Université François Rabelais
Institut Universitaire de Technologie de Tours

Département Génie Électrique et Informatique Industrielle

RAPPORT DE STAGE

Laboratoire d'informatique de TOURS – Département RFAI

Amélioration des paramètres des contours actifs par une technique issue de la théorie de l'apprentissage.

MICHOT Julien
Promotion 2002-2004

ROUSSELLE Jean-Jacques
NEBEL Fabien

Université François Rabelais
Institut Universitaire de Technologie de Tours

Département Génie Électrique et Informatique Industrielle

RAPPORT DE STAGE

Laboratoire d'informatique de TOURS – Département RFAI

Amélioration des paramètres des contours actifs par une technique issue de la théorie de l'apprentissage.

MICHOT Julien
Promotion 2002-2004

ROUSSELLE Jean-Jacques
NEBEL Fabien

Remerciements

Je tiens à remercier les Pr. C. Proust et H.Cardot de m'avoir accueilli au Laboratoire d'Informatique de Tours, au département Reconnaissance de formes et analyse d'image.

Je remercie aussi M.ROUSSELLE, M. BONE et M. NEBEL qui m'ont suivi tout au long de ce stage. Leurs conseils et indications m'ont permis d'avancer et d'améliorer l'efficacité de l'algorithme présenté dans ce rapport.

De plus, je tiens à remercier Fabrice RUBINO, élève de troisième année à l'EPU-DI, pour ses explications précieuses sur la programmation avec la librairie FOX-toolkit et sur le contour actif.

Enfin, je remercie Florian AGEN, étudiant à l'IUT GEII de Tours, de m'avoir aidé et soutenu tout au long de ce stage.

Sommaire

Introduction

Le stage de fin de deuxième année de l'IUT[1] s'est déroulé au Laboratoire d'Informatique de Tours, qui se situe au sein du département Informatique de l'École Polytechnique de Tours (EPU-DI[2]).

L'objectif de la mission était simple : étudier le comportement d'un contour actif doté d'une nouvelle technique issue de la théorie de l'apprentissage. En effet, un contour actif est encore de nos jours difficile à utiliser pour la détection de contour. Celui-ci requière l'ajustement de plusieurs paramètres suivant les images à traitées et ne peut donc être facilement exploité par des personnes inexpérimentées. Dès lors, le contour actif autonome, utilisant des paramètres aléatoires, fut inventé pour palier ce problème. Mais ce système, seul, reste totalement imprévisible et instable à chaque tentative. Aussi, plusieurs techniques sont aujourd'hui expérimentées pour améliorer la détection des contours d'une image, et on pourra citer notamment le « Line Search », le « Momentum » ou l' « Adaptation des coefficients suivant leur influence », le sujet du stage.

L'enjeu sera donc de déterminer si la modification des coefficients d'énergie suivant leur influence dans le choix du déplacement d'un contour actif, améliore la détection des contours d'une image. L'étude sera réalisée dans une première phase sur le contour actif à coefficients dits constants, et dans une autre phase, sur un snake[3] autonome.

Les différents résultats obtenus seront utilisés en vue d'une publication scientifique.

Nous allons voir dans un premier temps l'entreprise d'accueil, avec les différents départements de recherche, puis nous présenterons les multiples outils de développement utilisés au cours du stage.

Nous étudierons ensuite, le contour actif et l'algorithme Greedy dans une troisième partie, et l'adaptation des coefficients d'énergie dans une quatrième.

Enfin, les deux dernières parties seront consacrées à l'étude des tests du programme sur le contour actif à coefficients d'énergie statiques et sur le snake autonome.

[1] IUT : Institut Universitaire de Technologie

[2] EPU-DI : école polytechnique de l'université de Tours, département Informatique

[3] Snake : mot anglais signifiant serpent utilisé à cause du mode de déplacement du contour actif

1. Présentation de l'entreprise

1.1. Présentation

➢ Historique

Créé en 1970, le Laboratoire d'Informatique (L.I.) a évolué parallèlement au développement des enseignements d'Informatique au sein de l'Université François Rabelais de Tours. A compter de l'année universitaire 1991-1992, la gestion du L.I. a été confiée à la nouvelle Ecole d'Ingénieurs en Informatique pour l'Industrie (E3i), aujourd'hui devenu Département Informatique de L'Ecole Polytechnique Universitaire de Tours (EPU DI).

Depuis le 1er septembre 2003 le laboratoire est dirigé par le Professeur Christian Proust assisté de trois Professeurs Directeurs-Adjoints : Jean-Charles Billaut, Hubert Cardot et Denis Maurel.

➢ Structure

Son développement s'inscrit dans la filière « Mathématiques-Informatique-Physique Fondamentale » de l'Ecole Doctorale « Santé, Sciences, Technologie » de l'Université de Tours. Ses activités se développent autour d'enseignants-chercheurs qui encadrent des doctorants, des stagiaires du D.E.A.[4] d'Informatique, notamment, des élèves-ingénieurs et des stagiaires d'universités étrangères. Il comprend 74 enseignants-chercheurs et chercheurs à la rentrée 2003-2004, dont plus de 30 doctorants (hors visiteurs divers).

Les chercheurs sont répartis en trois catégories : chercheurs permanents, personnels techniques, correspondants-visiteurs-chercheurs temporaires. Grâce à une organisation par projets, un chercheur rattaché à une des équipes du laboratoire participe aux activités des autres équipes, selon les intérêts du LI et ses préoccupations du moment.

[4] Diplôme d'étude approfondie

Les ressources annuelles du LI proviennent du Ministère de la Recherche, de contrats CIFRE[5] ou assimilés, de contrats ponctuels (CNRS[6], ANVAR[7], RNTL[8], actions incitatives du MR, accord programme, collectivités territoriales : Région Centre, Départements de l'Indre, de l'Indre et Loire, …), de contrats de valorisation avec des entreprises, de subventions diverses (Université de Tours, collectivités territoriales …). Le LI est hébergé principalement dans les locaux de l'EPU-DI et de l'IUP de Blois.

1.2.Coordonnées et localisation

L'adresse de l'équipe au laboratoire où j'ai réalisé ce stage est la suivante :

Laboratoire d'Informatique de Tours,

Equipe reconnaissance des formes et analyses d'image 64 avenue Jean Portalis – 37200 TOURS.

La localisation est donnée sur la figure suivante :

[5] Conventions industrielles de formation par la recherche

[6] Centre national de la recherche scientifique

[7] Agence nationale de valorisation de la recherche

[8] Réseau national de recherche et d'innovation en technologies logicielles

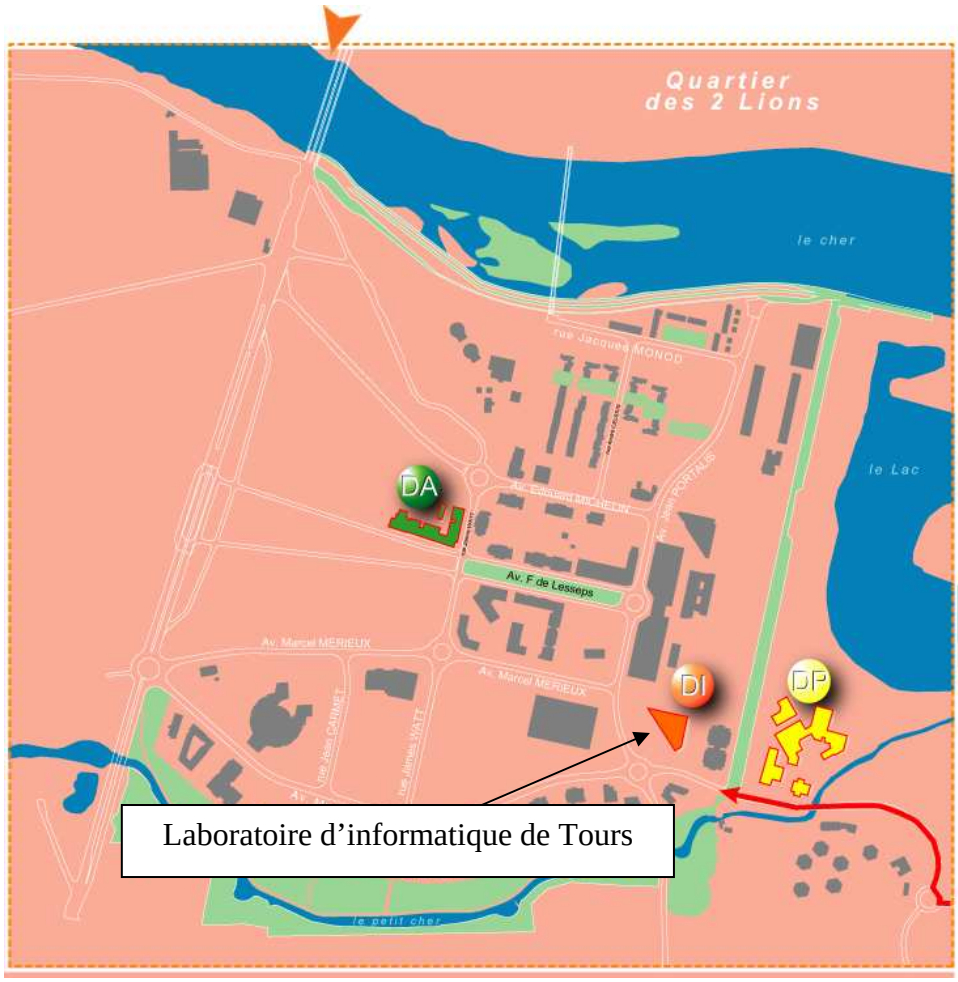


Figure 1. Localisation du laboratoire

1.3. Les Départements de recherche

Le laboratoire d'informatique de Tours est structuré en cinq équipes :

- **« Ordonnancement (systèmes informatiques, systèmes de production…) et Conduite des systèmes de production »**, sous la responsabilité de : Pr. J.-C. Billaut, Pr. P. Martineau

- **« Reconnaissance des Formes et Analyse d'Images »**, sous la responsabilité de : Pr. H. Cardot

- **« Bases de données et Traitement du Langage Naturel »**, sous la responsabilité de : Pr. D. Maurel.

- **« Réseaux (charges, sécurité) et TIC[9] (web) »**, sous la responsabilité de : Pr. G. Venturini.

[9] Technologies de l'information et de la communication

- « **Handicap et Nouvelles Technologies»**, sous la responsabilité de : P. Gaucher, Pr. M. Slimane.

Le thème central des recherches effectuées au laboratoire étant : « Algorithmes et heuristiques d'optimisation et de décision ».

Voici un petit aperçu des différentes équipes et de leurs axes de recherche.

a. Base De Données Et Traitement Du Langage Naturel (BdTln)

Responsable : Denis MAUREL
Chercheurs permanents : 14
Personnels techniques : 3
Correspondants, visiteurs, chercheurs temporaires: 6

Cette équipe propose trois axes de recherches :

- découverte de connaissances dans les bases de données,
- Informatique et linguistique.

b. Reconnaissance des Formes et Analyses d'Images (R.F.A.I.)

Responsable : Hubert CARDOT
Chercheurs permanents : 19
Personnels techniques : 2
Correspondants, visiteurs, chercheurs temporaires: 5

Cette équipe propose trois axes de recherches :

- reconnaissance des formes
- analyse d'images
- traitement du document

c. Réseaux/Technologies de l'Information et de la Communication (R.T.I.C.)

Responsable : Gilles VENTURINI
Chercheurs permanents : 7
Personnels techniques : 2
Correspondants, visiteurs, chercheurs temporaires: 0

Cette équipe de recherche est celle qui a été crée le plus récemment. Sa création est le fait de la place importante qu'occupe à présent, les réseaux et l'implantation sur Internet, dans tous les travaux du laboratoire informatique.

Cette équipe de recherche étudie trois axes de recherches :

- Création d'un site web générique,
- réalité virtuelle,
- sécurité des réseaux (détection d'évènements anormaux).

d. Ordonnancement et conduite (O.C.)

Responsables : Jean-Charles BILLAUT, Christian PROUST
Chercheurs permanents : 19
Personnels techniques : 2
Correspondants, visiteurs, chercheurs temporaires: 4

Cette équipe étudie trois axes de recherches :

- conception de nouveaux algorithmes pour résoudre des problèmes d'ordonnancement,
- mise en oeuvre d'ordonnancements pour la conduite, identification de problèmes d'ordonnancement,
- système d'exploitation et conduite : conduite des systèmes de production et systèmes d'exploitation.

e. Handicap et Nouvelles Technologies

Responsables : Pierre GAUCHER, Mohamed SLIMANE
Chercheurs permanents : 8
Personnels techniques : 0
Correspondants, visiteurs, chercheurs temporaires: 0

Cette équipe s'occupe, entre autres, de la conception d'un prototype d'une maison « intelligente » modulaire, facilement transportable, pour handicapé moteur lourd, lieu de vie en milieu ouvert. Ce projet nécessite des alliances multiples, y compris avec des économistes. Nous essayons de le promouvoir depuis 1997. Il est supporté par le CG 37 et l'entreprise CORONA Médical notamment.

f. Habitat Mobile pour Personnes Handicapées

Responsable : Pierre GAUCHER

Chercheurs permanents : 9

Personnels techniques : 2

Correspondants, visiteurs, chercheurs temporaires: 1

Ce projet est tranversal aux équipes de recherches : (R.F.A.I), BdTln, RTIC, et nécessite des alliances multiples, notamment avec des économistes.

Cette équipe de recherche étudie quatre axes de recherches :

- aides techniques pour personnes handicapées,
- domotique pour handicapé, habitat intelligent,
- outil d'aide à la conception d'un habitat adapté.

Le stage s'est déroulé au sein de l'équipe Reconnaissance des formes et analyses d'images.

2. Présentation des différents outils de développement

2.1. Le visual C++ et la MFC[10]

La programmation de logiciels se fait de nos jours à l'aide de programmes totalement dédiés, comme Borland C++ ou Visual C++. En effet, contrairement à la programmation sous DOS, la programmation sous Windows nécessite beaucoup plus de paramètres pour fonctionner correctement, paramètres automatiquement gérés pas ces derniers compilateurs. La MFC utilise le concept de *document/vue* et de *classe* pour nous offrir rapidement des objets tels que des boutons, des zones de textes etc.. Ainsi, il nous devient tout à fait possible des réaliser des programmes en *déposant* nos composants. La MFC gère aussi d'innombrables éléments comme des icônes, des fenêtres, ou des images. Voici un aperçu de l'espace de travail de visual.

[10] MFC : Microsoft Foundation Classes

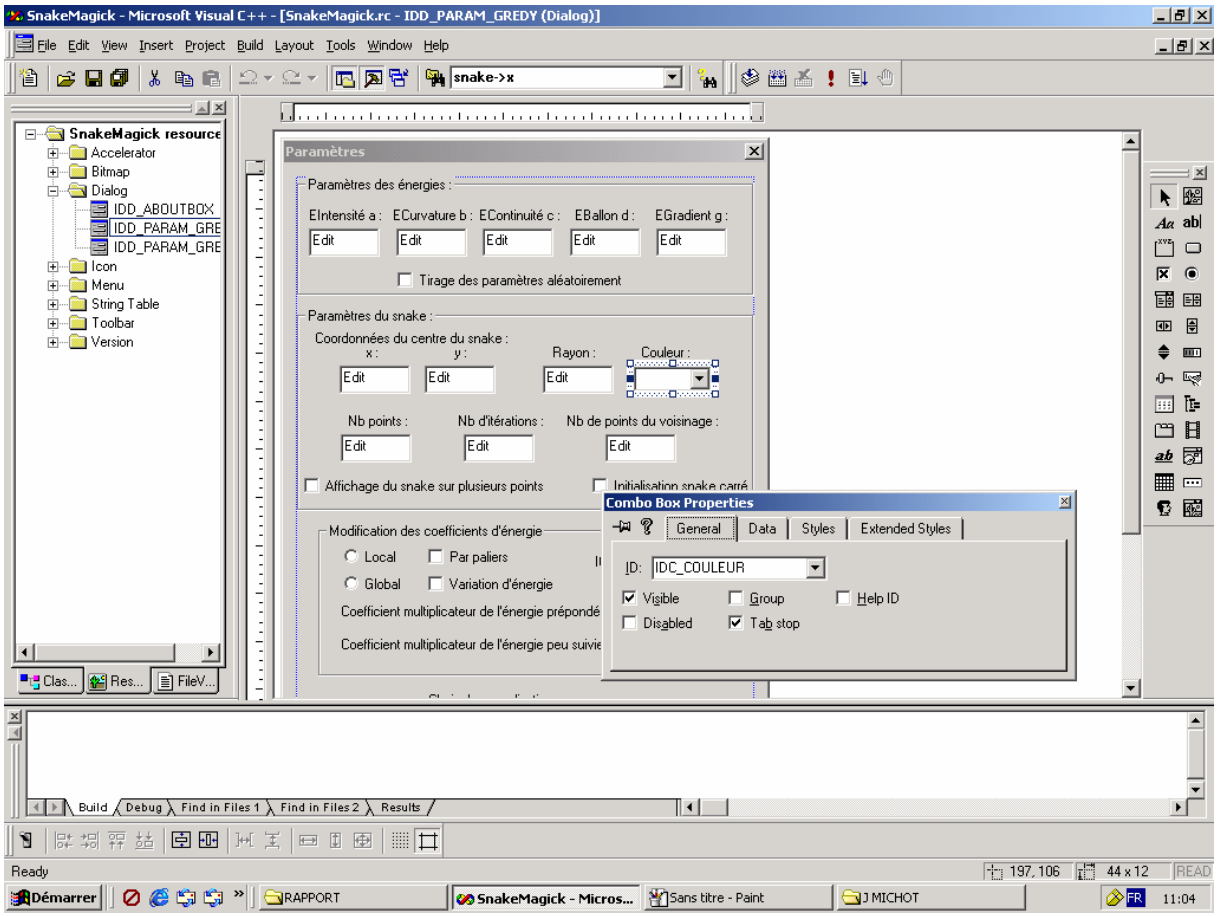


Figure 2. La MFC sous Visual C++ 6.0

Ce compilateur ressemble beaucoup au compilateur manipulé à l'IUT, Borland C++. Ils est néanmoins beaucoup moins simple d'utilisation.

Toutes les parties programmations du stage ont été effectuées en général en langage C++. Le langage Visual Basic a été aussi manipulé lorsque nous avons réalisé des Macros[11] sous Excel.

2.2. La librairie FOX

La bibliothèque *Fox Toolkits*, basée sur une conception C++ et une philosophie Java, permet de réaliser très simplement une interface graphique (GUI[12]). Les créateurs de la librairie fournissent plusieurs exemples de programmes réalisés entièrement avec FOX, ce qui nous permet de mieux comprendre comment manipuler la syntaxe.

