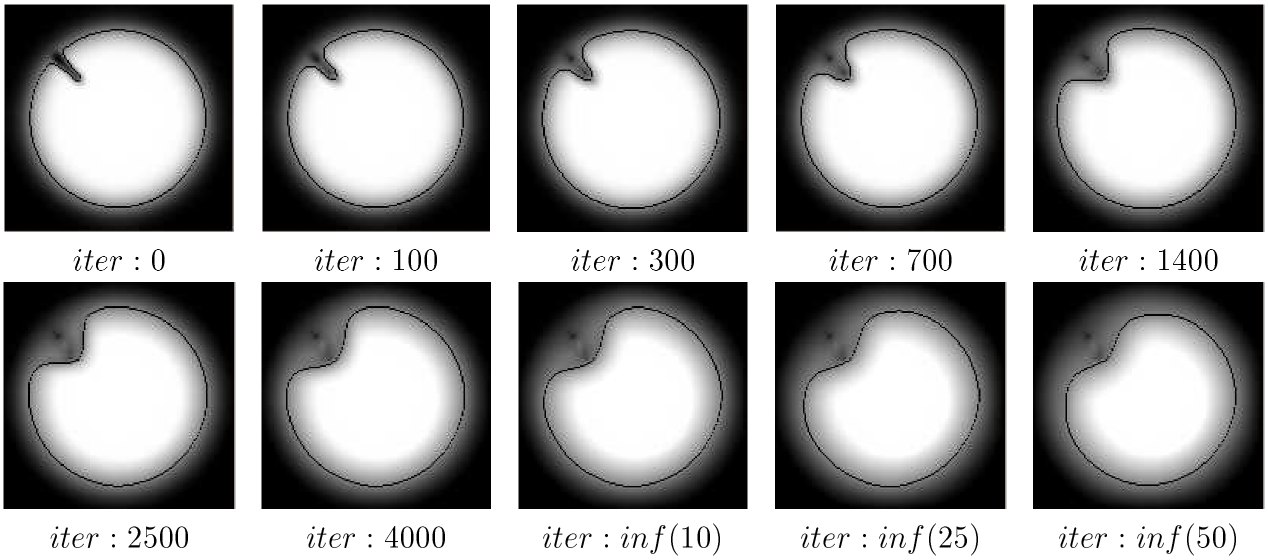


FIG. 3.4 – *Attraction instantanée par la méthode de convolution, puis évolution avec un point fixé*

miére attraction par quelques points (les extrema de courbure par exemple), qui servirait d'initialisation à un autre algorithme.

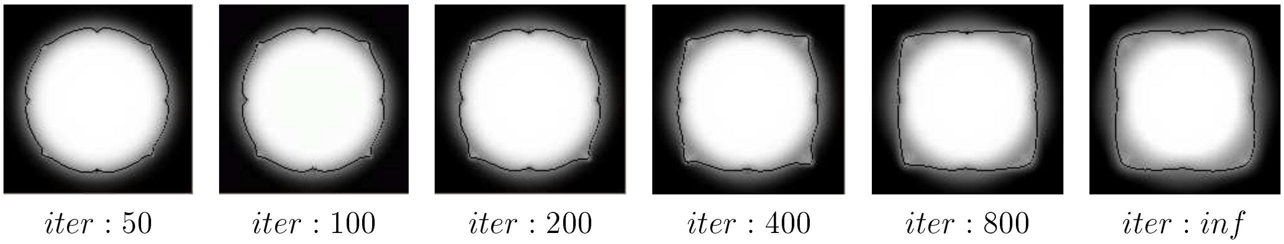
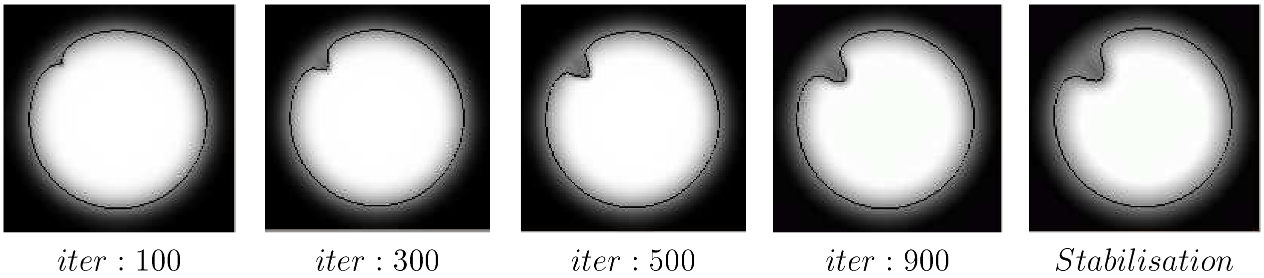


FIG. 3.5 – *Cercle attiré par huit points situés aux sommets et aux milieux des arêtes d'un carré*

Enfin, une première approche de simulation est présentée sur la figure 3.6. On fixe trois points, deux sur le cercle (pour délimiter la zone de déformation) et un à l'intérieur. La première ligne représente la phase de déformation. Ensuite on retire le point à l'intérieur, et on observe le retour à la position d'équilibre. Mais ce retour est lent, car il est uniquement engendré par la force interne de rappel. Il manque ici une notion de forme de référence, lacune que je me propose de combler dans le paragraphe suivant.