[11] Macro : macro-commande qui permet d'exécuter automatiquement des actions

[12] Graphical User Interfaces

L'installation de cette librairie reste simple, il suffit d'inclure le dossier « fox-1.0.46/include », contenant tous les fichiers d'entêtes « .h » lors de la compilation. Ces fichiers sont très intéressants car ils contiennent toutes les classes, et tous les prototypes des fonctions et des méthodes existants dans la librairie. Le fichier principal de la librairie FOX « fox.lib » (ou « foxd.lib » pour le mode debug) est aussi à inclure dans le projet, celui-ci contenant justement toutes les sources des méthodes mentionnées dans les fichiers d'entêtes.

La librairie permet de manipuler facilement aussi bien les flux de fichiers, que les images, ou les différents boutons/menus/objets des fenêtres de Windows, etc… (cf. figure 3) De plus, les concepteurs de la FOX ont eut l'ingénieuse idée d'insérer un « FX » au début de chaque classe appartenant à la FOX. Cela rend l'utilisation bien plus confortable.

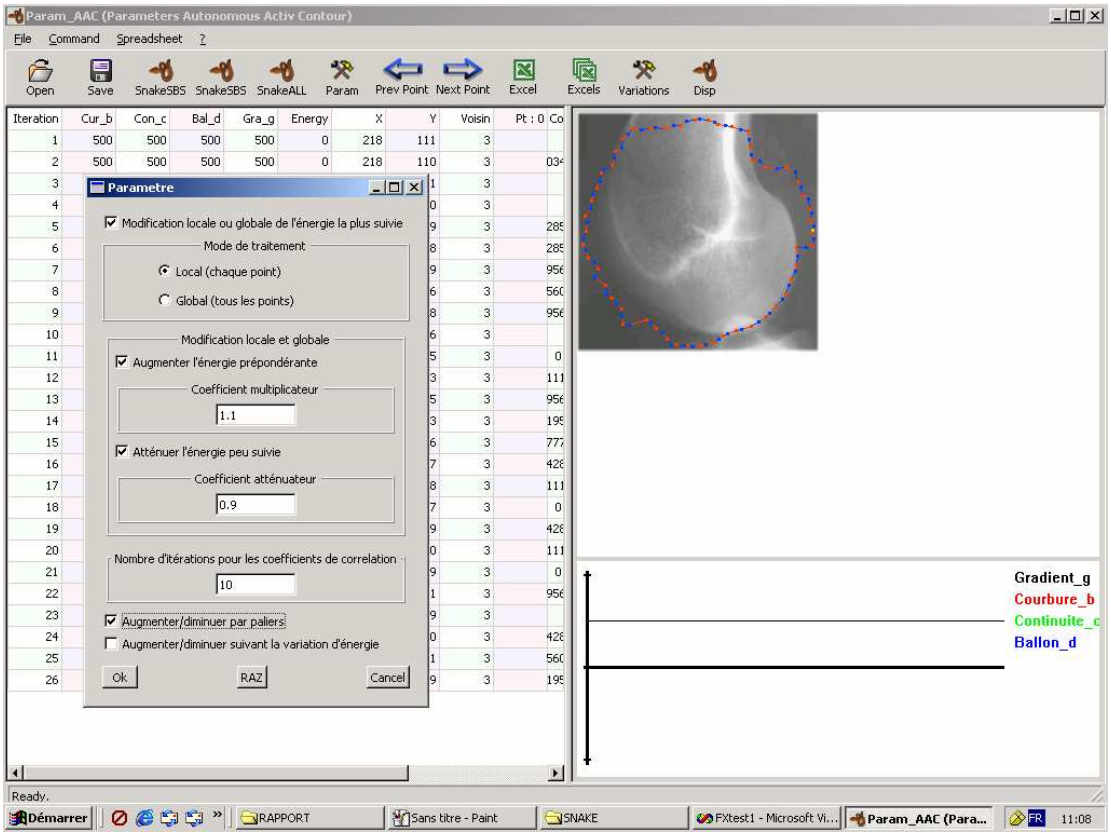


Figure 3. Le programme initialement modifié (Param_AAC[13]), utilisant la FOX

Cette librairie est aussi très portable puisque nous pouvons l'utiliser dans des systèmes d'exploitation tels que Windows ou Linux. L'application utilisant la FOX devra cependant être en Win32 (et non AppWizard) afin de pas rencontrer de conflits logiciels.

[13] Param_AAC : Parameters Activ Autonomous Contour

La bibliothèque *Fox ToolKits* est appropriée au traitement simple des images 2D/3D et peut inclure des bibliothèques comme OpenGL[14] pour être encore plus complète. Nous avons utilisé lors du stage la version 1.0.51 de la librairie.

.

Param_AAC fut le premier programme utilisé pour, non seulement étudier le contour actif lui même, mais aussi pour coder les différentes améliorations nécessaires à l'élaboration du sujet de stage. Mais en raison d'une multitude d'erreurs, de défauts et d'approximations, nous avons été contraint de reporter notre travail sur un autre logiciel : SnakeMagick.

2.3. La librairie ImageMagick

Le programme utilisé en second lieu (SnakeMagick) est celui que nous avons retenu pour tester notre algorithme. Ce programme est basé sur la plate-forme MFC et utilise la librairie ImageMagick. Pour utiliser ce programme, il faut installer au préalable la bibliothèque ImageMagick sur chaque poste de travail. Nous obtenons alors une vingtaine de DLL[15] directement utilisées par le programme SnakeMagick.
(Le détail de l'installation est fourni en annexe 3 p 70)

Voici un aperçu du logiciel utilisé.

[14] OpenGL : « Open Graphics Library » librairie graphique orienté 3D

[15] DLL : Dynamic Link Library

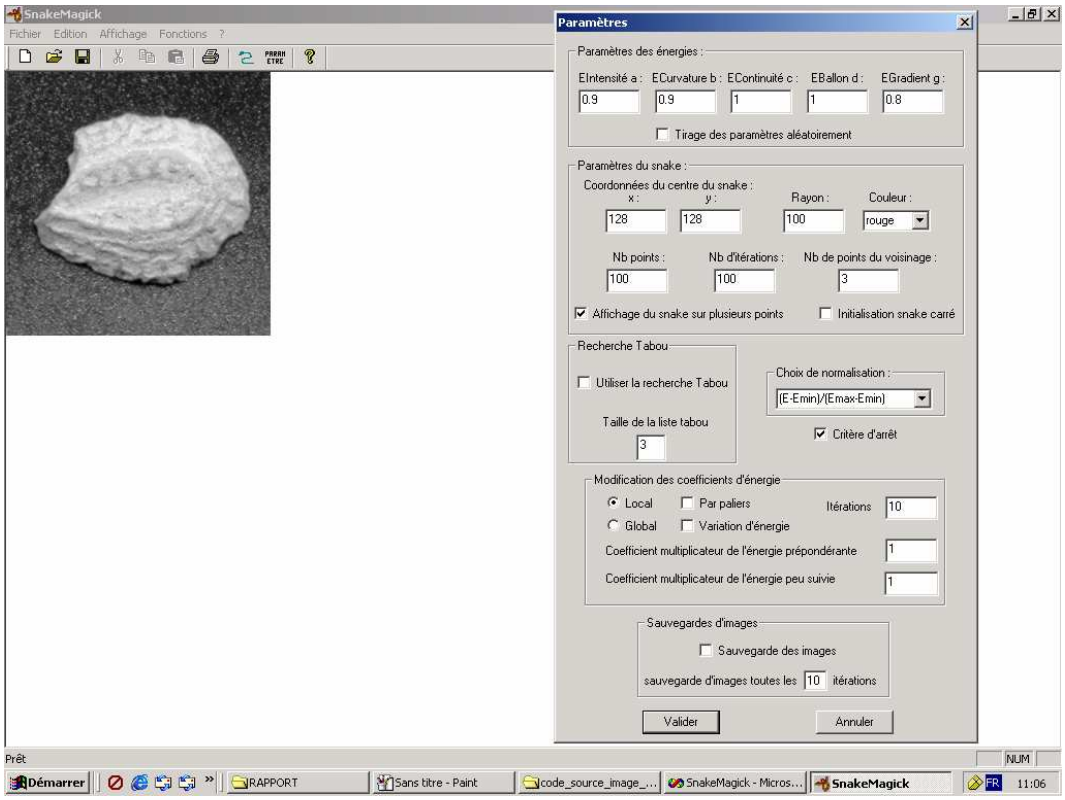


Figure 4. Programme finalement adopté, utilisant ImageMagick

Contrairement à la librairie FOX, ImageMagick ne génère aucune plate-forme, il faut donc l'utiliser avec la MFC par exemple. Cette librairie est beaucoup plus complète que la FOX, elle intègre beaucoup plus de fonctionnalité notamment sur les images. L'utilisation, tout comme l'installation, reste longue est fastidieuse, mais sa grande liberté de traitement la rend très efficace.

2.4. VIDEOMACH

Le logiciel VideoMach est un programme réalisant des vidéos dans plusieurs formats (dont : avi, mpeg, etc.…).

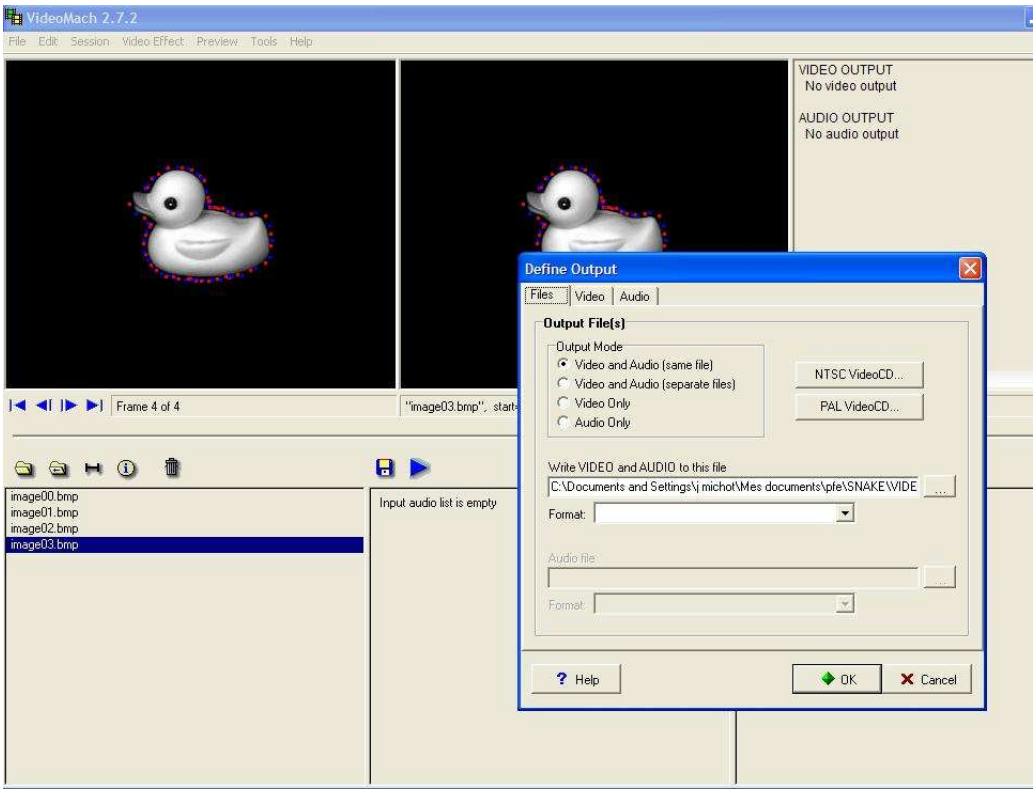


Figure 5. Le logiciel VideoMach

Ce logiciel nous a permis de créer des vidéos au format **avi** avec les différentes images du snake extraites automatiquement. La visualisation du déplacement du snake (et la sauvegarde de celle-ci) est utile pour régler les différents paramètres du snake (notamment le critère d'arrêt).

3. Etude du contour actif

3.1. Définition du snake

KASS *et al.*[16] proposèrent dès 1985, dans *Proceedings of First International Conference on Computer Vision*, une nouvelle méthode pour la détection de contours dans une image : les contours actifs. Ce contour est modélisé par une courbe qui se déforme en fonction de différentes énergies pour venir se positionner sur les frontières de l'objet. Pour les images numériques, le traitement s'effectuera avec un certain nombre de points chaînés, et dont le nombre varie en fonction de la précision désirée.

[16] *et al : latin, « et son équipe »*

Etant donné l'existence de différentes formes d'images, des coefficients sont attribués à chacune des énergies utilisées. La fonctionnelle d'énergie représentera alors l'ensemble des énergies, auxquelles sont appliqués leur coefficient respectif. L'approche variationnelle (utilisée dans notre projet) tendra à minimiser cette fonctionnelle d'énergie.

Il existe trois sortes d'énergies :

- les énergies internes
- les énergies externes
- et les énergies de contexte

Les énergies internes gèrent la cohérence de la courbe. Elles définissent la raideur de la courbe et la cohésion des points. Cette catégorie est composée de deux énergies : l'énergie de continuité (la dérivée première de la courbe), et l'énergie de courbure (deuxième dérivée).

Les énergies externes prennent en compte les caractéristiques des images traitées. Parmi celles-ci, il faut compter l'énergie d'intensité (la couleur du pixel de l'image) et l'énergie de gradient (la dérivée première de l'image cette fois-ci).

Enfin, les énergies de contexte, parfois appelées énergies de contrainte, permettent d'introduire des connaissances *a priori* sur ce que nous cherchons. Dans notre logiciel, nous utiliserons l'énergie de contexte de ballon.

Ces différentes énergies seront étudiées en détail par la suite.

Tout au long du stage, nous avons utilisé un seul et même algorithme, nommé Greedy, qui est reconnu pour être l'un des algorithmes les plus rapides pour les contours actifs.

3.2. L'Algorithme Greedy

Dans le cas de l'algorithme *greedy* utilisé, l'énergie globale (ou fonctionnelle globale d'énergie) du contour associée aux N points est :

$$E(C)=\sum_{i=1}^{N} a E_{\text{intensité}}(M_i)+c E_{\text{continuité}}(M_i)+ b E_{\text{courbure}}(M_i)+ g E_{\text{gradient}}(M_i)+d E_{\text{ballo}}(M_i))$$

avec M_i, i=1..N, un ensemble de points ordonnés.

Équation 1. Fonctionnelle d'énergie

Il s'agit de trouver l'ensemble des N points M_i qui définissent la courbe C. Pour cela, il existe plusieurs méthodes et parmi celles-ci, l'algorithme *greedy*.

a. Principe

La méthode du *greedy* consiste à faire évoluer le contour actif en minimisant la fonctionnelle d'énergie : pour chaque point du *snake*, on choisit un nombre de voisins pour lesquels 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.

C'est donc un algorithme itératif qui déplace un point unique pour constituer un nouveau contour actif à chaque itération. Tous les points sont traités successivement lors de chaque itération.

Dans l'algorithme *greedy* utilisé, la fonctionnelle d'énergie est une somme de 5 énergies (intensité, continuité, courbure, gradient et ballon) normalisées et pondérées chacune par un coefficient (a, b, c, g et d).

Dans l'algorithme *greedy* avec tirage aléatoire des paramètres (le contour actif autonome), il s'agit de déterminer, pour chaque itération et pour chaque point du *snake M*, le meilleur jeu de coefficients qui minimisent la fonctionnelle d'énergie à cette itération.

Pour déterminer le meilleur jeu de coefficients, on tire aléatoirement un nouveau jeu de paramètres, à chaque point du snake, à chaque point du voisinage et à chaque itération. Puis, en comparant la fonctionnelle d'énergie avec ces coefficients et la précédente, on garde le nouveau jeu de coefficients si celui-ci livre une énergie plus faible.

b. Les algorithmes

Pour mieux comprendre comment fonctionne le greedy, voici les deux algorithmes utilisés :

Le Greedy à coefficients d'énergie constants

```
Faire
|  Pour tous les points du snake
|  |  Pour tous les points du voisinage
|  |  |  Calculer les energies
|  |  FinPour
|  |  Pour tous les points du voisinage
|  |  |  Normalisation
|  |  FinPour
|  |  Minimiser pour obtenir le nouveau point
|  FinPour
Jusqu'au Critère d'arrêt
```

Figure 6. L'algorithme Greedy

Le contour actif autonome

```
Tirage aléatoire des paramètres AncienParm(a, b, c, d, g)
Faire
|  Pour tous les points du snake
|  |  Pour tous les points du voisinage
|  |  |  Calculer les énergies
|  |  FinPour
|  |  Pour tous les points du voisinage
|  |  |  Normalisation
|  |  FinPour
|  |  Pour tous les points du voisinage
|  |  |  Tirage aléatoire des paramètres NouvParm(a1, b1, c1, d1, g1)
|  |  |  Jusqu'à Σ des carrés des coefficients = 1 ±5 %
|  |  |  Si EnergTot avec NouvParm < EnergTot avec AncienParm
|  |  |  Alors AncienParm ←NouvParm
|  |  |  FinSi
|  |  FinPour
|  |  Minimiser EnergTot pour obtenir le nouveau point
|  FinPour
Jusqu'au critère d'arrêt
```

Figure 7. Algorithme pour le contour actif autonome

Voici la boîte de dialogue regroupant tous les paramètres (a b c d et g en haut) du snake et plus particulièrement ceux de l'algorithme Greedy.