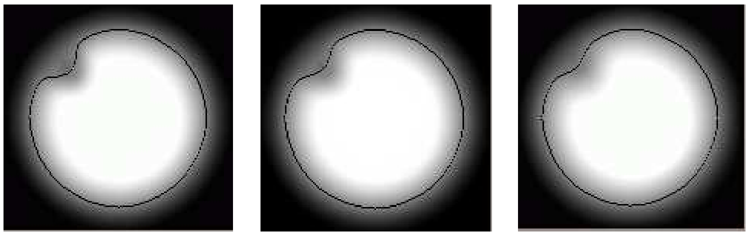


FIG. 3.6 – *Animation: cercle attiré par un point, puis retour à l'équilibre*

3.3 Forme de référence et mémoire de forme

Au lieu d'attirer un isocontour vers des points, il s'agit ici de l'attirer vers une forme de référence, c'est à dire vers une courbe. Le principe est une simple généralisation du cas du point. A chaque itération, en chaque point M de l'isocontour est appliquée une force égale à la composante normale du vecteur $\overrightarrow{MM'}$ (M' est le plus proche voisin de M sur la courbe de référence). Donc dans ce cas, la force externe est:

$$\mathcal{F}_{ext} = \overrightarrow{MM'}.\vec{n} \tag{3.3}$$

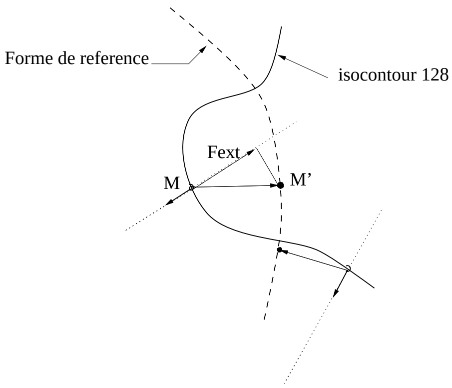


FIG. 3.7 – *Évolution vers une forme de référence*

Je propose pour illustrer ces propos deux exemples: l'évolution d'une forme vers une autre (figure 3.8 et 3.9), et le retour vers une forme initiale après déformation (figure 3.13).

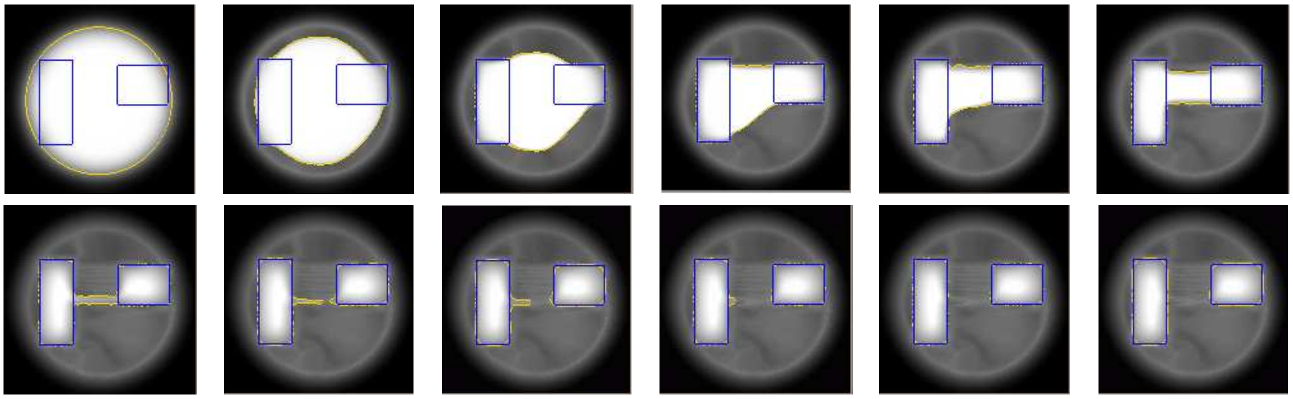


FIG. 3.8 – *Cercle attiré vers deux quadrilatères*

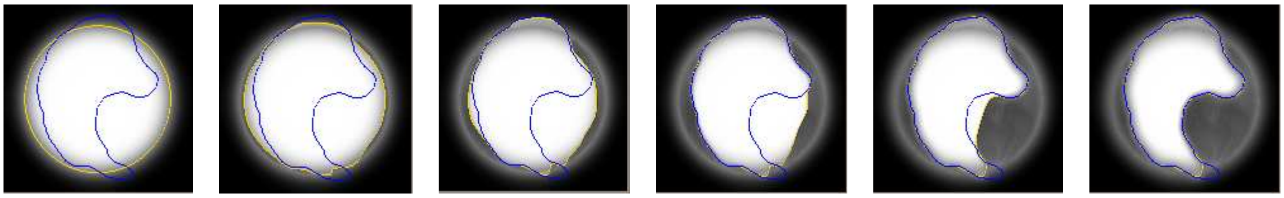


FIG. 3.9 – *Cercle attiré vers une forme de foie*

Ces résultats mettent en évidence plusieurs caractéristiques intéréssantes du modèle. D'abord, sur la figure 3.8, l'isocontour se scinde bien en deux pour aller s'appliquer sur les deux composantes de la forme de référence. Ensuite, aucune attention particuliére n'a été porter à la forme initiale du contour. On voit sur la figure 3.9 que la forme de référence dépasse vers l'extérieur du cercle en deux endroits, et que cela n'a aucune influence sur le résultat final. Ce genre d'initialisation ne conviendrait pas pour la technique de segmentation développée par Malladi, Sethian et Vemuri dans [RMV95].

On observe un petit problème au bas de la dernière image de la figure 3.9, qui met en évidence les limites de cette force externe. Si on observe de plus près, on constate que pour les points sur le contour jaune, le vecteur $\overrightarrow{MM'}$ est perpendiculaire à la direction locale du gradient, donc la force externe est nulle (ou trop faible pour compenser la force interne) (figure 3.10). La convergence vers la forme de référence va tout de même se faire, mais très lentement.

On voit en blanc sur cette figure l'alternative à ce problème. On raisonne un peu diffèremment, en considérant que chaque point de l'isocontour subit l'attraction du point de la forme de référence situé dans la direction orthogonale à l'isocontour. Le problème revient donc à déterminer l'intersection d'une droite et d'une courbe. Un algorithme très efficace, proposé par Bresenham, traîte ce problème, en y ajoutant même une certaine liberté. En

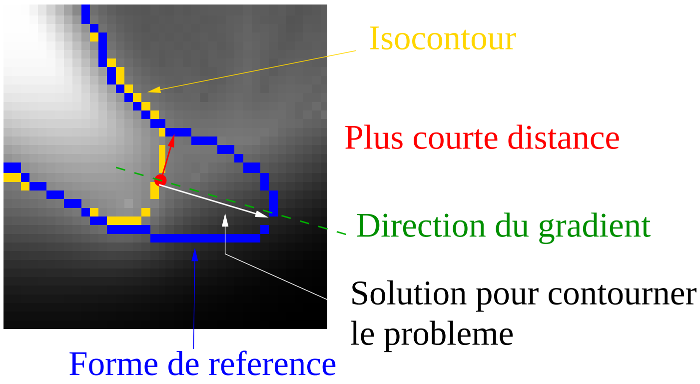
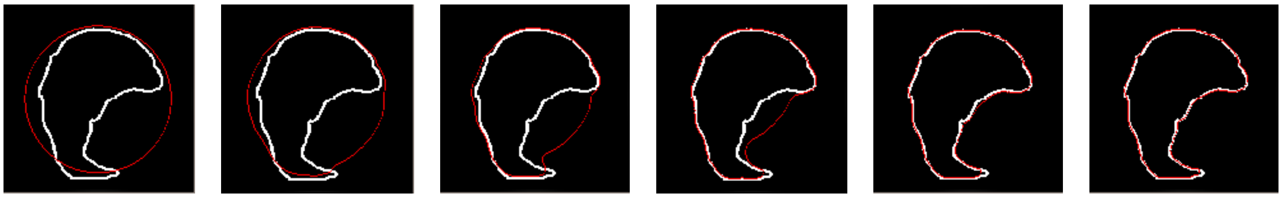


FIG. 3.10 – *Problème de l'évolution vers une forme de référence*

partant d'un point et dans une direction donné, l'algorithme parcourt la droite point à point, et teste à chaque fois si une condition est remplie. Cette condition peut être l'appartenance du point à la forme de référence: voici résolu le probléme rencontré précédemment. Mais cela ouvre la voie à plusieurs autres applications. Ce test peut porter sur la norme gradient d'une image à segmenter, ou le passage par zéro du laplacien.
J'en ai donc profité pour faire quelques applications à la segmentation. Les images que l'on peut voir sur les figures 3.11 et 3.12 sont des images synthétiques de ce que l'on peut obtenir en seuillant le gradient d'une image réelle. L'isocontour de départ est donc attiré par les points de plus fort gradient de l'image à segmenter.
Dans le premier cas, j'ai pris une image de gradient idéale, c'est à dire un contour sans trou. Dans ces conditions, l'isocontour converge vers la forme de référence quelle que soit la position initiale. Ceci peut bien sûr être appliqué pour la simulation (mémoire de forme).



$iter$: 0	$iter$: 25	$iter$: 75	$iter$: 150	$iter$: 250	$iter$: 300

FIG. 3.11 – *Attraction du contour par une image de gradient seuillé de bonne qualité*

Pour les 250 premières itérations, la force interne utilise un voisinage de 15 points pour éviter la formation d'oscillations lors du déplacement rapide du contour. Les 50 dernières utilisent un voisinage de 3 seulement, permettant ainsi au contour d'aller jusque dans les coins de la forme de référence.

J'ai ensuite effectué la même expérience avec une image de référence plus réaliste, formée presque exclusivement de points ou de petits segments. Il faut alors choisir un voisinage de grande taille pour la force interne (50 points) car certains points seulement seront attirés, et le rôle de la force interne est alors de faire suivre les autres. Comme dans le premier cas, au bout d'un certain nombre d'itérations la longueur du voisinage est reduite, pour affiner la segmentation.