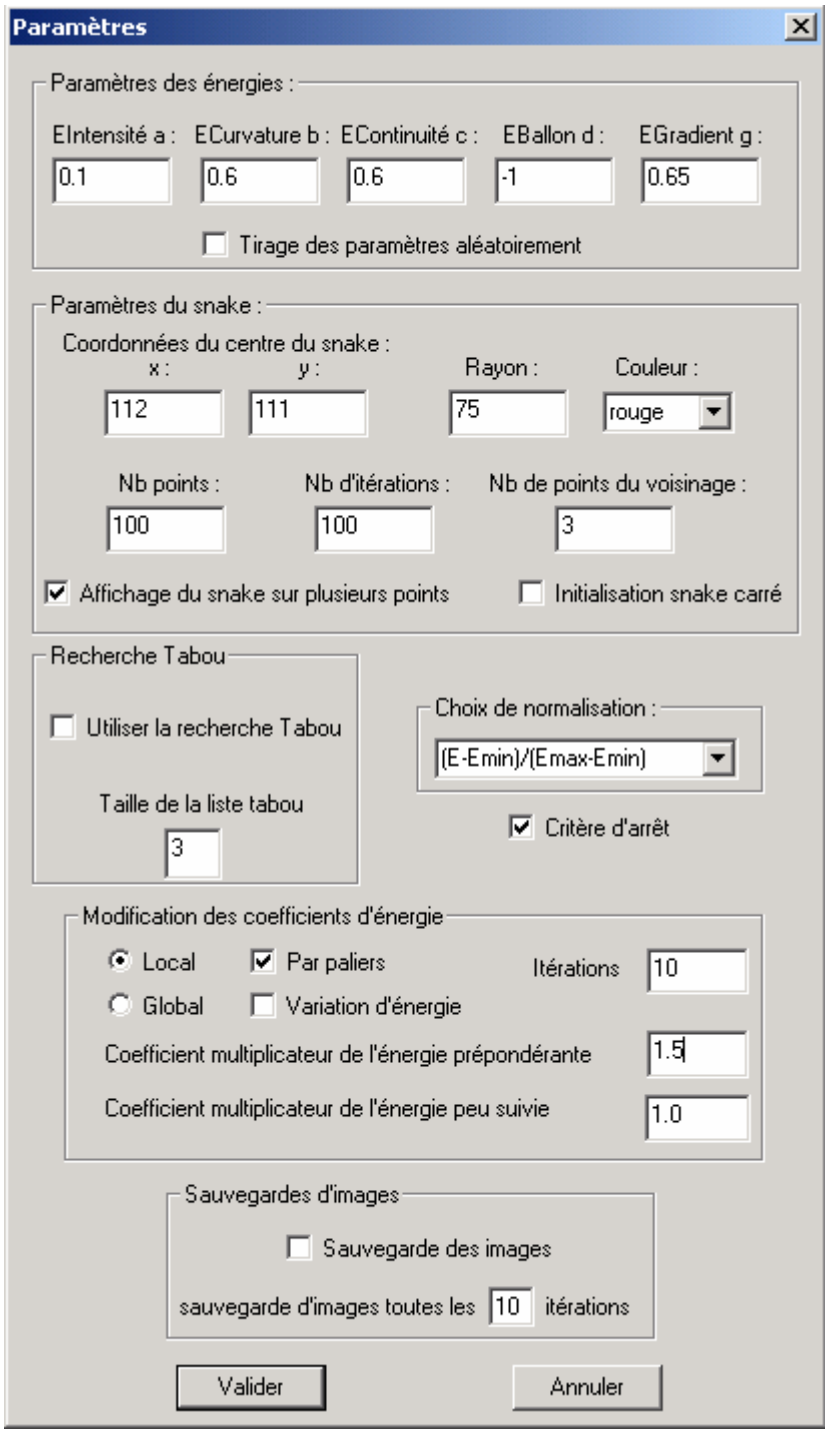


Figure 8. Boîte de dialogue regroupant les différents paramètres

Nous allons dans la suite de cette partie tenter d'expliquer quel est le rôle de chaque énergie dans la décision du déplacement du snake. Nous allons parcourir aussi les autres paramètres propres à l'algorithme Greedy.

c. L'énergie d'Intensité

L'énergie d'Intensité fait partie des énergies externes. Cette énergie représente simplement la valeur du pixel (sur un octet, donc de 0 à 255) de l'image transformée en niveaux de gris.

Eintensité = NiveauxDeGris[x][y] ;

Équation 2. Calcul de l'énergie d'Intensité

Remarque : la transformation d'une image en niveau de gris est explicitée dans la partie 3.2.f.

d. L'énergie de Courbure

L'énergie de courbure définit l'angle formé par trois points adjacents du snake. Plus le coefficient attribué à cette énergie sera fort, et plus le snake aura la forme connue pour être la forme la moins « coûteuse » en énergie : la ligne droite ou le cercle pour des objets fermés. (cf. figure 9)

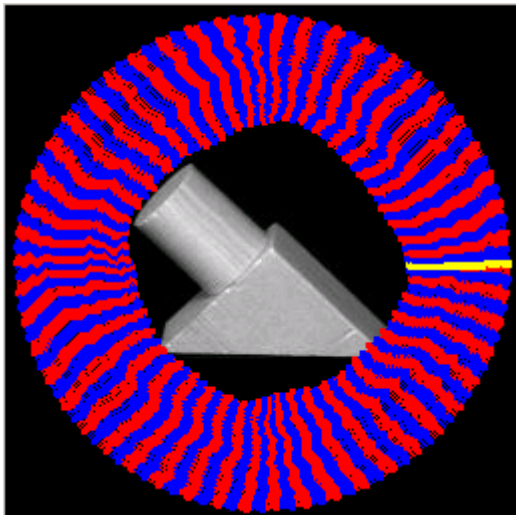


Figure 9. Test effectué avec un fort coefficient de courbure

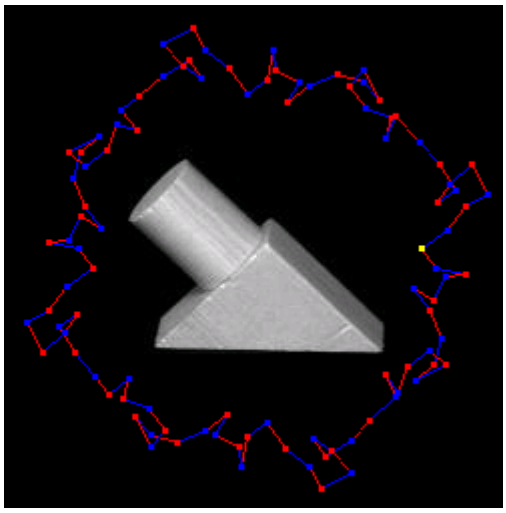
L'énergie de courbure est donnée par la relation suivante :

$$Ecourbure = 0{,}5 \times \left((Point_{i-1}.X - 2 \times Point_i.X + Point_{i+1}.X)^2 + (Point_{i-1}.Y - 2 \times Point_i.Y + Point_{i+1}.Y)^2 \right)$$

Équation 3. Calcul de l'énergie de Courbure

e. L'énergie de Continuité

L'énergie de continuité fait partie des énergies dites internes au contour actif. Cette énergie régit la distance entre les différents points du snake. Ainsi, lorsque le coefficient de l'énergie de continuité est nul, les points du snake pourront se déplacer aussi loin les uns des autres que l'image leur permettra. Dans le cas contraire, c'est-à-dire lorsque le coefficient de continuité est très élevé, le snake sera rigide. (cf. figure suivante)



Du fait de la forte valeur du coefficient de continuité, la distance entre les points du snake est fixe. Cela oblige les points du snake à se déplacer à une distance fixe de ses points adjacents. Nous observons que, entre trois points du snake, il existe des angles très obtus. (haut de l'image) Cela est principalement dû au coefficient de courbure nul, et non à l'énergie de continuité.

L'énergie de continuité se calcule par rapport à la moyenne des distances entre les points du snake, telle que :

$$\mathbf{E continuité} = 0{,}5 \times abs(\text{DistanceMoyenne} - \sqrt{(\text{Point2.X - Point1.X})^2 + (\text{Point2.Y - Point1.Y})^2})$$

Équation 4. Calcul de l'énergie de continuité

Nous remarquons que pour minimiser cette expression, le point *Point1* doit se positionner à une distance égale à la distance moyenne (*DistanceMoyenne*) du point *Point2*.

f. L'énergie de Gradient

L'énergie de Gradient est la deuxième énergie du contour actif qui dépende de l'image. Cette énergie externe est d'une importance première pour la détection du contour. En effet, un contour est généralement caractérisé par une forte différence entre les valeurs de plusieurs pixels. Ainsi, si la dérivée d'une fonction représente les pentes d'une courbe, le gradient montrera les fortes différences, les contours de l'image.

En mathématique, le gradient d'une fonction spatiale (trois dimensions au minimum) représente simplement la dérivée de celle-ci suivant toutes les directions du plan (O,i,j) :

$$\vec{grad}\left(f(x,y)\right) = \frac{\partial f}{\partial x}\vec{i} + \frac{\partial f}{\partial y}\vec{j}$$

Équation 5. Formule du gradient

En informatique, les fonctions sont discrètes, et il devient difficile voir impossible d'appliquer cette formule. Aussi, pour obtenir le gradient d'une image, donc d'une matrice de pixels, nous pouvons utiliser les méthodes de Roberts, Prewitt ou de Sobel.

Premièrement, afin d'alléger les calculs, nous réalisons le gradient sur une image en niveau de gris. Les images en couleur, possédant trois composantes de couleur : Rouge Vert et Bleu, sont transformées en images possédant une seule composante grise, représentative des

trois anciennes composantes pour ne pas détériorer trop fortement l'image et donc les contours. La valeur de cette dernière est calculée selon la relation :

$$Gris = (0{,}299*Rouge+0{,}587*Vert+0{,}114*Bleu)$$

Figure 10. Transformation d'une image en niveau de gris

La transformation en niveau de gris conserve l'information essentielle de l'image : le contours (cf. figure 10).

Nous appliquons ensuite l'algorithme de calcul du gradient de l'image.

Pour obtenir le gradient d'une image, chaque pixel, chaque case de la matrice image sera calculé en fonction de ses voisins pondérés par une matrice de coefficients déterminés. Voici un exemple de matrices de coefficients, les plus utilisées.

Matrices de Roberts

1	0
0	-1

0	1
-1	0

Matrices de Prewitt :

1	1	1
0	0	0
-1	–1	–1

1	0	-1
1	0	-1
1	0	–1

Matrices de Sobel :

1	2	1
0	0	0
-1	–2	–1

1	0	-1
2	0	-2
1	0	–1

Le gradient de Sobel de la matrice *m* à la ligne *i* et la colonne *j* sera donc :

$$Grad(m_{ij}) = \sqrt{\begin{pmatrix} (1\times m_{i-1\,j-1} + 2\times m_{i\,j-1} + 1\times m_{i+1\,j-1} - 1\times m_{i-1\,j+1} - 2\times m_{i\,j+1} - 1\times m_{i+1\,j+1})^2 \\ + \\ (1\times m_{i-1\,j-1} + 2\times m_{i-1\,j} + 1\times m_{i-1\,j+1} - 1\times m_{i+1\,j-1} - 2\times m_{i+1\,j} - 1\times m_{i+1\,j+1})^2 \end{pmatrix}}$$

Équation 6. Le gradient de sobel

Voici une comparaison entre le gradient d'une image et l'image originale :

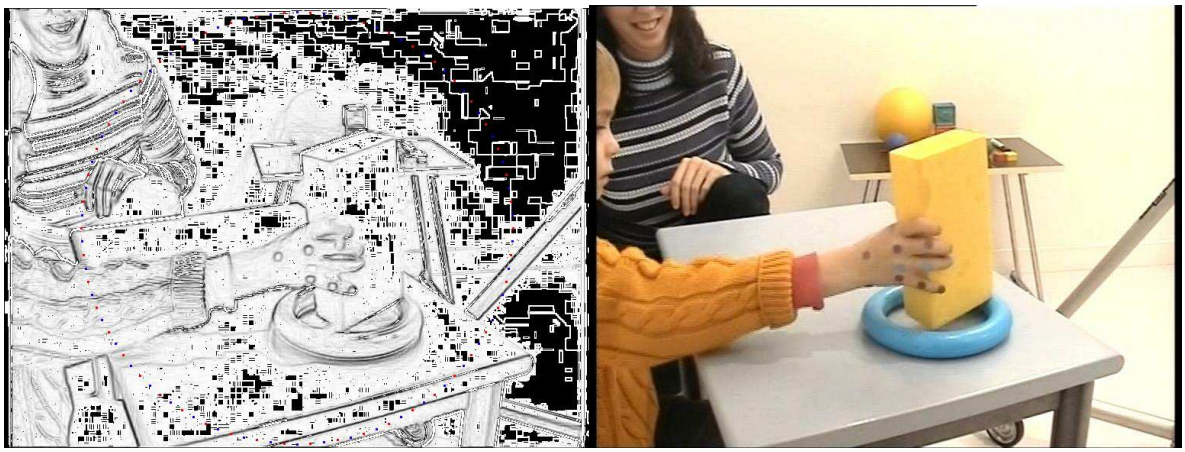


Figure 11. Le gradient d'une image et l'image originale

Nous observons que les contours sont bien « renforcés » (couleurs sombres). Si nous effectuons ensuite un seuillage de l'image du gradient, nous filtrerons alors les faibles pentes, les contours peu importants.

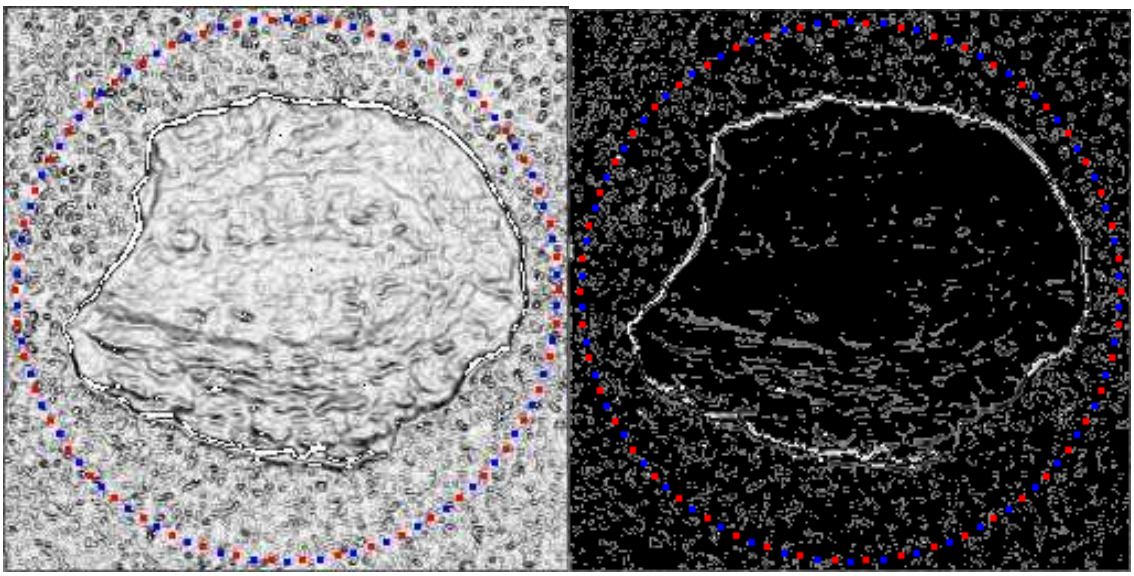


Figure 12. Gradient seul et gradient + seuillage (150)

Voici un test effectué avec un coefficient d'énergie de gradient nul :

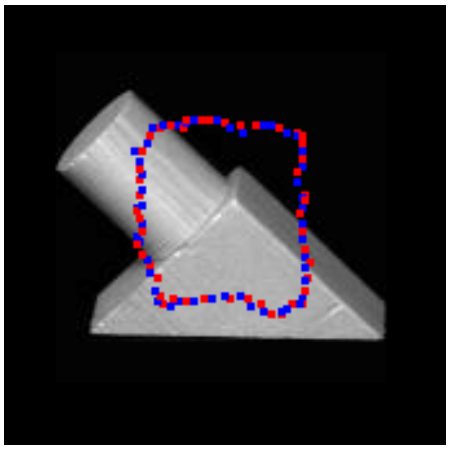
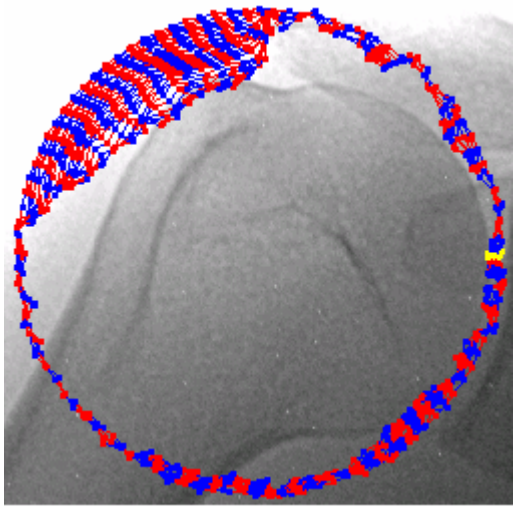


Figure 13. Coefficient d'énergie de gradient nul

Le snake n'a capté aucun contour, il ne s'arrête donc pas. Ce test montre bien que sans gradient, le contour actif ne pourrait pas connaître les emplacements des contours.

Voici un autre test effectué sur une image réelle, avec un fort coefficient d'énergie de gradient:



A l'inverse, le coefficient élevé du gradient bloque le snake. En effet, le gradient étant une dérivée spaciale de l'image, plus la différence de couleur entre deux pixels est élevée, plus le gradient sera fort, et plus le snake pensera qu'il s'agit d'un contour. Un fort gradient va donc stopper le snake sur une faible variation de couleur. Cette image étant fortement bruitée, le snake ne peut avancer.

Ce problème de bruit est caractéristique des images réelles. En effet, celles-ci possèdent des pixels généralement différents de ses voisins, ce qui donne au gradient de nombreux « piques » de couleur noire. (cf. figure 14 et 15)

Figure 14. Le "bruit" d'une image réelle

Voila pourquoi le gradient d'une image réelle ne va pas donner une très bonne indication pour les contours, contrairement aux images synthétiques qui fournissent des contours nets. (cf. figure 15)

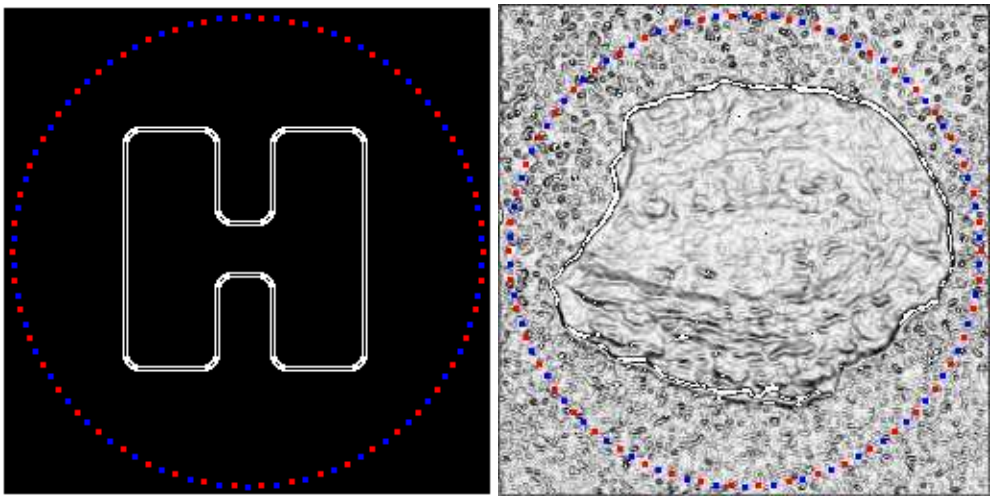


Figure 15. Gradient d'une image synthétique et d'une image réelle

g. L'énergie de Ballon

L'énergie de Ballon est l'énergie qui décide du sens de propagation du contour actif. Un coefficient d'énergie de ballon positif va concentrer le snake, alors qu'un coefficient négatif va rendre le snake expansif. (cf. figure 16)

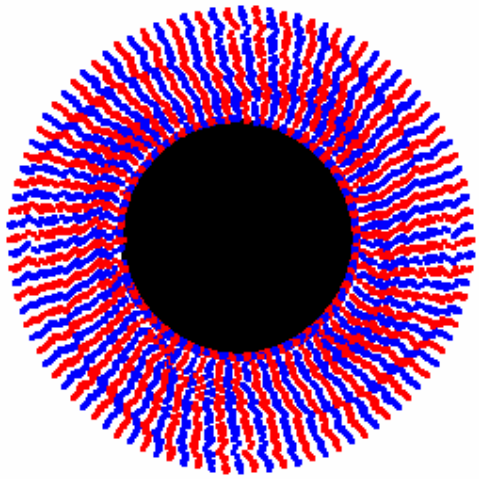
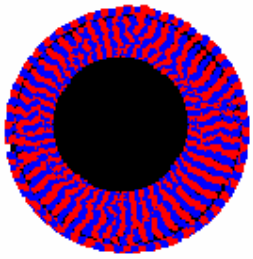
Energie de ballon positive	Energie de ballon négative
	
Le snake va se contracter	Le snake va se gonfler

Figure 16. Importance du signe de l'énergie de ballon

Pour calculer l'énergie Ballon, nous étudions la distance entre un point du snake et le barycentre des différents points composant le snake.

$$Eballon = abs(\sqrt{(Po\,\mathrm{int}.X - MoyenneX)^2 + (Po\,\mathrm{int}.Y - MoyenneY)^2}$$
$$\times\sqrt{(Po\,\mathrm{int}.X - Po\,\mathrm{int}\_avant.X)^2 + (Po\,\mathrm{int}.Y - Po\,\mathrm{int}\_avant.Y)^2}$$

Nous remarquons que pour minimiser cette énergie, le point *Point* de coordonnées *Point.X* et *Point.Y*, aura tendance à vouloir atteindre le barycentre du snake.