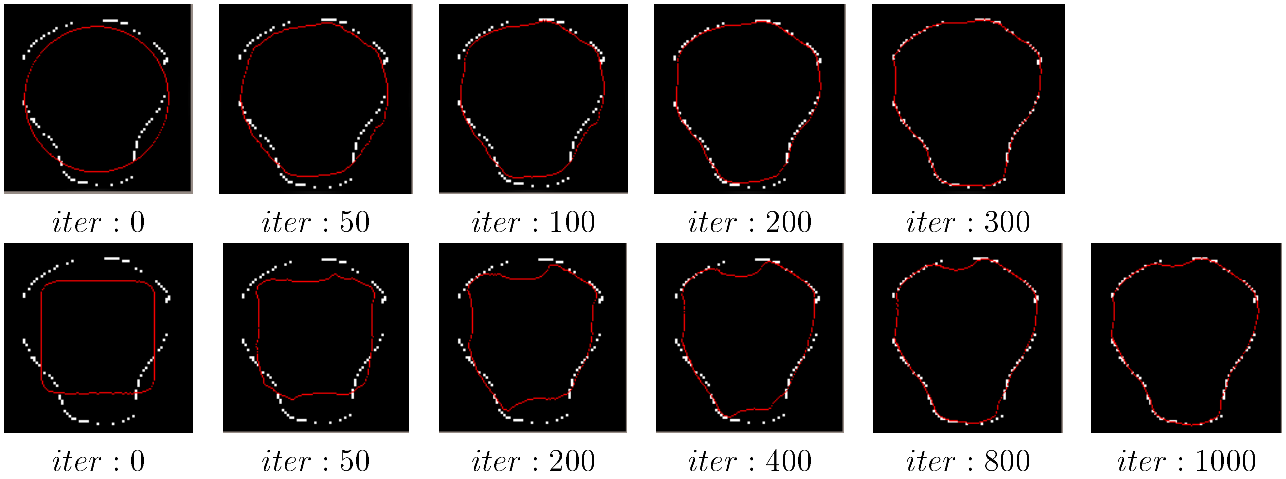


FIG. 3.12 – *Attraction du contour par une image de gradient de mauvaise qualité*

Pour la deuxième application (figure 3.13), la force qui déforme l'objet est dérivée de la méthode du paragraphe 3.2.1. Le point situé au bout de l'instrument génère un potentiel gaussien négatif qui repousse l'isocontour en faisant baisser la valeur des points qui se trouvent dans son voisinage. Lorsque l'instrument se retire, cette force est mise à zéro et l'isocontour revient à l'équilibre, attiré vers sa position d'origine qui est prise comme forme de référence.

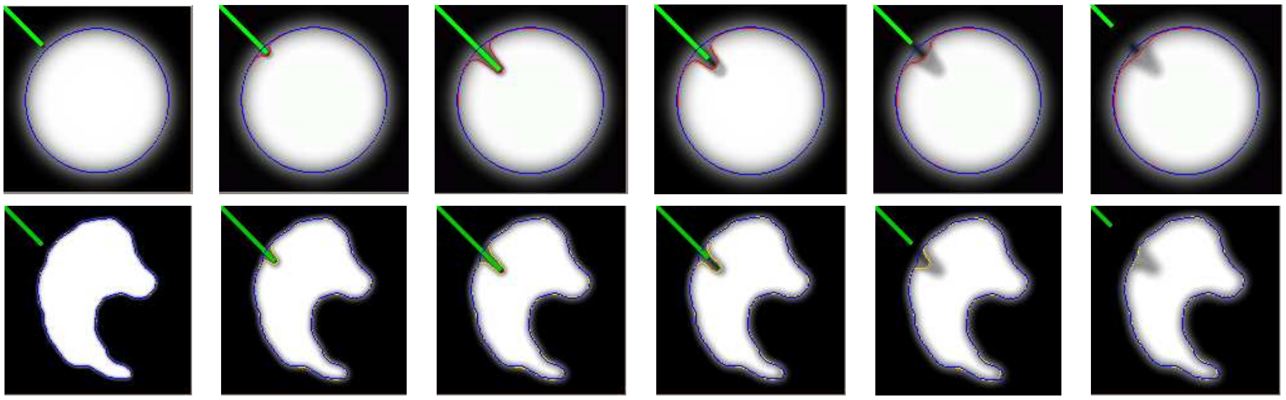


FIG. 3.13 – *Simulation d'un instrument qui déforme un objet*

Un dernier cas particulier de cette application est illustré par la figure 3.14, où l'instrunent va jusqu'à découper le foie en deux morceaux. Aprés le retrait de l'instrument, j'ai appliqué quelques itérations avec uniquement la force interne pour montrer que les deux composantes connexes évoluent indépendemment.

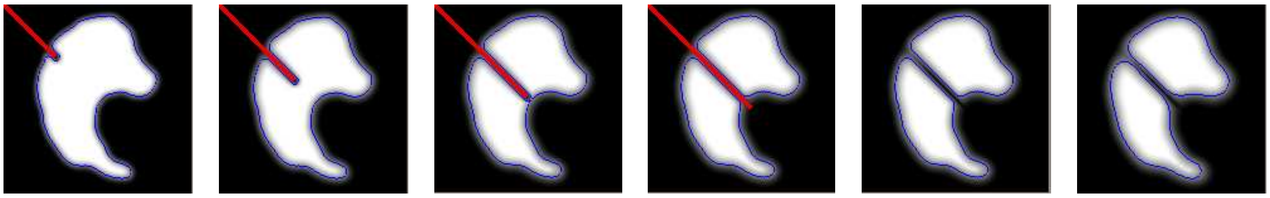


FIG. 3.14 – *Découpe d'un objet à l'aide d'un instrument*

Chapitre 4

L'interface utilisateur

Avant de passer à la conclusion de ce travail, je tenais à dire un mot sur les programmes et langages de programmation utilisés lors de cette étude. Le logiciel est entièrement basé sur des interactions entre C et Tcl/Tk. Les algorithmes sont programmés en C, et la gestion des images, des fenêtres, et de l'environnement interactif machine-utilisateur est écrite en Tcl/Tk. Une interface de base m'a été fournie par Johan Montagnat (en première année de thèse au sein du projet Épidaure). Cette interface (comme on le verra lors du descriptif) permet de visualiser des **Inrimages**, format d'images tridimentionnelles propre à l'INRIA, selon trois coupes, dans les plans verticaux, horizontaux et transversaux. Si on se limite, comme je l'ai fait lors de ce stage, à des images bidimensionnelles, une seule des coupes est utilisée.
A partir de cette interface de base, j'ai pu rajouter des fonctionnalités par l'intermédiaire de boutons ou de *slider*, au fur et à mesure de mes besoins.

Voici tout d'abord la première fenêtre qui va permettre de choisir l'image sur laquelle on veut travailler, mais qui possède aussi un éditeur de texte et un compilateur pour les commandes Tcl/Tk (figure 4.1).

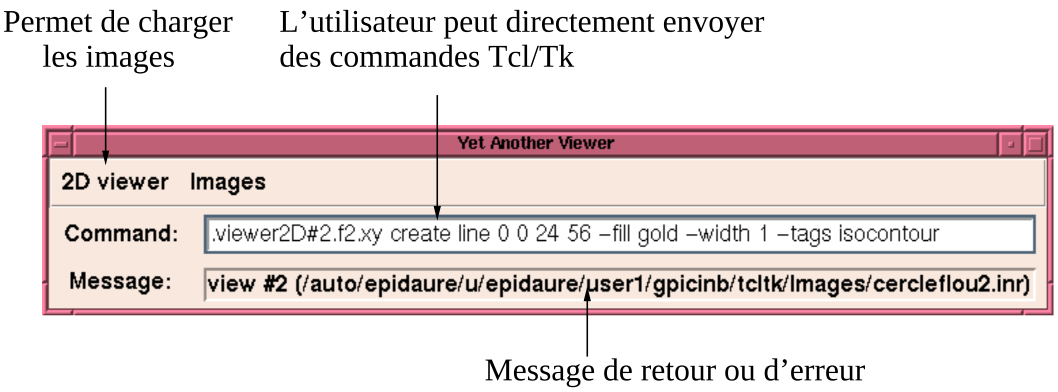


FIG. 4.1 – *La fenêtre YAV*

Et voici la fenêtre contenant l'image et toutes les possibilités d'intéraction avec le modèle (figure 4.2).