Mais l'algorithme Greedy apporte lui-aussi d'autres paramètres.

3.3. Parcours des différents paramètres supplémentaires réglables

a. Le mode Aléatoire

Comme nous l'avons vu précédemment, un contour actif est composé de plusieurs énergies, auxquelles nous appliquons différents coefficients pour savoir où se déplacer. Le grand handicap de cette méthode est qu'il est nécessaire de régler, d'ajuster les coefficients pour obtenir de bons résultats, et cela peut prendre parfois beaucoup de temps. Aussi, un contour actif autonome a été conçu. Celui-ci génère en effet aléatoirement les coefficients d'énergie, et garde le meilleur jeu de coefficients pour les itérations suivantes.

En théorie, au bout d'un certain nombre d'itérations, le jeu de coefficient devient optimal et la détection du contour est probable. Mais les tests que nous effectué montrent qu'il est nécessaire d'améliorer cette méthode, en ajoutant par exemple un algorithme génétique[17].

b. Le voisinage

Pour déplacer un point, l'algorithme greedy calcule toutes les énergies des pixels présents dans le voisinage du pixel où le point est.

Pour un voisinage de trois par exemple, l'algorithme va calculer les énergies d'une matrice de 7x7 pixels pour choisir son déplacement (cf. figure 17).

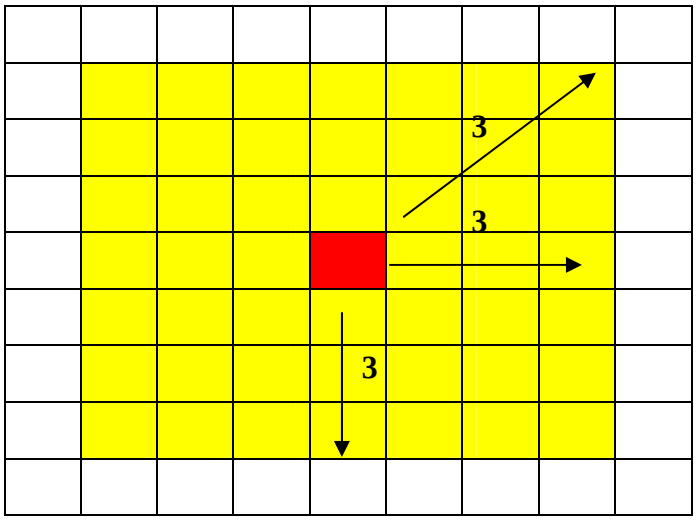


Figure 17. Exemple d'un voisinage de 3

Ce paramètre du Greedy permet au snake de se déplacer assez rapidement, en terme d'itération, car si l'on prend l'exemple du cercle (cf. figure 18), dès la première itération presque la moitié du contour est détectée. Dans cette figure, nous avons initialisé volontairement le snake (itération 0) en décalage par rapport au cercle, et cela permet de bien voir que là où les points sont les plus proche du cercle, la partie du contour est détecté dès la première itération. Cependant, ce que nous gagnons en itération, nous en perdons en temps car avec un voisinage de neuf, cela revient à calculer les énergies d'une matrice de 19x19 pixels pour chaque point. Nous restons tout de même gagnant en temps.

[17] cf. Thèse de J.J. Rousselle

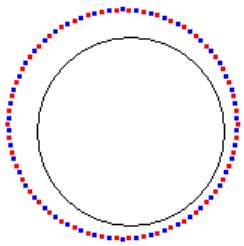
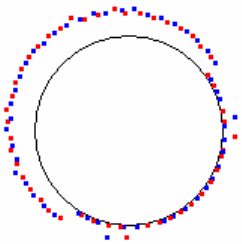
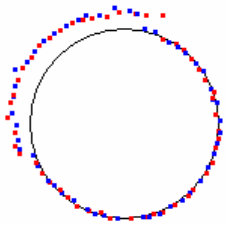
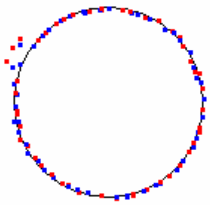
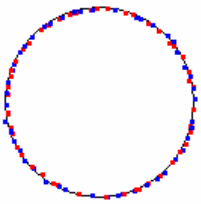
 Itération 0 (Initialisation)	 Itération 1	 Itération 7
 Itération 8	 Itération 10	Voisinage : 9 Autour de chaque point du snake, l'algorithme calcule les énergies de tous les pixels du carré de 19x19 pixels (dont le centre est le point du snake). Il choisira ensuite l'énergie minimale pour déplacer le point.

Figure 18. Evolution du snake avec un fort voisinage

Un grand voisinage peut aussi s'avérer inutile voir néfaste pour le snake car, si l'on prend l'exemple du condyle, le snake va alors « capter » les contours internes au condyle (cf. figure 19).

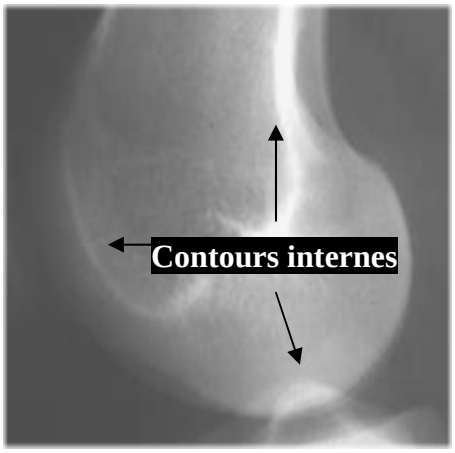


Figure 19. Contours internes suivis du condyle avec un fort voisinage

c. La normalisation

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 nous permet ainsi, d'obtenir des variations comprises entre -1 et 1, pour toutes les énergies.

d. Les critères d'arrêt

Les critères d'arrêt sont très importants, notamment lorsque l'on travaille avec des images réelles, comme le condyle par exemple. En effet, il faut parfois préciser au snake quand s'arrêter car, dans certains cas, le contour actif s'arrête pendant un moment sur les contours de l'image puis dépasse ceux-ci, attiré par des voisins dont l'énergie est plus faible.

Dans le tout premier programme, trois types de critères d'arrêt étaient programmés.
Le snake s'arrête lorsque :

- moins de *x* % des points du snake se sont déplacés entre deux itérations.
- la moyenne des déplacements sur une itération est inférieure à *x* pixel(s).
- la variation de l'énergie est inférieure à *x* % sur les six dernières itérations.

Après avoir manipuler et tester le programme, nous nous sommes rendus compte que dans des cas particuliers (avec des images réelles dont les contours sont assez « forts » comme le cauryBroken), le snake avait tendance à « osciller » entre deux positions, sur le contour de l'image. Nous étions donc devant un cas où le contour actif avait bien détecté le contour et où le programme ne s'arrêtait pas du fait de l'oscillation du snake. Le premier critère d'arrêt ne se révéla pas du tout efficace dans ce cas précis.

Aussi, nous avons pensé qui serait intéressant de modifier le premier critère d'arrêt, et de comparer les déplacements des points non pas avec la position à l'itération précédente, mais avec l'avant dernière position. Ainsi, le problème de l'oscillation du snake a été résolu. Cette dernière méthode ne change de plus pas beaucoup l'action première de l'ancien critère d'arrêt, puisqu'elle rajoute en théorie une seule itération supplémentaire.

4. Adaptation des coefficients d'énergie

4.1. Présentation du projet

a. Cahier des charges et objectifs

Le problème majeur des contours actifs est la nécessité de régler, d'ajuster les coefficients pondérateurs des différentes énergies mises en jeu. Et le snake reste très dépendant de ces coefficients. Plusieurs méthodes ont été inventées pour améliorer le contour

actif, comme le « Line Search », le « momentum » ou la « liste Tabou », mais celles-ci n'utilisaient pas les coefficients pour améliorer la convergence du snake. Aussi, il m'a été demandé d'écrire et de tester une technique issue de la théorie de l'apprentissage (c'est-à-dire dont le comportement dépend d'informations antérieures).

L'algorithme devait dans un premier temps savoir quelle était l'énergie la plus suivie par le snake, puis il devait modifier le coefficient de cette énergie pour les itérations suivantes. Deux cas se sont alors précisés : nous calculons l'énergie la plus suivie pour chaque point, ce que nous appellerons le mode local, ou nous calculons l'énergie prépondérante pour tous les points (en faisant une moyenne par exemple), ce que nous nommerons le mode global.

De plus, un troisième mode de modification des coefficients s'incorpora dans l'algorithme. Au lieu d'augmenter ou de diminuer le coefficient de l'énergie la plus suivie avec une valeur constante (10 ou 15 % par exemple), ce dernier mode modifie le paramètre suivant la variation de l'énergie d'un même point (donc localement), entre deux itérations.

Enfin, nous déterminerons à l'aide de tests, si notre algorithme est efficace et améliore bien le contour actif à coefficients constants et le contour actif autonome.

b. Planning

Voici le planning final élaboré et modifié chaque semaine.

Travaux \ Semaines	5 – 12 avril	12 – 19 avril	19 – 26 avril	26 avril – 03 mai	03 – 10 mai	10 – 17 mai	17 – 24 mai	24 – 31 mai	31 mai – 07 juin	07 – 14 juin	14 – 21 juin
Etude du contour actif et du programme d'un PFE				VACANCES							
Amélioration du programme (ajout de nouveaux paramètres, correction d'erreurs…)											
Création de fichiers Excel et images (bmp) pour le stockage des informations (images et valeurs)											
Etude de l'algorithme sur l'adaptation des coefficients d'énergie											
Ajout d'une boite de dialogue et de nouveaux paramètres											
Programmation de l'algorithme au niveau local et global											
Réécriture de l'algorithme dans un autre programme											
TESTS de l'algorithme Pour le snake non-autonome											
Programmation de l'algorithme pour le snake autonome											
TESTS de l'algorithme pour le snake autonome											
Rédaction du rapport											

Tableau 1. Planning

4.2. Analyse fonctionnelle

Nous allons dans ce paragraphe reprendre l'analyse fonctionnelle de l'algorithme, élaboré avant chaque ligne de programmation.

a. Détermination de l'influence de chaque énergie

Pour connaître la proportion de suivi des différentes énergies et, par conséquent, l'énergie la plus suivie, plusieurs méthodes existent. Nous avons tout d'abord utilisé la méthode dite des distances euclidiennes, puis la méthode de corrélation. Mais avant de voir ces méthodes, il faut définir au préalable ce qu'on appelle le suivi des énergies.

Comme nous l'avons vu précédemment, le contour actif se déplace dans l'image et s'arrête dès qu'un contour est détecté. Pour ce faire, le snake interroge toutes les énergies en leur demandant quel est le pixel du voisinage qui a le moins d'énergie. Chaque énergie va donc sélectionner le meilleur pixel suivant son propre critère. Le contour actif choisit alors le pixel le plus cité et par conséquent le plus propice au déplacement.

La première méthode utilisée fut donc celle des distances euclidiennes. Le procédé fut le suivant :

- récupérer les déplacements désignés par chaque énergie *(Pn)*
- récupérer le déplacement réellement effectué par le snake *(P)*
- mettre dans un tableau les distances euclidiennes *Dn* entre les pixels désignés par chaque énergie et le pixel choisi.

La distance *Dn* d'une énergie *n* sera telle que :

$$Dn = \left\| \vec{P}n - \vec{P} \right\| = \sqrt{\left(Pn.X - P.X\right)^2 + \left(Pn.Y - P.Y\right)^2}$$

Ainsi, lorsque le point désigné correspond au point du déplacement, la valeur *Dn* est nulle. L'énergie la plus suivie sera donc celle qui aura la valeur la plus faible.

En réalisant une moyenne de ces résultats à chaque point et à chaque itération, nous obtenons une bonne estimation du suivi des énergies. (Récapitulatif des résultats en Annexe 1 p 67)

Cette méthode est certes assez précise, mais elle est aussi coûteuse en temps de calcul : il y a deux carrés et une racine. Une autre méthode fut alors adoptée. La méthode de corrélation utilise comme son nom l'indique le calcul des coefficients de corrélation des énergies.

Le procédé est similaire à la méthode des distances euclidiennes :

- récupérer les déplacements désignés par chaque énergie *(Pn)*
- récupérer le déplacement réellement effectué par le snake *(P)*
- mettre dans un tableau les coefficients de corrélation entre les pixels désignés par chaque énergie et le pixel choisi.

La coefficient de corrélation *Cn* d'une énergie sera tel que :

Si Pn = P

Alors Cn = Cn + 1

Sinon Si Pn = - P

Alors Cn = Cn - 1

Cet algorithme délivre alors directement l'énergie la plus suivie puisque plus la valeur *Cn* de l'énergie *n* est élevée est plus l'énergie est influente.

Cette méthode est moins précise que la première, mais avec un snake de 100 points, nous pouvons considérer que le calcul des coefficients de corrélation sur dix itérations est suffisant (cela fait quand même 100x10 = 1000 valeurs en global, ou 10 en local).

Voici une comparaison des deux méthodes faite sur 100 points et 10 itérations:

COEFFICIENTS DE CORRELATION

Courbure	Continuité	Gradient	Ballon
0,98873188	0,24902142	0,06059258	0,46796626

DISTANCES EUCLIDIENNES

Courbure	Continuité	Gradient	Ballon
0,15140926	0,31958281	0,49387699	0,44177461

Tableau 2. Comparaison Corrélation - Distance Euclidienne

L'énergie la plus suivie possède la valeur la plus grandes suivant la méthode de Corrélation et la plus faible suivant la méthode des distances euclidiennes. Ces deux méthodes donnent donc le même résultat. Nous avons normalisé les distances euclidiennes par normalisée par la racine de 72.

La figure suivante montre les différents éléments ajoutés dans la boîte de dialogue, regroupant tous les coefficients introduits par notre algorithme, et expliqués dans la suite de ce chapitre.

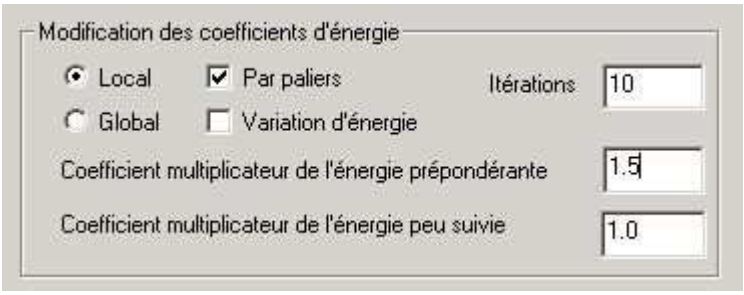


Figure 20. La paramètres ajoutés

b. La modification locale et globale

Pour qu'une modification locale puisse voir le jour, il nous a fallu intégrer au programme SnakeMagick des paramètres propres à chaque points du snake et à chaque

itération, car initialement, celui-ci ne gardait aucune information sur l'état de ces points. Pour réaliser cela, nous avons décidé de modifier le programme en ajoutant une classe nommée PtSnake, contenant de multiples paramètres comme : les valeurs des énergies, les coordonnées du point, les coefficients d'énergie, etc… Nous en créons donc un nombre assez important : *Nombre de Point du Snake* x *Nombre d'itérations*. Une fois que nous avons substitué cette classe aux valeurs globales anciennement utilisées, nous pouvons calculer, notament, les coefficients de corrélation des énergies sur plusieurs itérations, ce que nous n'aurions pu entreprendre auparavant.

Après avoir déterminer quelle était le classement des énergies suivant leur influence, il ne reste plus qu'à modifier le coefficient de cette énergie. L'organigramme ci-dessous résume les étapes.

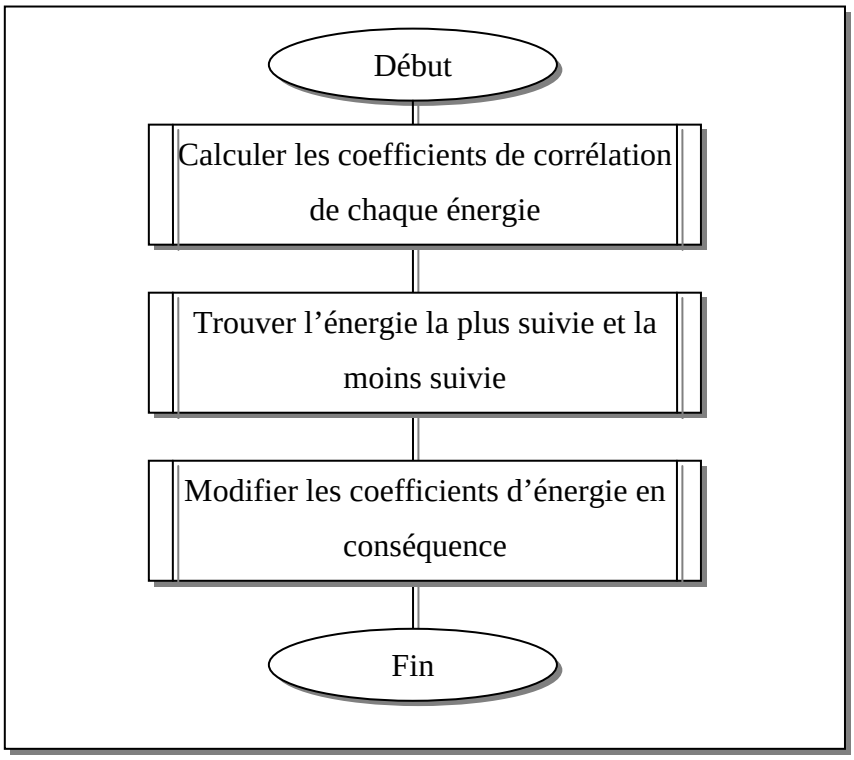


Figure 21. Organigramme de l'algorithme de modification des coefficients

Le contour actif à coefficients constants était bien trop rigide avec ses coefficients d'énergie statiques, car entre le début et la fin des itérations, les énergies évoluent beaucoup, elles sont dynamiques. Aussi, il me semble ingénieux de calquer cette dynamique sur les coefficients d'énergie et ceux, de façon local pour que chaque point puisse être influent sur le lui et sur ses voisins.