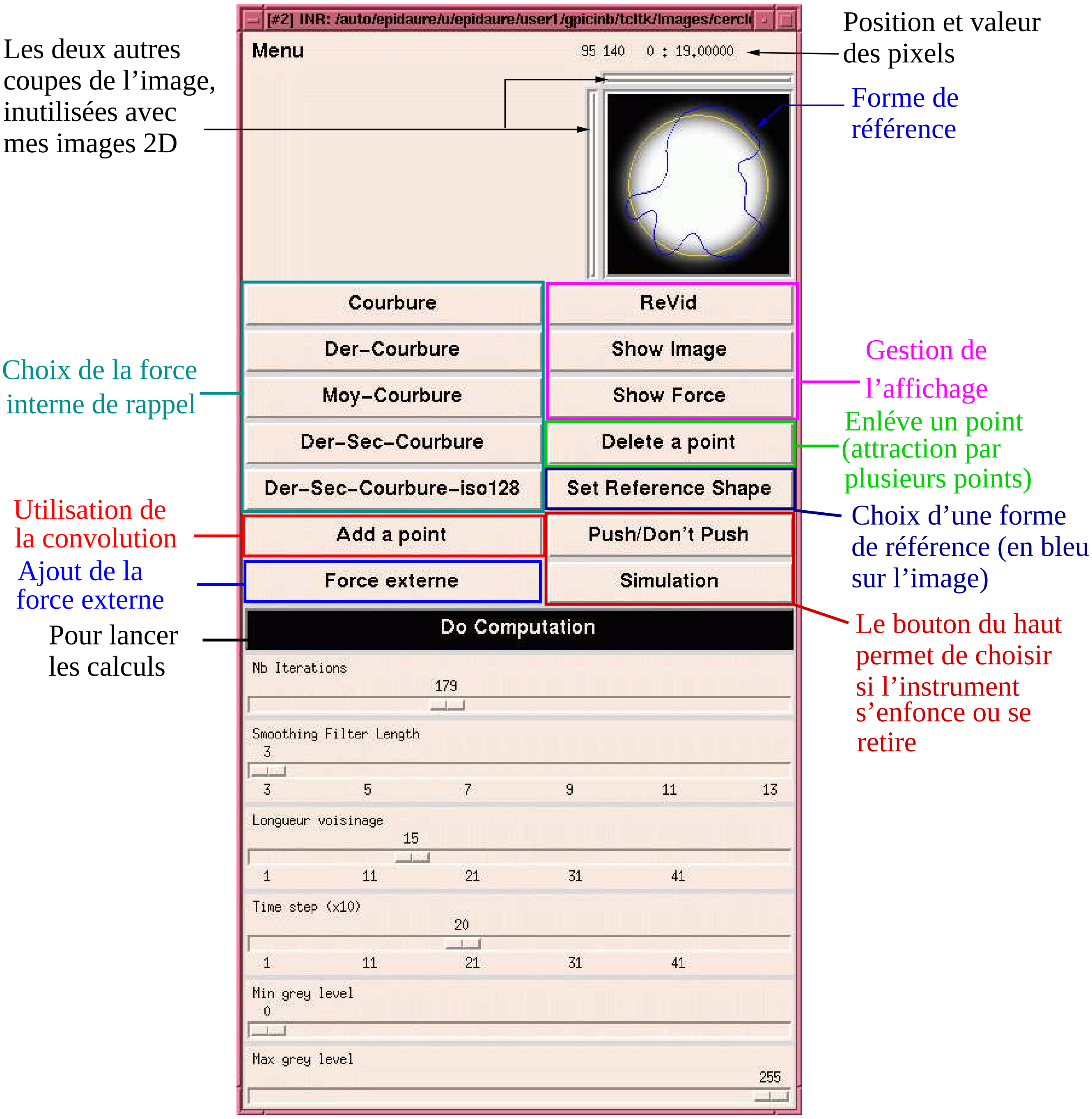


FIG. 4.2 – *L'interface utilisateur*

Conclusion

Nous voici au moment de faire le bilan de ces cinq mois de travaux. Le modèle issu de ces recherches possède des caractéristiques importantes qui m'ont guidé tout au long de son élaboration.
La première est bien sûr le choix d'une représentation implicite, dont la propriété la plus intéressante est d'être totalement insensible aux changements de topologie.
La seconde caractéristique importante est l'abscence de squelette ou de tout autre élément de construction qui pourrait à un moment ou un autre limiter les possibilités de déformation et surtout de fission du modèle en plusieurs composantes.
Ensuite, il fallait une force interne qui fasse tendre toute forme initiale vers une position d'équilibre. Ceci excluait totalement les forces de dilatation ou contraction permanente utilisées par Sethian, qui ne sont pas du tout adaptées à la simulation de déformation.
Enfin, en l'état actuel des choses, nous avons à disposition des forces externes permettant de couvrir nombre d'applications, depuis la segmentation jusqu'à la simulation.

Mais ce modèle mérite d'être approffondi et développé.
Avant toute chose, il faudra mettre en œuvre la version définitive de la force interne, c'est à dire trouver un moyen efficace de déterminer, pour chaque point de l'image, son plus proche voisin sur l'isocontour 128. Le premier cap à franchir sera celui de la troisième dimension, rendue nécessaire par les nouvelles techniques d'acquisition d'images médicales volumiques, mais surtout de plus en plus accessible grâce à la puissance des ordinateurs. Or toutes les fondations et les developpements théoriques présentés, ainsi que les algorithmes programmés sont généralisables en dimension trois. Ceci montre en particulier l'avantage décisif des modèles implicites sur leurs homologues paramétriques: par exemple le passage d'un *Snake* à un "*Snake 3D*" nécessite beaucoup plus que le simple ajout d'une dimension à tous les tableaux.
Ensuite, je pense qu'il faudra introduire dans ce modèle une notion de comportement dynamique global, qui permettra de passer d'une modélisation "de surface" (avec effet de peau), vers une vrai structure volumique (je pense à cela en particulier pour la simulation chirurgicale). De nombreuses solutions à ce problème sont proposées dans la littérature, mais il me semble que le sujet reste très ouvert.

Bibliographie

[Blo94] J. Bloomenthal. *Graphics Gems IV*, chapter IV.8: An Implicit Surface Polygonizer, pages 324–349. Academic Press, 1994.

[BS91] Jules Bloomenthal and Ken Shoemake. Convolution Surfaces. *Computer Graphics*, 25(4):251–256, July 1991.

[BW90] Jules Bloomenthal and Brian Wyvill. Interactive Techniques for Implicit Modeling. *Computer Graphics*, 24(2):109–116, March 1990.

[D8̈8] M. J. Düurst. Additionnal references to marching cubes (letter). *Computer Graphics: a Quartely Report of SIGGRAPH-ACM*, 22:72–73, 1988.

[DF95] R. Deriche and O. Faugeras. Les EDP en Traitement des images et Vision par Ordinateur. Rapport de recherche 2697, INRIA Sophia-Antipolis, November 1995.