Cependant, dans certaines situations il est plus nécessaire d'éliminer des points s'égarant dans une zone non désirée, que d'augmenter son influence aussi, nous avons donc entrepris

une modification globale. Ainsi, tous les points ont exactement les mêmes coefficients d'énergies, que nous modifions suivant le classement des énergies de l'ensemble des points.

Les différents tests sur ces deux modes (local et global) sont présentés dans la partie 5.

c. La modification suivant la variation d'énergie

La modification suivant la variation d'énergie est une extension de la modification locale. En effet, au lieu d'augmenter les coefficients d'énergie avec une valeur, un pourcentage fixe, nous lui additionnons un pourcentage de la variation de l'énergie. La variation d'énergie représente simplement la différence entre l'énergie du point *P* à l'itération *n* et l'énergie de ce même point à l'itération *n+1*. Cela semble tout à fait logique car de cette manière, nous gardons le « signe de l'influence ». En effet, si l'énergie d'un point diminue, alors la variation sera négative, et nous retrancherons un pourcentage de cette variation. Les coefficients vont par conséquent suivrent les fluctuations générées par le snake et par l'image.

Nous avons inséré un coefficients (pourcentage) sur la variation d'énergie car en additionnant directement la valeur, l'énergie serait bien trop dépendante des variations.

$$Coef = Coef \times (1 + \mathbf{K} \times (Energie\_it_n - Energie\_it_{n-1})/\max(Energie\_it_n, Energie\_it_{n-1}))$$

*Avec **K** le coefficient réglable mentionné dans le tableau*

Équation 7. Modification des coefficients suivant la variation d'énergie

d. Notions de paliers

Lorsque nous avons testé notre programme sur différentes images (cf. partie 4.3), nous nous sommes aperçus que l'augmentation du coefficient de l'énergie prépondérante était trop importante, au regard notamment du coefficient multiplicateur paramétré.

Si l'on regarde les valeurs des coefficients d'énergie en deux points, et avec une augmentation locale de 10 % (calculs sur dix itérations), nous obtenons ceci :

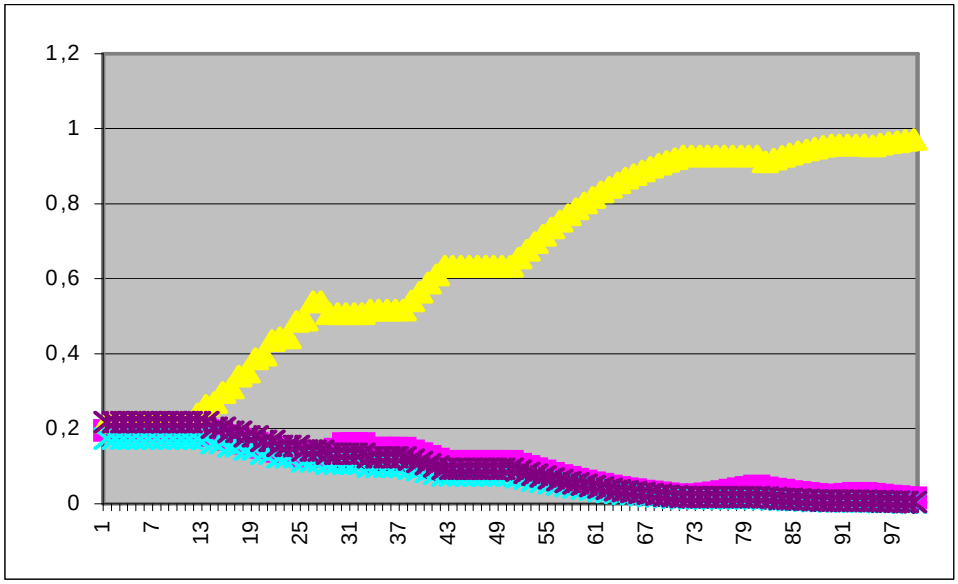


Figure 22. Augmentation locale - Point N° 29

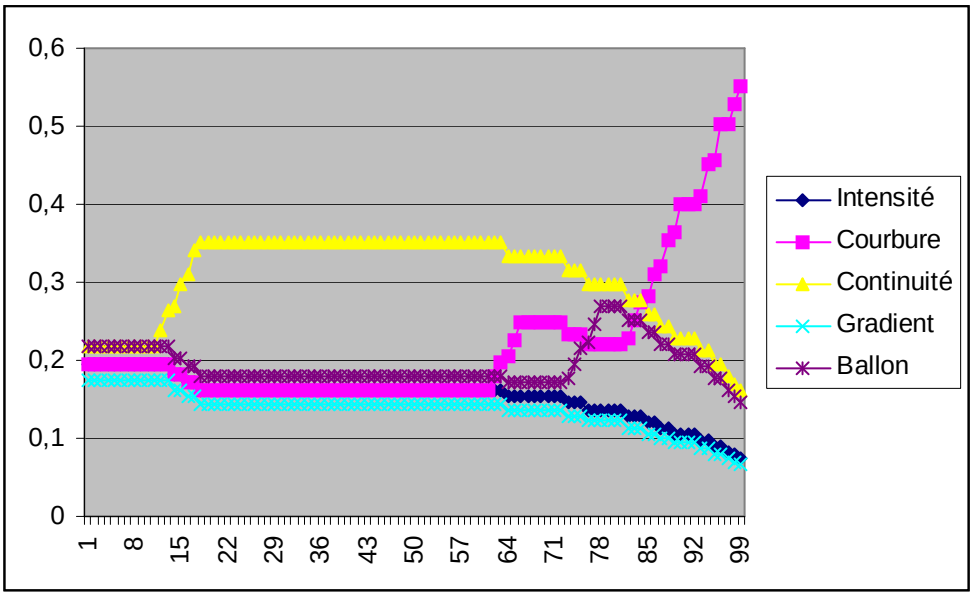


Figure 23. Augmentation locale - Point N° 30

Nous remarquons, tout d'abord, que l'augmentation est belle et bien locale puisque les coefficients d'énergie sont inégaux entre les deux points, mais surtout que pour le point N° 29, après l'itération 70, les coefficients d'intensité, de courbure, de gradient, et de ballon sont presque totalement nul. En effet, l'augmentation très forte du coefficient de l'énergie de courbure implique une certaine diminution, répartie sur les autres coefficients (du fait de la normalisation des coefficients). Et, au bout d'un certain nombre d'itérations, les valeurs deviennent nulles. Dès lors, le contour actif sera restreint à une seule énergie, ce qui ne peut que détériorer la qualité du snake.

Aussi, pour ralentir cette augmentation, nous avons décidé d'ajouter une option qui modifiera les coefficients de façon discrète, par paliers. L'algorithme augmente alors le coefficient une fois toutes les *n* itérations, ce qui donne la forme de paliers. (cf. figure 24)

Augmentation locale de 20% par palier de 10 itérations

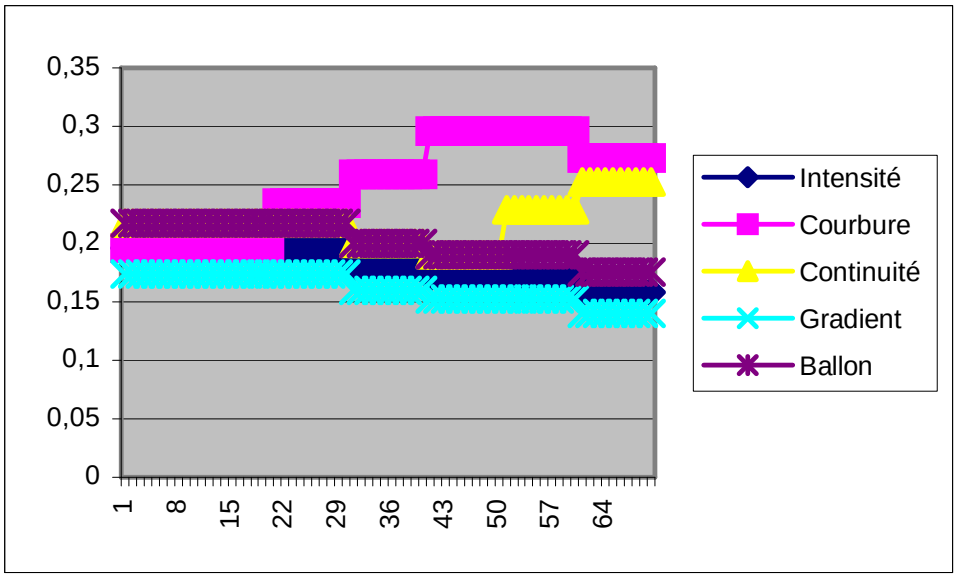


Figure 24. Augmentation locale par paliers - Point N° 29

Dans cet exemple de relevé, il n'y a plus de coefficient tendant vers de fortes valeurs, ce qui permet au snake de garder une certaine liberté dans le choix du déplacement.

Pour plus de précision, d'autres cas sont présentés en annexe 4 p 72.

e. Modification du contour actif autonome

Au cours de l'utilisation du logiciel SnakeMagick, nous nous somme rendu compte de l'existence d'une erreur avec le contour actif autonome. La figure suivante est un relevé des valeurs aléatoires des cinq coefficients d'énergie en un point du snake, suivant les itérations, avec le snake autonome.

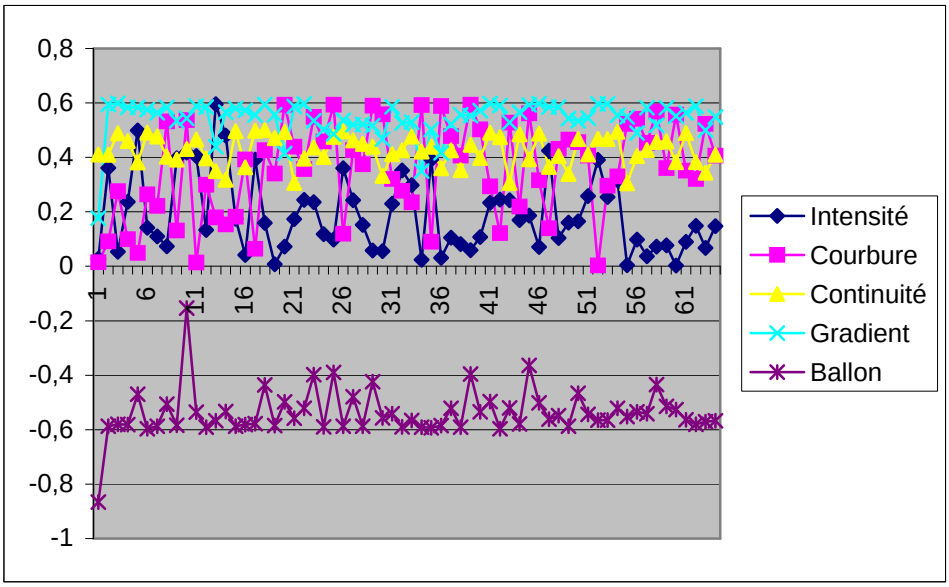


Figure 25. Variation des coefficients en un point, avec l'ancien programme

Il est en effet très surprenant d'obtenir ceci car, normalement, le contour actif ne tire pas aléatoirement les coefficients à chaque itération mais garde les anciennes valeurs si celles-ci sont meilleurs. Après ré-écriture d'une partie de la fonction *greedy*, voici ce que nous obtenons :

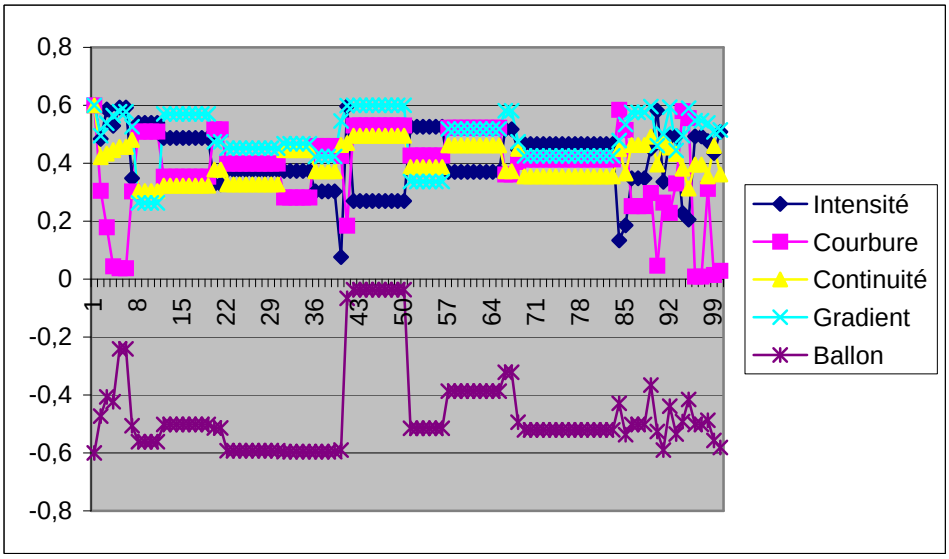


Figure 26. Variation des coefficients en un point, avec le bon algorithme

Ici, on s'aperçoit très vite que les anciennes valeurs ont bien été conservées, ces zones linéaires sont de plus très fréquentes.

5. Tests de l'algorithme avec un snake non-autonome

Afin de valider notre algorithme et surtout l'efficacité de celui-ci, nous avons entrepris différents tests sur quatre images : le CauryBroken, le Condyle, le Losange et l'Épaule.

Cependant, la réalisation de ces tests demandait de connaître au départ des coefficients d'énergie assez bons, et cela a été difficile d'en obtenir avec un programme initial erroné ! Heureusement que d'autres stagiaires en avaient déjà calculés.

Voici la synthèse des résultats sur les quatre images, mentionnant toutes les informations nécessaires à la reproduction des essais.

5.1. Le CauryBroken

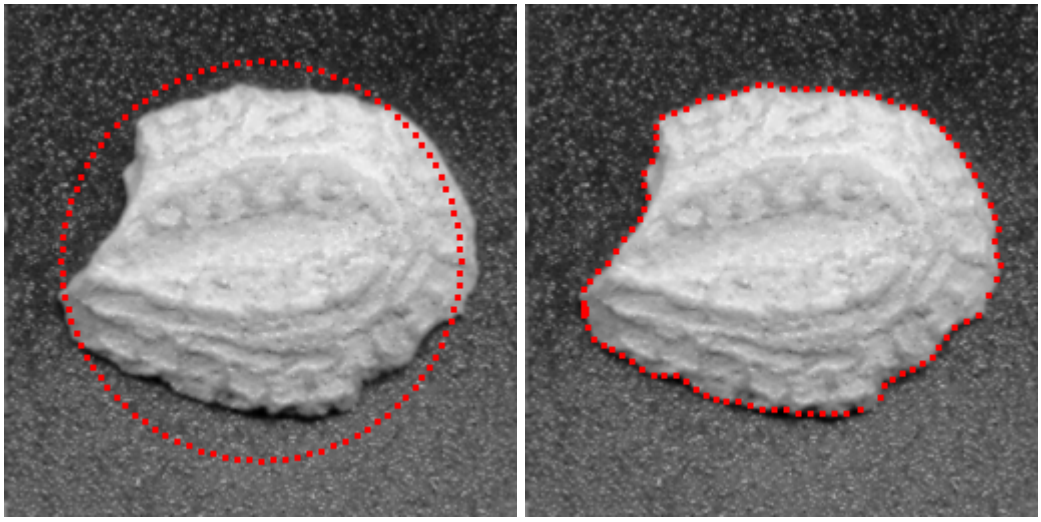


Figure 27. Le CauryBroken, lors de l'initialisation des points et à la fin des itérations

Paramètres initiaux

Image CauryBroken.bmp

Paramètres

X	Y	rayon	Points	Voisinage	Normalisation
128	128	100	100	3	(E-Emin)/(Emax-Emin)

Coefficients de départ

Intensité	Courbure	Continuité	Gradient	Ballon
0,9	0,9	1	0,8	1

Critère d'arrêt : plus de 98% des points oscillent entre 2 positions
Calcul des coefficients de corrélation sur 10 itérations

Résultats

Modification locale									
Cas N°	0	1	2	3	4	5	6	7	8
Energie prépondérante	1	1,5 - 1,05	1,5 - 1,1	0,95	0,95 - 0,9	1,5 - 1,05	1,5 - 1,1	0,95	0,95
Energie la moins suivie	1	1	1	1	1	0,95	0,95	1,5 - 1,05	1,5 - 1,05
Paliers (10 itérations)	sans	sans	avec	sans	avec	sans	avec	sans	avec
Contour respecté	OUI	NON	NON OUI	Moyen	Moyen OUI	NON	NON OUI	NON	NON
Nombre d'itérations	60	100*	100* - 51	94	100* - 46	100*	100* - 75	100*	100*
Amélioration (itérations)			9		14				

Modification globale									
Cas N°	0	1	2	3	4	5	6	7	8
Energie prépondérante	1	1,5 - 1,05	1,5 - 1,05	0,95	0,95	1,5 - 1,05	1,5 - 1,1	0,95	0,95
Energie la moins suivie	1	1	1	1	1	0,95	0,95	1,5 - 1,05	1,5 -1,1
Paliers (10 itérations)	sans	sans	avec	sans	avec	sans	avec	sans	avec
Contour respecté	OUI	NON	MOYEN	MOYEN	OUI	NON OUI	NON OUI	NON OUI	NON OUI
Nombre d'itérations	60	100*	100*	100	57*	100* - 30*	100* - 75	100* - 41	100* - 42
Amélioration						30		19	18

Modification locale suivant la variation d'énergie					
Coefficient multiplicateur de la variation d'énergie	0	0,1	0,1	-0,1	-0,2
Paliers (10 itérations)	sans	sans	avec	sans	avec
Contour respecté	OUI	OUI	OUI	OUI	OUI
Nombre d'itérations	60	42	45	26	30
Amélioration		18**	15	34***	30***

Tableau 3. Résultats des tests sur le CauryBroken, avec un snake non-autonome

Légende pour les tableaux des 4 essais de cette partie 5

100 : Arrêt à l'itération 100.*

*100** : Arrêt lorsque la distance moyenne des déplacements est inférieure à 1,4 pixel*

*100*** : Arrêt lorsque 70 % des points oscillent entre deux positions*

Précision sur les tableaux

Les lignes « Energie prépondérante » et « Energie la moins suivie » correspondent aux coefficients multiplicateurs attribués au cours du traitement.