[DG96] Mathieu Desbrun and Marie-Paule Gascuel. Smoothed Particles: A new Paradigm for Animating Highly Deformable Bodies. *Computer Animation and Simulation'96*, September 1996.

[Gas93] Marie-Paule Gascuel. An Implicit Formulation for Precise Contact Modeling between Flexible Solids. *Computer Graphics Proceeding*, pages 313–320, August 1993.

[GH91] Tinsley A. Galyean and John F. Hughes. Sculting: An Interactive Volumetric Modeling Technique. *Computer Graphics*, 25(4):267–274, July 1991.

[Gra87] M. Grayson. The Heat Equation Shrinks Embedded Plane Curves to Round Points. *Journal of Differential Geometry*, 26:285, 1987.

[KB89] Devendra Kalra and Alan H. Barr. Guaranteed Ray Intersections with I,plicit Surfaces. *Computer Graphics*, 23(3):297–306, July 1989.

[KMA96] Karl Krissian, Grégoire Malandin, and Nicolas Ayache. Directionnal Anisotropique Diffusion Applied to Segmentation of Vessels in 3D Images. Rapport de recherche 3064, INRIA Sophia-Antipolis, December 1996.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface reconstruction algorithm. *Computer Graphics*, 21(4), July 1987.

[LHdFVen] Demetri Terzopoulos Luiz H. de Figueiredo, Jonas de Miranda Gomes and Luiz Velho. Physically-Based Methods for Polygonization of Implicit Surfaces. *I don't know where*, I don't know when.

[MD95] Marie-Paule Gascuel Mathieu Desbrun. Animating Soft Substances with Implicit Surfaces. *Computer Graphics Proceedings, Annual Conference Series*, pages 287–290, August 1995.

[Mon92] J. J. Monaghan. Smoothed Particles Hydrodynamics. *Annu. Rev. Astronom. Astrophys.*, 1992.

[Mur91] Shigeru Muraki. Volumetric Shape Description of Range Data using "Blobby Model". *Computer Graphics*, 25(4):227–235, July 1991.

[OMD94] Richard Lengagne Olivier Monga and Rachid Deriche. Crest lines extraction in volume 3d medical images : a multi-scale approch. Rapport de recherche 2338, I.N.R.I.A., July 1994.

[PM90] P. Perona and J. Malik. Scale-Space and edge detection using anisotropique diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, July 1990.

[RMV95] James A. Sethian Ravikanth Malladi and Baba C. Vermuri. Shape Modeling with Front Propagation: A Level Set Approach. *IEEE Trans. on Patern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[Set96] J. A. Sethian. *Level Set Methods*. Cambridge Monograph on Applied and Computational Mathematics, 1996.

[TG93] Jean-Philippe Thirion and Alexis Gourdon. The Marching Lines Algorithm: new results and proofs. Rapport de recherche 1881-1,2, I.N.R.I.A., April 1993.

[TG95] Nicolas Tsingos and Marie-Paule Gascuel. Un Modeleur Interactif d'Objets Définis par des Surfaces Implicites. *Revue Internationale de CFAO et d'Informatique Graphique*, 10(4), 1995.

[Val95] Patrice Vallmajo. Utilisation d'images couleur pour la réalisation d'un simulateur de chirurgie du rocher de l'oreille interne. Rapport de stage de d.e.a., I.N.R.I.A. Sophia-Antipolis, projet Epidaure, 1995.

[Velen] Luiz Velho. Simple and Efficient Polygonisation of Implicit Surfaces. *I don't know where*, I don't know when.

Annexe A

Calcul de la courbure

Par définition la courbure de la ligne de niveau passant par un point d'une image est donnée par la formule:

$$\mathcal{K} = \nabla . \frac{\nabla \Phi}{|\nabla \Phi|} \tag{A.1}$$

Or $\frac{\nabla \Phi}{|\nabla \Phi|}$ n'est autre que $\vec{n}$, le vecteur normal à l'isocontour. Donc

$$\begin{aligned} \mathcal{K} &= \nabla . \vec{n} \\ &= \left| \begin{array}{c} \frac{d}{dx} \\ \frac{d}{dy} \end{array} \right. \cdot \left| \begin{array}{c} \frac{\Phi_x}{\sqrt{\Phi_x^2+\Phi_y^2}} \\ \frac{\Phi_y}{\sqrt{\Phi_x^2+\Phi_y^2}} \end{array} \right. \end{aligned} \tag{A.2}$$

Or

$$\begin{aligned} \frac{d}{dx}\left(\frac{\Phi_x}{\sqrt{\Phi_x^2+\Phi_y^2}}\right) &= \frac{\sqrt{\Phi_x^2+\Phi_y^2}\,\Phi_{xx} - \Phi_x \frac{\Phi_x\Phi_{xx}+\Phi_y\Phi_{xy}}{\sqrt{\Phi_x^2+\Phi_y^2}}}{\Phi_x^2+\Phi_y^2} \\ &= \frac{\left(\Phi_x^2+\Phi_y^2\right)\Phi_{xx} - \Phi_x\left(\Phi_x\Phi_{xx}+\Phi_y\Phi_{xy}\right)}{\left(\Phi_x^2+\Phi_y^2\right)^{3/2}} \\ &= \frac{\Phi_{xx}\Phi_y^2 - \Phi_x\Phi_y\Phi_{xy}}{\left(\Phi_x^2+\Phi_y^2\right)^{3/2}} \end{aligned} \tag{A.3}$$

On calcule de même:

$$\frac{d}{dy}\left(\frac{\Phi_y}{\sqrt{\Phi_x^2+\Phi_y^2}}\right) = \frac{\Phi_{yy}\Phi_x^2 - \Phi_x\Phi_y\Phi_{xy}}{\left(\Phi_x^2+\Phi_y^2\right)^{3/2}} \tag{A.4}$$

D'où en additionnant les deux, on obtient la formule:

$$\mathcal{K} = \frac{\Phi_{xx}\Phi_y^2 - 2\Phi_x\Phi_y\Phi_{xy} + \Phi_{yy}\Phi_x^2}{(\Phi_x^2+\Phi_y^2)^{3/2}} \tag{A.5}$$

Annexe B

Calcul des masques de dérivées partielles

Le calcul des dérivées partielles d'une fonction par la méthode des éléments finis revient à filtrer l'image par des masques. La détermination de ces masques se fait de la maniére suivante:
Soit f une fonction de $\mathbb{R}^2$ dans $\mathbb{R}$, notons f_x la dérivée partielle de f par rapport à x, en appliquant un dévellopement de Taylor de $f(x+h,y)$ et de $f(x-h,y)$ à l'ordre un, on obtient:

$$\begin{cases} f(x+h,y) = f(x,y) + hf_x(x,y) + o(h^2) \\ f(x-h,y) = f(x,y) - hf_x(x,y) + o(h^2) \end{cases} \tag{B.1}$$

En soustrayant ces deux équations on trouve que

$$f(x+h,y) - f(x-h,y) = 2hf_x(x,y) \implies f_x(x,y) = \frac{f(x+h,y) - f(x-h,y)}{2h} \tag{B.2}$$

D'où en prenant $h = 1$, on a le masque $f_x = \frac{1}{2}\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$ Le raisonnement est exactement le même pour f_y. Et f_{xy} n'est autre que la composition des deux:

$$f_{xy} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \tag{B.3}$$

En poussant le développement de Taylor à l'ordre deux, on fait apparaître la dérivée seconde:

$$\begin{cases} f(x+h,y) = f(x,y) + hf_x(x,y) + \frac{h^2}{2}f_{xx}(x,y) + o(h^3) \\ f(x-h,y) = f(x,y) - hf_x(x,y) + \frac{h^2}{2}f_{xx}(x,y) + o(h^2) \end{cases} \tag{B.4}$$

Et cette fois en additionnant les deux lignes, il vient:

$$\begin{aligned} & f(x+h,y) + f(x-h,y) = 2f(x,y) + h^2 f_{xx}(x,y) \\ \implies & f_{xx}(x,y) = \frac{f(x+h,y) - 2f(x,y) + f(x-h,y)}{h^2} \end{aligned} \tag{B.5}$$

D'où on tire le masque: $f_{xx} = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}$. Le calcul de f_{yy}, ainsi que de la dérivée croisée f_{xy}, se font de la même manière.

Enfin, en passant à des développement à l'ordre trois de $f(x+h,y)$, $f(x+2h,y)$, $f(x-h,y)$ et $f(x-2h,y)$, on trouve:

$$\left\{\begin{array}{l} f(x+2h,y) = f(x,y) + 2hf_x(x,y) + \frac{4h^2}{2}f_{xx}(x,y) + \frac{8h^3}{6}f_{xxx}(x,y) + o(h^4) \\ f(x+h,y) = f(x,y) + hf_x(x,y) + \frac{h^2}{2}f_{xx}(x,y) + \frac{h^3}{6}f_{xxx}(x,y) + o(h^4) \\ f(x-h,y) = f(x,y) - hf_x(x,y) + \frac{h^2}{2}f_{xx}(x,y) - \frac{h^3}{6}f_{xxx}(x,y) + o(h^4) \\ f(x-2h,y) = f(x,y) - 2hf_x(x,y) + \frac{4h^2}{2}f_{xx}(x,y) - \frac{8h^3}{6}f_{xxx}(x,y) + o(h^4) \end{array}\right. \tag{B.6}$$

En faisant la première ligne moins deux fois la deuxième plus deux fois la troisième moins la quatrième, on trouve:

$$\begin{aligned} & f(x+2h,y) - 2f(x+h,y) + 2f(x+h,y) - f(x-h,y) = 2h^3 f_{xxx}(x,y) \\ \Longrightarrow \quad & f_{xxx}(x,y) = \frac{f(x+2h,y) - 2f(x+h,y) + 2f(x+h,y) - f(x-h,y)}{2h^3} \end{aligned} \tag{B.7}$$

Ce qui donne le masque $f_{xxx} = \begin{bmatrix} 1 & -2 & 0 & 2 & -1 \end{bmatrix}$.

La composition de ce masque avec ceux calculés précedemment donne tous les outils nécessaires pour le calcul de la courbure et de sa dérivée.

Table des figures