La ligne « Amélioration » correspond au gain final d'itérations, par rapport à l'essai N° 0, sans modification de coefficient.

Au terme « Moyen » correspond un contour quasi-détecté. (cf. annexe 2 p 68, cas N° 3 en mode local)

Le mode « par paliers » signifie que l'algorithme modifie les coefficients toutes les 10 itérations, et garde ces mêmes valeurs pour les itérations 10 x n + k (n,k∈N ; k<10).

La modification du coefficient de l'énergie prépondérante, suivant la variation d'énergie est telle que :

Coef = Coef x (1 + **K** x (Energie_it$_n$ - Energie_it$_{n-1}$)/max(Energie_it$_n$,Energie_it$_{n-1}$))

*Avec **K** le coefficient réglable mentionné dans le tableau*

Les images des différents cas proposés sont en annexe 2 p 68.

Conclusion

Sur cette image très bruitée, plusieurs jeux de coefficients multiplicateurs améliorent la rapidité de convergence du snake.

En effet, une faible augmentation ou diminution de 10% de l'énergie prépondérante, à l'échelle locale et par paliers, permettent de gagner de 9 à 14 itérations.

Cependant, la modification globale des coefficients permet d'obtenir de meilleurs résultats. En augmentant sensiblement l'énergie la moins suivie de 5 à 10% et en diminuant l'énergie la plus suivie de 5%, nous obtenons le contour de l'image dès l'itération 41-42, soit une amélioration de 18 (avec paliers) à 19 (sans palier) itérations, ou une diminution du nombre d'itération de 30%. Mais cette diminution peut encore augmenter puisqu'elle atteint les 50% lorsque l'on augmente de 5% l'énergie la plus suivie, et que l'on diminue de 5% l'énergie la moins suivie.

En ce qui concerne le modification locale des coefficients suivant la variation d'énergie, nous obtenons de bons résultats puisqu'un gain de 15 à 34 itérations est envisageable, et celui-ci pourrait être encore amélioré en ajustant le critère d'arrêt.

5.2. Le Condyle

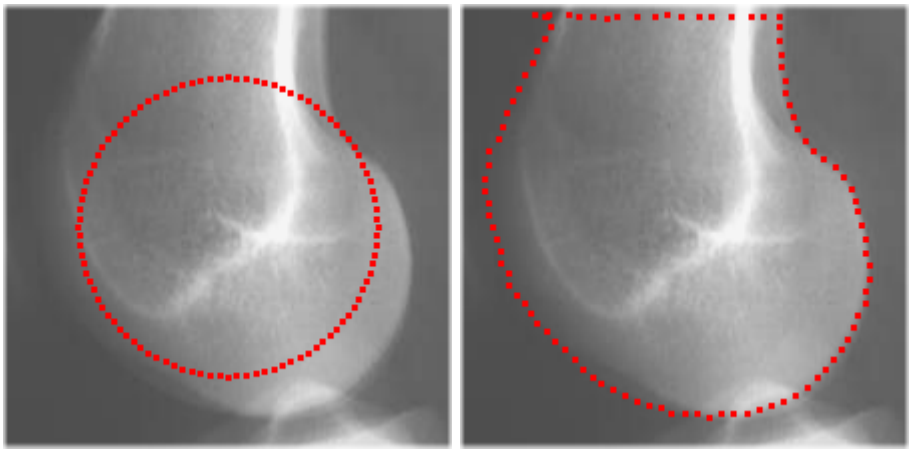


Figure 28. Le Condyle, lors de l'initialisation des points et à la fin des itérations

Paramètres initiaux

Image **Condyle224x224.bmp**
Paramètres

X	Y	rayon	Points	Voisinage	Normalisation
112	111	75	100	3	(E-Emin)/(Emax-Emin)

Coefficients de départ

Intensité	Courbure	Continuité	Gradient	Ballon
0,1	0,6	0,6	0,65	-1

Critère d'arrêt : plus de 90% des points oscillent entre 2 positions
Calcul des coefficients de corrélation sur 10 itérations

Résultats

Modification locale									
Energie prépondérante	1	1,5 - 1,05	1,5 - 1,05	0,95	0,95	1,5 - 1,05	1,5 - 1,05	0,95	0,95
Energie la moins suivie	1	1	1	1	1	0,95	0,95	1,5 - 1,05	1,5 - 1,05
Paliers (10 itérations)	sans	sans	avec	sans	avec	sans	avec	sans	avec
Contour respecté	OUI	NON	NON MOYEN	MOYEN	MOYEN	NON	NON MOYEN	NON	NON MOYEN
Nombre d'itérations	76	100*	100* - 76	100*	56	100*	100* - 57	100*	100* - 84

Amélioration

Modification globale									
Energie prépondérante	1	1,5 - 1,05	1,5 - 1,05	0,95	0,95 - 0,9	1,5 - 1,05	1,5 - 1,05	0,95	0,95
Energie la moins suivie	1	1	1	1	1	0,95	0,95	1,5 - 1,05	1,5 -1,05
Paliers (10 itérations)	sans	sans	avec	sans	avec	sans	avec	sans	avec
Contour respecté	OUI	NON MOYEN	NON MOYEN	NON	OUI	NON	NON MOYEN	NON	NON
Nombre d'itérations	76	100* - 62	100*	100*	76	100*	100*	100* - 76	100* - 72

Amélioration

Modification locale suivant la variation d'énergie					
Coefficient multiplicateur de la variation d'énergie	0	0,1	0,1	-0,05	-0,1
Paliers (10 itérations)	sans	sans	avec	sans	avec
Contour respecté	OUI	MOYEN	MOYEN	MOYEN	MOYEN
Nombre d'itérations	76	99	100*	62	58

Amélioration

Tableau 4. Résultats des tests sur le Condyle, avec un snake à coefficients non-aléatoires

Conclusion

Les tests effectués sur cette image ne sont pas probants, aucun jeu de coefficients n'améliore le contour actif. Il est même difficile d'atteindre le contour de l'image avant les 100 itérations.

L'adaptation locale, globale et suivant la variation d'énergie détériorent plus le snake qu'il ne l'aide. Il faut remarquer néanmoins que pour un coefficient négatif, le snake s'arrête avant les 76 itérations et, même si le contour pourrait être meilleur, il se peut qu'un ajustement du coefficient et du critère d'arrêt permette d'obtenir un bon contour.

L'image du condyle est difficile à travaillée car les contours sont beaucoup moins nets qu'avec une image comme le CauryBroken. Il se peut que notre algorithme améliore le snake sur le condyle, mais ce serait avec des valeurs d'augmentation-diminution inférieures à 1%. Cela n'apporterait alors rien à notre sujet, le but de celui-ci étant d'arriver à obtenir le contour d'une image en partant de coefficient non-optimaux, et ce, afin justement d'éviter les réglages des paramètres.

5.3. Le Losange

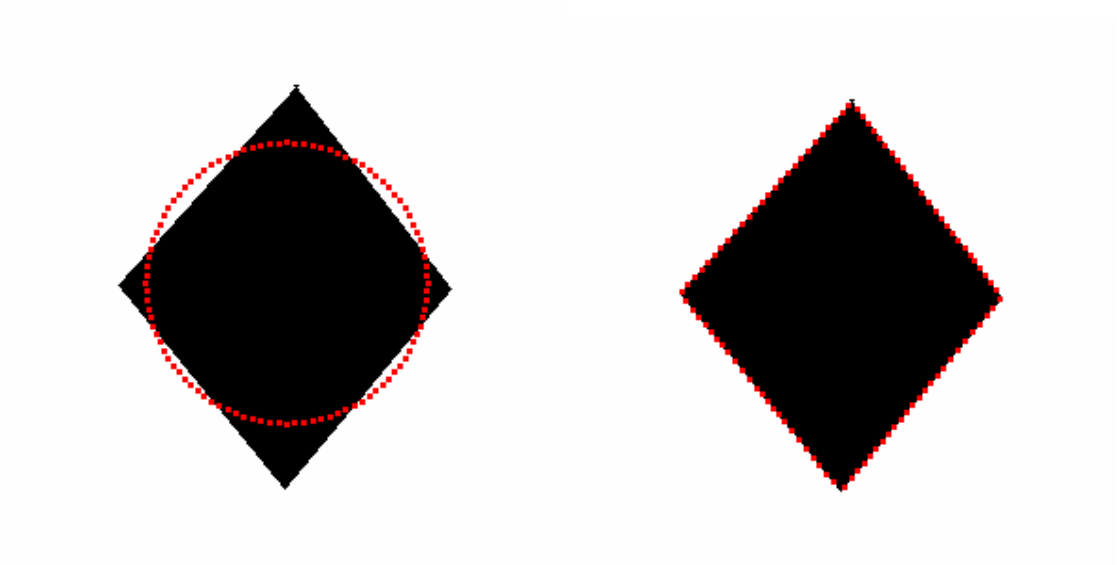


Figure 29. Le Losange, lors de l'initialisation des points et à la fin des itérations

Paramètres initiaux

Image losange.bmp
Paramètres

X	Y	rayon	Points	Voisinage	Normalisation
150	150	75	100	3	(E-Emin)/(Emax-Emin)

Coefficients de départ

Intensité	Courbure	Continuité	Gradient	Ballon
0,1	0,6	0,8	1	-1

Critère d'arrêt : plus de 90% des points oscillent entre 2 positions
Calcul des coefficients de corrélation sur 10 itérations

Résultats

Modification locale									
Energie prépondérante	1	1,5 - 1,05	1,5 et +	0,95	0,95	1,5 - 1,15	1,5	0,95	0,95
Energie la moins suivie	1	1	1	1	1	0,95	0,95	1,5 - 1,05	1,5
Paliers (10 itérations)	sans	sans	avec	sans	avec	sans	avec	sans	avec
Contour respecté	OUI	NON - OUI	OUI	OUI	OUI	NON - OUI	OUI	NON - OUI	OUI
Nombre d'itérations	32	100* - 29	28	41	26	100* - 28	29	100* - 26*	30
Amélioration		3	4		6	4	3	6	2

Modification globale									
Energie prépondérante	1	1,5 - 1,05	1,7	0,95	0,95 - 0,8	1,5 - 1,05	1,5	0,95	0,95
Energie la moins suivie	1	1	1	1	1	0,95	0,95	1,5 - 1,05	1,5 -1,05
Paliers (10 itérations)	sans	sans	avec	sans	avec	sans	avec	sans	avec
Contour respecté	OUI	NON - OUI	OUI	OUI	OUI	NON - OUI	OUI	NON	OUI
Nombre d'itérations	32	100* - 28	27	28	27 - 21	100* - 32	28	100*	32
Amélioration		4	5	4	9		4		

Modification locale suivant la variation d'énergie					
Coefficient multiplicateur de la variation d'énergie	0	0,2	0,1	-0,1	-0,15
Paliers (3 itérations)	sans	sans	avec	sans	avec
Contour respecté	OUI	OUI	OUI	NON	OUI
Nombre d'itérations	32	27	25	100*	21
Amélioration		5	7		9

Tableau 5. Résultats des tests sur le Losange, avec un snake à coefficients non-aléatoires

Conclusion

Sur cette image synthétique, les résultats sont assez homogènes. Dans presque tous les cas, nous réduisons le nombre d'itérations de départ 32 à 21, au maximum (soit environ de 30 %).

Il faut remarquer que c'est avec de faibles augmentations-diminutions que nous obtenons les meilleurs résultats, sauf peut-être lorsque l'on augmente seulement l'énergie prépondérante par paliers.

Il est toutefois surprenant d'obtenir une amélioration de quatre à cinq itérations en augmentant l'énergie prépondérante et en la diminuant. Aussi, nous avons entrepris de vérifier si l'algorithme fonctionnait correctement. Voici les relevés des coefficients des énergies du snake effectués au point numéro dix, sur une même image (le losange), dans les deux cas suivants :

- Diminution globale de 5 % de l'énergie prépondérante, sans palier
- Augmentation globale de 70 % de l'énergie prépondérante, avec paliers (en bas)

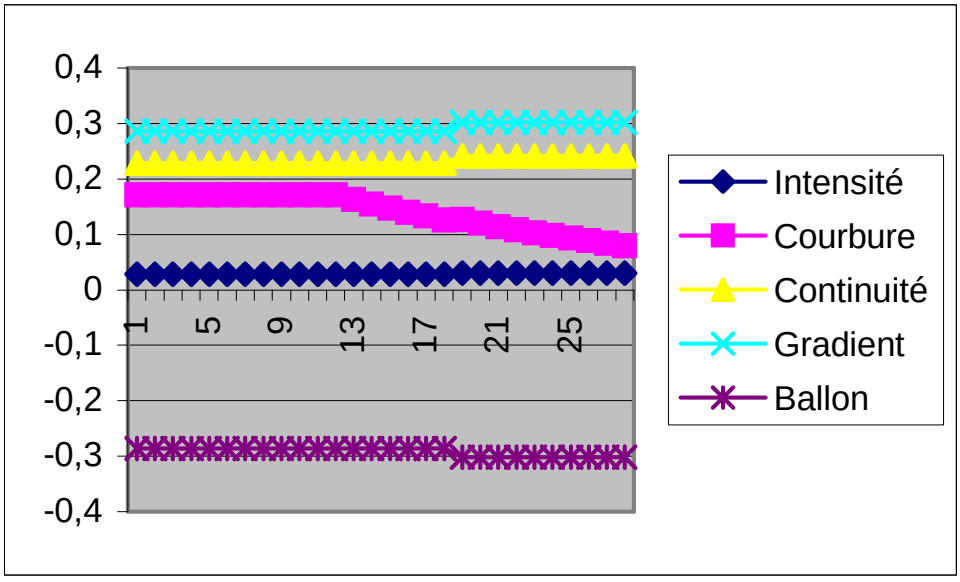


Figure 30. Diminution globale de 5% de l'énergie prépondérante, sans palier

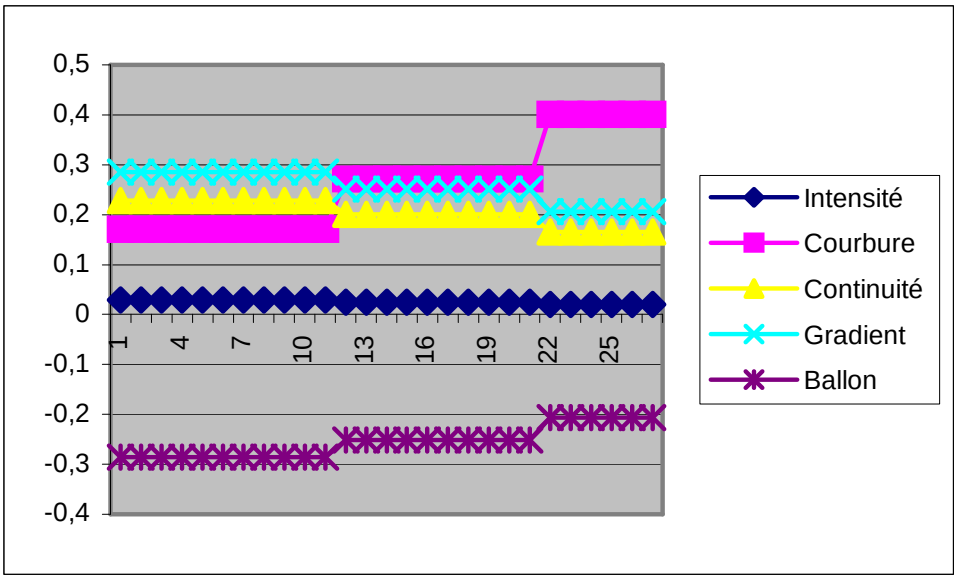


Figure 31. Augmentation globale de 70 % de l'énergie prépondérante, avec paliers

Ces deux graphiques montrent bien que la première énergie prépondérante (courbure) est atténuée dans un cas et augmentée dans l'autre, ce qui atteste de la validité de notre programmation.

De plus, la modification locale suivant la variation d'énergie permet elle aussi de gagner neuf itérations dans le meilleur des cas.

En somme, notre algorithme améliore nettement la rapidité du snake. Celui-ci atteint le contour avec une vitesse moyenne 20 % plus grande (30 % au maximum) qu'avec des coefficients d'énergies statiques.

5.4. L'Épaule

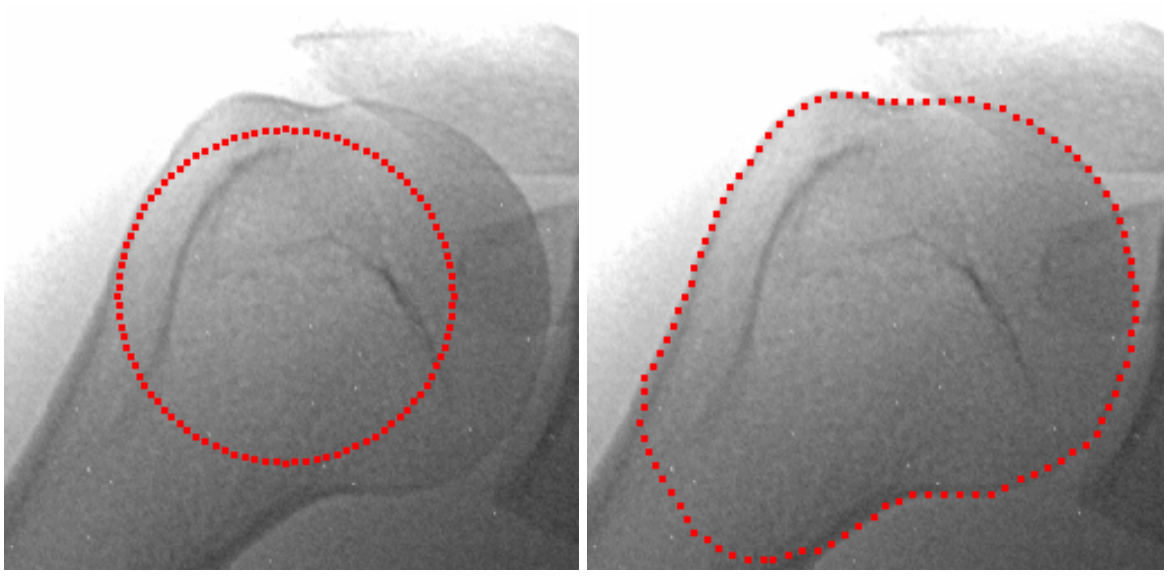


Figure 32. L'Épaule, lors de l'initialisation des points et à la fin des itérations

Paramètres initiaux

Image **Épaule.bmp**

Paramètres

X	Y	rayon	Points	Voisinage	Normalisation
125	130	75	100	3	(E-Emin)/(Emax-Emin)

Coefficients de départ

Intensité	Courbure	Continuité	Gradient	Ballon
0,4	0,8	1	0,467	-1

Critère d'arrêt : plus de 90% des points oscillent entre 2 positions
Calcul des coefficients de corrélation sur 10 itérations

Résultats

Modification locale									
Energie prépondérante	1	1,5 - 1,05	1,5 - 1,05	0,95	0,95	1,5 - 1,05	1,5 - 1,05	0,95	0,95
Energie la moins suivie	1	1	1	1	1	0,95	0,95	1,5 - 1,05	1,5 - 1,05
Paliers (10 itérations)	sans	sans	avec	sans	avec	sans	avec	sans	avec
Contour respecté	OUI	NON MOYEN	NON	NON	NON	NON	NON	NON	NON
Nombre d'itérations	84	100*	100*	100*	100*	100*	100*	100*	100*

Amélioration

Modification globale									
Energie prépondérante	1	1,5 - 1,05	1,5 - 1,05	0,95	0,95	1,5 - 1,05	1,5 - 1,05	0,95	0,95
Energie la moins suivie	1	1	1	1	1	0,95	0,95	1,5 - 1,05	1,5 -1,05
Paliers (10 itérations)	sans	sans	avec	sans	avec	sans	avec	sans	avec
Contour respecté	OUI	NON	NON	NON	NON	NON	NON	NON	NON
Nombre d'itérations	84	51	100*	100*	100*	100*	100*	100*	100*

Amélioration

Modification locale suivant la variation d'énergie					
Coefficient multiplicateur de la variation d'énergie	0	0,1	0,1	-0,02	-0,045
Paliers (10 itérations)	sans	sans	avec	sans	avec
Contour respecté	OUI	NON	OUI	OUI	OUI
Nombre d'itérations	84	100*	100*	57***	57***
Amélioration				27	27

Tableau 6. Résultats des tests sur l'épaule, avec un snake à coefficients non-aléatoires

Conclusion

L'image de l'épaule est très proche de celle du condyle, en terme de bruit et de netteté des contours. Aussi, tous les tests effectués au niveau local ou global, ou suivant la variation d'énergie, se sont avérés totalement inefficaces, voir même néfastes pour le contour actif.

5.5. Temps supplémentaire

Nous avons de plus réalisé plusieurs tests sur le CauryBroken pour savoir si l'algorithme prenait beaucoup de temps de calcul, auquel cas, l'ajout de temps compenserait la diminution du nombre d'itérations et notre algorithme serait peu intéressant.
Tests sur le CauryBroken, avec 100 itérations et 100 points (sans critère d'arrêt) :

- Temps pour les 100 itérations : **7,0 secondes.**
- Temps avec l'algorithme en mode local (énergie prépondérante augmentée de 10 %, énergie la moins suivie diminuée de 10 %, sans palier, 10 itérations pour le calcul des coefficients de corrélation) : **7,1 secondes.**
- Temps avec l'algorithme en mode global (énergie prépondérante augmentée de 10 %, énergie la moins suivie diminuée de 10 %, sans palier, 10 itérations pour le calcul des coefficients de corrélation) : **7,1 secondes.**
- Temps avec l'algorithme en mode variation d'énergie : **7,1 secondes.**

L'algorithme ne prend pas beaucoup de temps supplémentaire pour calculer les coefficients de corrélation des énergies et pour modifier les coefficients.

(Les temps proposés si dessus sont fortement dépendants de chaque ordinateur, suivant la puissance de ceux-ci et des programmes fonctionnant en même temps.)

5.6. Synthèse des résultats

L'ensemble des résultats présents dans cette partie tend à montrer que la modification des coefficients suivant l'importance des énergies dans la décision du déplacement était assez prometteuse sur des images comme celle du CauryBroken ou du losange, ou plus

généralement sur des images où les contours sont assez prononcés, comme dans les images synthétiques par exemple. Les tests sur l'épaule ou le condyle montrent que l'algorithme n'a pas un effet bénéfique, bien au contraire, sur le contour actif. Le bruit, la netteté des contours, ou la forme de l'objet sont peut-être à l'origine de cette défaillance.

Cette méthode est de plus peu coûteuse en temps de calculs aussi, le gain en itération est tout à fait révélateur de l'amélioration apportée par notre algorithme.

Discutions

Les tests effectués sur le CauryBroken et le losange montrent qu'une augmentation du coefficient de l'énergie la plus suivie entraîne une diminution du nombre d'itérations nécessaire à la détection des contours. Cela semble cohérent : puisque cette énergie est très suivie par le snake, pourquoi ne pas augmenter son influence vis-à-vis de celui-ci. Nous obtenons dès lors une certaine amélioration, sauf sur le condyle et l'épaule, je le rappelle.

Mais ce qui est intéressant de voir, c'est qu'une diminution locale de cette même énergie entraîne aussi une amélioration du contour actif.

La cause venait peut-être de la normalisation des coefficients, commune aux deux cas. Mais après avoir lancer le programme sans augmentation ni diminution des valeurs, mais en ajoutant simplement la normalisation des coefficients, nous obtenons exactement le même résultat que sans normalisation.

Cette situation ne vient donc pas de paramètres auxiliaires mais bien des coefficients eux-mêmes, et de l'influence qu'ils ont sur la décision du snake. Une autre explication peut être à l'origine de cette particularité.

Nous sommes partis pour réaliser nos expérimentations de coefficients d'énergies statiques pseudo-optimaux. Nous pourrions alors penser que l'énergie ayant le plus fort coefficient serait l'énergie prépondérante. Or, entre la première itération et la dernière, le fait est que le classement de suivi des énergies change. Pendant un certain temps ce sera l'énergie de courbure, puis l'énergie de continuité, etc… (cf. figure suivante : à une diminution correspond la prépondérance de l'énergie)

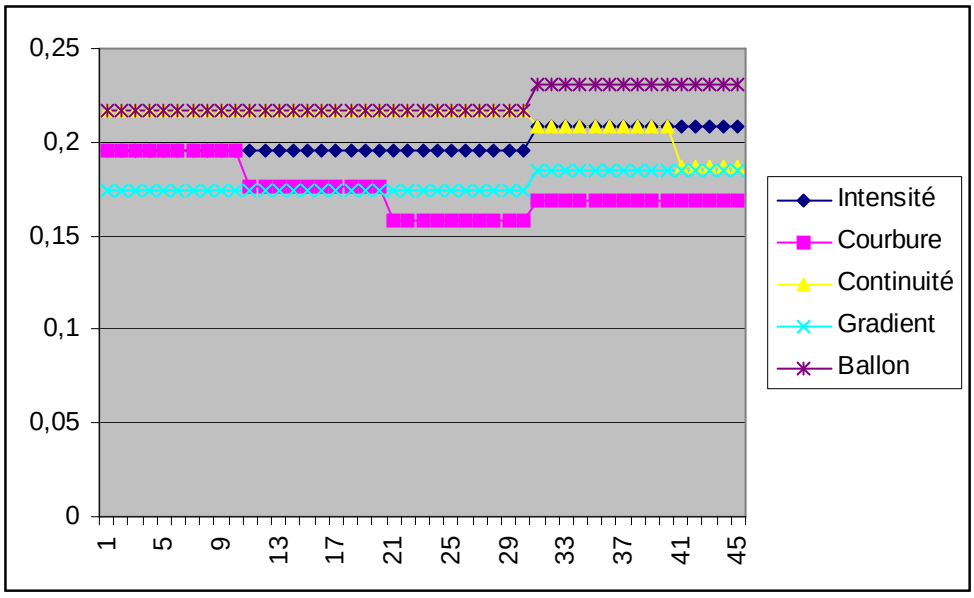


Figure 33. Etude de l'évolution des coefficients de corrélations des énergies

Par conséquent, puisque l'influence des énergies varient au cours du traitement, les coefficients de départ sont sensés être des moyennes (pour tous les points et pour toutes les itérations). Et, puisqu'il s'agit de moyennes, les valeurs peuvent osciller au dessus et en dessous de celle-ci. Cela pourrait justifier l'amélioration lorsque nous mettons un coefficient négatif.

A la fin de mon stage, il m'a été demandé de tester les capacités de cet algorithme sur le contour actif autonome. Nous allons donc présenter dans la partie suivante, les différents tests effectués sur deux images.

6. Tests de l'algorithme avec un contour actif autonome

Pour tester la capacité de notre algorithme sur le snake autonome, nous avons décidé d'entreprendre deux séries de tests sur le losange (image synthétique) et le condyle. Etant donné le caractère aléatoire des coefficients, nous avons réalisé trois essais minimum pour chaque jeu de coefficients d'adaptation. Le nombre d'itérations et l'appréciation annotée ne sont donc que les moyennes de ces essais.

Précisions sur les appréciations

Le terme « MOYEN » signifie que le taux de contour parfait est inférieur à 50 %.

Nous avons de plus mentionné le taux de réussite (contour parfait) lorsque nous avions obtenu les bons contours car les résultats ne sont jamais identiques à chaque essais (cf. figure 34). Cela permet de mieux apprécier l'influence de notre amélioration.

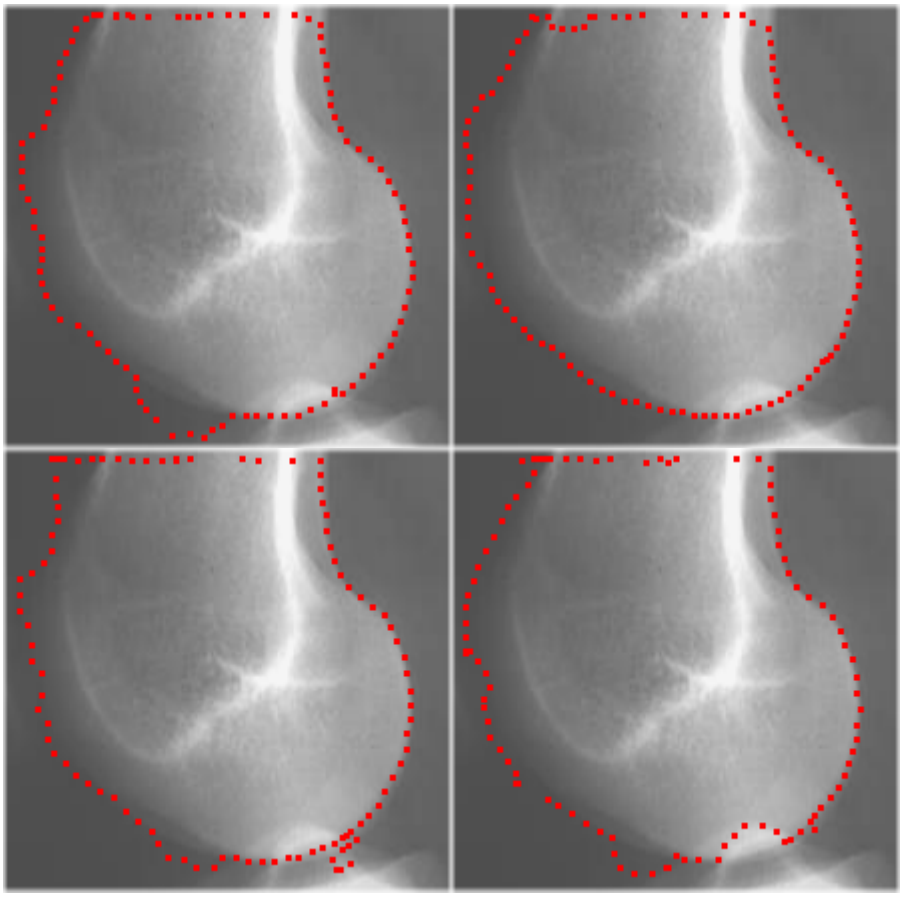


Figure 34. Tests répétés sur le condyle, en mode autonome

6.1. Le losange

L'initialisation du snake sur le losange a été modifiée car tous tests effectués avec un rayon initial de 75 pixels étaient erronés.

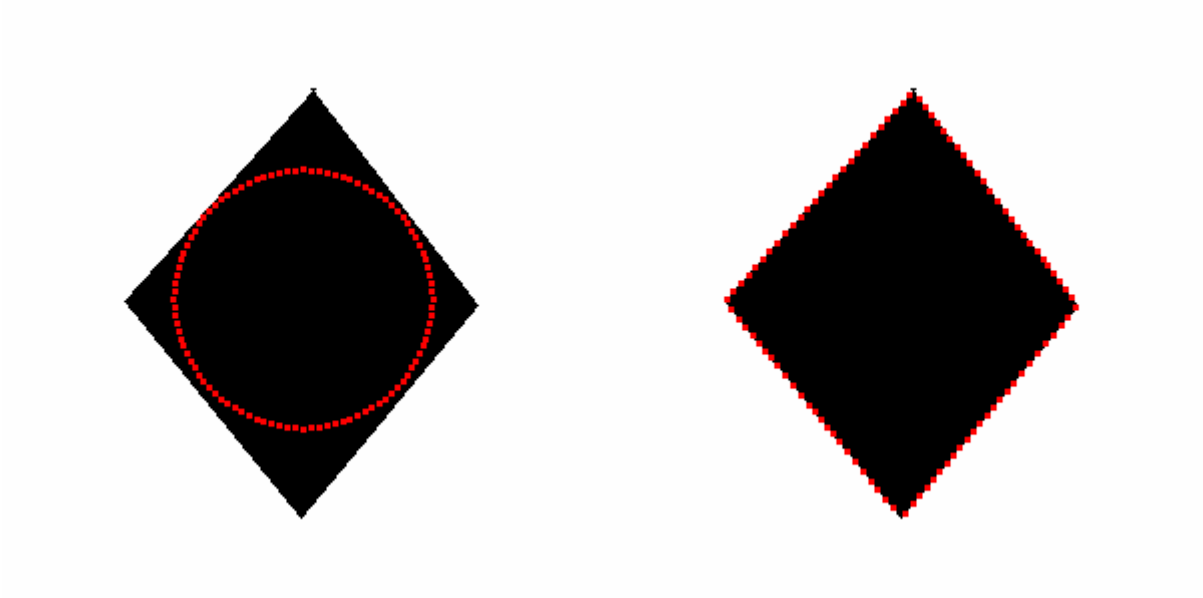


Figure 35. Le Losange, lors de l'initialisation des points et à la fin des itérations

Paramètres initiaux

Image losange.bmp
Paramètres

X	Y	rayon	Points	Voisinage	Normalisation
150	150	65	100	3	(E-Emin)/(Emax-Emin)

Coefficients de départ

Intensité	Courbure	Continuité	Gradient	Ballon
1	1	1	1	-1

Critère d'arrêt : AUCUN car ils ne fonctionnent pas avec le snake autonome
Calcul des coefficients de correlation sur 10 itérations

Les coefficients de départ mentionnés ci-dessus ne représentent pas la valeur des coefficients (puisqu'ils sont choisis aléatoirement), mais il s'agit des valeurs maximales de ces mêmes coefficients. Ainsi, pour l'énergie de ballon par exemple, le coefficient variera entre 0 et –1.

Après avoir essayer de nombreux critères d'arrêt, nous nous sommes rendus compte que ceux-ci n'arrêtaient jamais le contour actif. En effet, les coefficients aléatoires changeaient la direction des points à chaque itération.

Modification locale									
Energie prépondérante	1	1,1	1,1	0,9	0,9	1,1	1,1	0,9	0,9
Energie la moins suivie	1	1	1	1	1	0,9	0,9	1,1	1,1
Paliers (10 itérations)	sans	sans	avec	sans	avec	sans	avec	sans	avec
Contour respecté	MOYEN	MOYEN	OUI (4/4)	OUI (4/4)	OUI (4/4)	OUI (2/4)	OUI (4/4)	MOYEN	OUI (3/4)
Nombre d'itérations	42	34	33	34	35	70	32	33	36
Amélioration			**OUI**	**OUI**	**OUI**	**OUI**	**OUI**		**OUI**

Modification globale									
Energie prépondérante	1	1,1	1,1	0,9	0,9	1,1	1,1	0,9	0,9
Energie la moins suivie	1	1	1	1	1	0,9	0,9	1,1	1,1
Paliers (10 itérations)	sans	sans	avec	sans	avec	sans	avec	sans	avec
Contour respecté	MOYEN	NON	OUI (4/4)	OUI (4/4)	OUI (4/4)	NON	MOYEN	OUI (4/4)	OUI (4/4)
Nombre d'itérations	42	100	53	29	35	100	70	26	32
Amélioration			**OUI**	**OUI**	**OUI**			**OUI**	**OUI**

Modification locale suivant la variation d'énergie					
Coefficient multiplicateur de la variation d'énergie	0	0,1	0,1	-0,1	-0,1
Paliers (3 itérations)	sans	sans	avec	sans	avec
Contour respecté	MOYEN	OUI (4/4)	OUI (4/4)	OUI (4/4)	OUI (4/4)
Nombre d'itérations	42	35	32	33	32
Amélioration		**OUI**	**OUI**	**OUI**	**OUI**

Tableau 7. Résultats des tests sur le Losange, avec un snake autonome

Conclusion

Tout d'abord, il est intéressant de remarquer que dans le cas N° 0, c'est-à-dire lorsque nous avons le snake autonome seul, les différents essais se sont avérés imparfaits. Nous n'avons jamais réussi à obtenir un contour parfait, ce qui montre bien la nécessité d'améliorer le snake autonome.

D'après le relevé ci-dessus, notre algorithme semble apporter une forte amélioration dans la détection des contours car dans 15 cas sur 20, le contour est bien suivi. Nous obtenons en moyenne le contour dès l'itération 30, en mode local, global ou suivant la variation d'énergie.

De plus, la probabilité d'obtenir le bon contour est satisfaisante car, avec la modification locale, nous avons obtenu dans quatre cas 100% de réussite. Cette remarque peut aussi être faite à la modification suivant la variation d'énergie puisque dans tous les cas les 100 % de réussite ont été atteint.

Le nombre d'itérations moyen nécessaire à la capture du contour avec un snake autonome pourvu de notre algorithme, reste toutefois assez proche du nombre d'itération

moyen avec un snake à coefficients constants. Mais il faut remarquer que dans ces différents essais nous n'avons pas passé 30 minutes à régler les coefficients au préalable.

Comme l'image du Losange est une image totalement réalisée par ordinateur, nous ne pouvons établir de généralités sur ce seul type d'image, et nous avons donc réalisé les mêmes essais sur une image réelle : le condyle.

6.2. Le condyle

Le condyle est une image assez difficile à traiter, les résultats obtenus avec le contour actif à coefficients constants n'ont pas été satisfaisants, nous avons donc choisi cette image pour l'étude de notre amélioration du contour actif autonome.

Paramètres initiaux

Image **Condyle224x224.bmp**
Paramètres

X	Y	rayon	Points	Voisinage	Normalisation
112	111	75	100	3	(E-Emin)/(Emax-Emin)

Coefficients de départ

Intensité	Courbure	Continuité	Gradient	Ballon
1	1	1	1	-1

Critère d'arrêt : AUCUN car ils ne fonctionnent pas avec le snake autonome
Calcul des coefficients de corrélation sur 10 itérations

Modification locale									
Energie prépondérante	1	1,1	1,1	0,9	0,9	1,1	1,1	0,9	0,9
Energie la moins suivie	1	1	1	1	1	0,9	0,9	1,1	1,1
Paliers (10 itérations)	sans	sans	avec	sans	avec	sans	avec	sans	avec
Contour respecté	MOYEN	OUI (3/4)	OUI (2/4)	MOYEN	OUI (3/4)	NON	OUI (2/4)	NON	MOYEN
Nombre d'itérations	100	62	70	70	65	100	90	100	60
Amélioration		**OUI**	**OUI**		**OUI**		**OUI**		

Modification globale									
Energie prépondérante	1	1,1	1,1	0,9	0,9	1,1	1,1	0,9	0,9
Energie la moins suivie	1	1	1	1	1	0,9	0,9	1,1	1,1
Paliers (10 itérations)	sans	sans	avec	sans	avec	sans	avec	sans	avec
Contour respecté	MOYEN	NON	MOYEN	NON	MOYEN	NON	MOYEN	NON	NON
Nombre d'itérations	100	100	90	100	100	100	90	100	100
Amélioration									

Modification locale suivant la variation d'énergie					
Coefficient multiplicateur de la variation d'énergie	0	0,1	0,1	-0,1	-0,1
Paliers (3 itérations)	sans	sans	avec	sans	avec
Contour respecté	MOYEN	OUI (3/4)	OUI (3/4)	MOYEN	MOYEN
Nombre d'itérations	100	85	100	80	80
Amélioration		**OUI**	**OUI**		

Tableau 8. Résultats des tests sur le condyle, avec un snake autonome

Conclusion

Les résultats de nos essais sur le condyle sont assez prometteurs pour les images réelles. En effet, alors que nous n'avions pas obtenu la détection parfaite des contours du condyle avec le snake autonome seul, nous parvenons à capter le contour en seulement 62 itérations (en moyenne !) dans le meilleur des cas.

La modification globale des coefficients d'énergie est néanmoins bien moins satisfaisante puisque tous les tests n'ont pas été convaincants. Sur une image comme le condyle, il est très important de rester avec des points totalement autonomes car les énergies influentes diffèrent beaucoup suivant le point étudié.

Les meilleurs taux de réussite sont présents dans la modification suivant la variation d'énergie. Les 100 % ne sont toutefois pas atteint.

6.3. Synthèse des résultats

La modification des coefficients d'énergie sur le snake autonome apporte bien une amélioration. Sur des images artificielles, notre algorithme s'avère efficace puisque le nombre moyen d'itérations avec un snake autonome (qui a donc des coefficients d'énergie aléatoires) est très proche du nombre moyen d'itérations avec des coefficients constants.

De plus, le comportement du snake autonome sur une image réelle comme le condyle, est plus que prometteur car, rappelons-le, la modification des coefficients d'énergie avec un contour actif à coefficients constants (non-aléatoires serait maintenant le terme exact), n'avait aucunement améliorer le snake.

De meilleurs résultats pourraient être envisagés car nous avons réalisé les tests avec des valeurs de 1,1 et 0,9 pour l'adaptation des coefficients en mode local et global, et avec ± 0,1 suivant la variation d'énergie. Il est en effet probable de réduire le nombre d'itérations en ajustant ces nouveaux paramètres.

Enfin, un autre point intéressant de l'algorithme est le taux de réussite des essais. Contrairement au contours actifs à coefficients non-aléatoires, le snake autonome ne donne

jamais le même résultat, il peut parfois se coller au contours d'une image et parfois non. Mais dans certains cas de la modification des coefficients, nous obtenons à chaque fois le même résultat, bon, ce qui est très avantageux.

Voici une simple comparaison entre le snake autonome seul (à gauche) et le snake autonome avec notre algorithme (à droite).

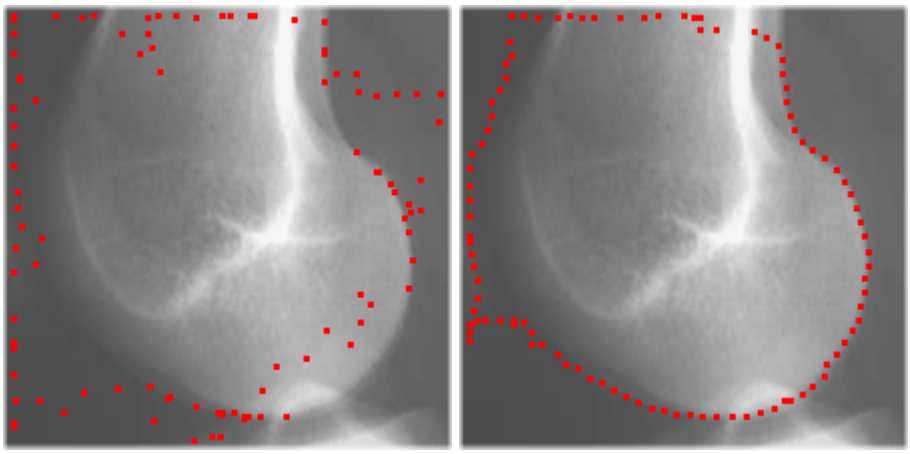


Figure 36. Le snake autonome seul et avec notre algorithme

Cela montre bien que la modification des coefficients d'énergie suivant leur influence sur le snake apporte bel et bien une amélioration, mais elle a besoin d'être encore améliorée.

Conclusion

Dans ce travail, nous nous sommes intéressés à une technique issue de la théorie de l'apprentissage pour améliorer les contours actifs. Celle-ci a fourni de bons résultats lorsque nous l'avons utilisée sur un snake à coefficients constants, pour des images artificielles comme le losange. En effet, dans le meilleur des cas nous obtenons une amélioration de 30 % du nombre d'itérations. Mais, en ce qui concerne les images réelles, la méthode s'est avérée plus néfaste que bénéfique.

Avec le contour actif autonome, nous obtenons cependant de bien meilleurs résultats avec la modification des coefficients d'énergie, que sans celle-ci. Dans de nombreux cas, les contours de l'image traitée sont bien suivis, pour des images synthétiques ou réelles, et avec une probabilité qui semble être correcte.

Cette méthode apporte aussi quelques désavantages, avec notamment, une forte augmentation de l'espace de mémoire nécessaire au traitement de l'image (la détermination de l'énergie la plus suivie requière nombre de données à stocker), et deux à trois nouveaux paramètres à régler en plus. Néanmoins, après avoir réalisé nombre d'essais de l'algorithme, nous pouvons dire que le meilleur jeu de paramètres de l'algorithme de l'amélioration est une augmentation locale de 1,1 (donc de 10 %) par paliers, de l'énergie directrice.

Pour encore améliorer cette technique, nous pourrions peut-être augmenter chaque énergie suivant leur influence dans la décision du snake sur le déplacement, et non plus augmenter seulement la première ou la dernière énergie.

Le stage m'a permis aussi d'apprendre beaucoup de choses sur la programmation orientée objet, sur le langage C++, les contours actifs biensûr, mais principalement sur le traitement d'image en général. J'ai aussi appris à être de plus en plus autonome vis-à-vis de mon travail, et à essayer de m'imposer une certaine rigueur, la rigueur d'un scientifique, que je n'avais pas encore acquise au cours de l'IUT.

Résumé

Le contour actif, appelé aussi snake, est employé pour détecter les contours d'une image. Il est composé de multiples énergies externes, internes et de contexte, qui permettent au snake de ce déplacer dans l'image et de s'arrêter sur un contour. Dans le cadre du stage, nous avons utilisé l'algorithme Greedy qui utilise cinq énergies pondérées : l'énergie de courbure, d'intensité, de continuité, de gradient et de ballon. Le déplacement s'effectue en comparant les différentes fonctionnelles d'énergies des pixels voisins d'un point du snake.

Dans l'algorithme Greedy initial, la pondération des énergies était statique et nous avons décidé de changer cela et de rendre dynamique cette pondération, suivant l'influence de chaque énergie. Nous avons dans un premier temps essayé de modifier les coefficients à l'échelle locale, c'est-à-dire que nous avons pour chaque point déterminé quelle était l'énergie la plus suivie, puis nous avons modifié le coefficient de celle-ci. Ensuite, nous avons réalisé le même travail mais de façon globale, en modifiant le même coefficients pour tous les points du snake. Enfin, au lieu de modifier les coefficients d'énergie avec un pourcentage fixe, nous avons entrepris de modifier les coefficients avec un pourcentage de la variation d'énergie d'un même point, entre deux itérations.

Pour valider l'efficacité de notre algorithme, de nombreux tests ont été réalisés, sur différents types d'images (artificielles et réelles), avec différents pourcentages d'augmentation (± 10 % etc.), et sur deux types de contours actifs : le contour actif à coefficients d'énergie constants, et le snake autonome.

Environ 249 mots

Abstract

Active contours, called snake, is used to detect image's contours. It is based on several energies, externs, interns and contexts energies, which make snake move into images and make it stop if it detect a contour. The Greedy algorithm we used, is splited on five ponderated enregies : curve, intensity, continuity, gradient and balloon energy. The move is determined comparing differents fonctionnels of energy between all neighborhood of pixels of each point in a snake.

In the Greedy algorithm alone, the ponderation of energies is static and we decided to change this and make it dynamic, according to influence of each energie. The first step was determine which energie was the more chosen. Then, we modify the coefficient of this more followed energie according each point for the local method and according all of points for the global method. Moreover, we modify coefficients in two differents ways : either we add a constant percentage (ex 10 %), or we add a percentage depending of the variation of energy.

We check our algorithm in differents ways, with two kinds of images (real and artificial), with several percents of augmentation, and especially with two kinds of actives contours : the active contour with constants coefficients, and with the Autonomous Snake.

Index des mots clés

Table des illustrations

Figures

Tableaux

Equations

CP : créations personnelles.

Bibliographie

- **Œuvres, revues et thèses :**

  Thèse de M. Mraghni, « Détection de chaînes de contours dans une image numérique par approche symbolique et par grammaire de formes », 1997

  Thèse de J.J. Rousselle, « Les contours actifs, une méthode de segmentation », Juillet 2003

- **sites Internet et documents électroniques :**

  Thèse de J.J. Rousselle - « Les contours actifs, une méthode de segmentation »
  http://rfai.li.univ-tours.fr/rfai/rapports/rou03a.pdf - consulté le 10/04/2004.

  « La librairie FoxToolkit »
  www.fox-toolkit.org - consulté le 20/04/2004.

  « Un modèle de contour actif pour le suivi rapide d'objets en movement »,
  http://rfai.li.univ-tours.fr/rfai/rapports/lef01d.pdf - consulté le 20/04/2004.

  « A web oriented recurrent neural network simulator »,
  www-rocq.inria.fr/~crucianu/src/iconib98.pdf- consulté le 20/04/2004.

  « La librairie ImageMagick »
  www.ImageMagick.org - consulté le 20/04/2004.

  Nicolas Sorel - « Codes Sources en C++»,
  www.cppfrance.com - consulté le 20/04/2004.

  « Laboratoire d'informatique de Tours »
  www.li.univ-tours.fr - consulté le 20/04/2004.

  J.J. Rousselle - « Remarques sur la présentation des rapports »
  http://www.rfai.li.univ-tours.fr/rousselle/docum/pdf/Rqmemo02.pdf - consulté le 27/05/2004.

Sommaire des annexes

Annexe 1

Aperçu des données écrites dans un fichier Excel pour y être traitées

Coordonnées du cercle de départ :

Rayon	Centre X	Centre Y
123	128	128

Moving Actif
Normalisation : Maximum
Critère d'arret n°1 Valeur : 20
Line search Inactif Valeur : 0
Momentum Inactif Valeur : 0
Coefficients constants :

Courbure	Continuité	Gradient	Ballon
0,5	0,5	0,5	0,5

Itération	Num Point	Courbure		Continuité		Gradient		Ballon		Déplacement	
		dx	dy	dx	dy	dx	dy	dx	dy	dx	dy
1	1	-1	0	2	0	3	3	-3	0	-3	1
1	2	1	1	-2	0	3	3	-3	-3	-1	1
1	3	-1	0	3	-1	3	3	-3	-3	0	0
1	4	1	0	-1	-1	3	3	-3	-3	-1	-1
1	5	-1	1	-2	0	3	3	-3	-3	-3	1
1	6	1	0	0	-1	3	3	-3	-3	-1	-1
1	7	0	0	-1	0	3	3	-3	-3	-2	-1
1	8	0	0	0	0	3	3	-3	-3	-2	-1
1	9	0	0	1	0	3	3	-3	-3	-2	-2
1	10	0	1	1	1	3	3	-3	-3	-2	-1
1	11	0	0	1	0	3	3	-3	-3	-2	-2
1	12	0	0	2	1	3	3	-3	-3	-1	-1
1	13	0	0	2	1	3	3	-3	-3	-1	-2
1	14	0	0	2	2	3	3	-3	-3	-2	-1
1	15	0	0	3	2	3	3	-3	-3	-1	-2
1	16	0	0	3	2	3	3	-3	-3	-2	-2
1	17	1	0	0	-1	3	3	-3	-3	-1	-2
1	18	0	0	3	3	3	3	-3	-3	-1	-2
1	19	0	0	0	-1	3	3	-3	-3	-1	-2
1	20	0	0	0	0	3	3	-3	-3	-1	-2
1	21	0	1	0	1	3	3	-3	-3	-1	-1
1	22	1	-1	1	0	3	3	-3	-3	1	-3
1	23	0	1	0	2	3	3	-3	-3	-1	-1

etc…

Et voici le résultat du traitement des différentes données mentionnées ci-dessus.

COEFFICIENTS DE CORRELATION

Courbure	Continuité	Gradient	Ballon
0,98873188	0,24902142	0,06059258	0,46796626

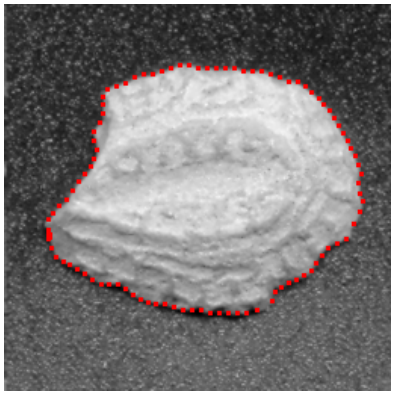
DISTANCES EUCLIDIENNES normalisée (/RACINE(72))

Courbure	Continuité	Gradient	Ballon
0,15140926	0,31958281	0,49387699	0,44177461

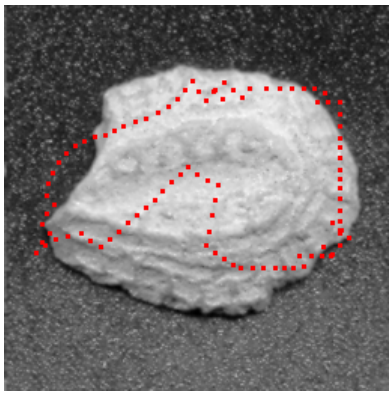
Source : Création personnelle

Annexe 2

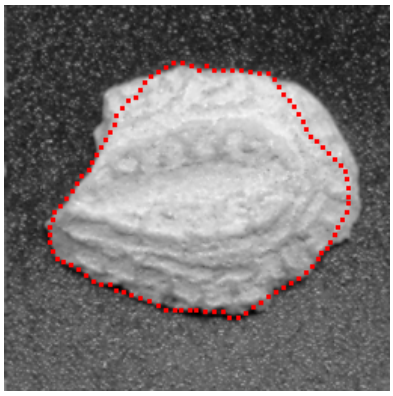
Résultats des tests effectués sur le CauryBroken, selon une modification locale des coefficients.



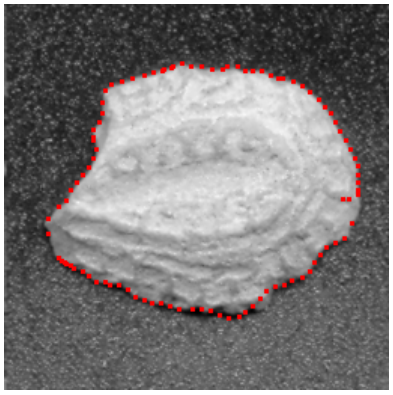
Cas n°0



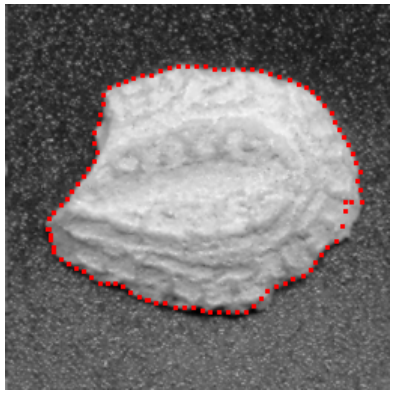
Cas n°1



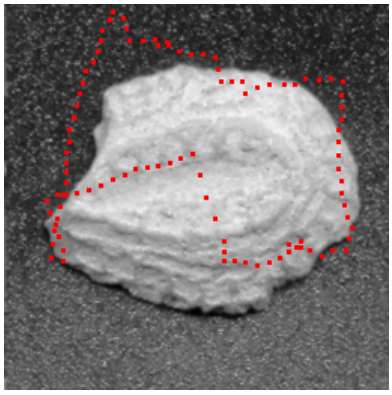
Cas n°2



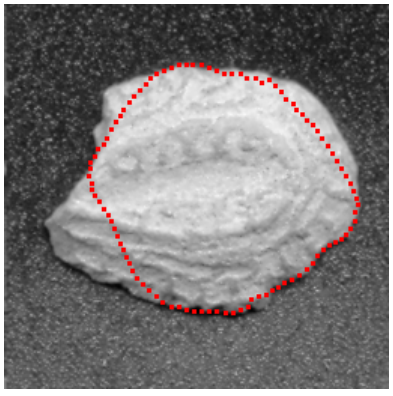
Cas n°3



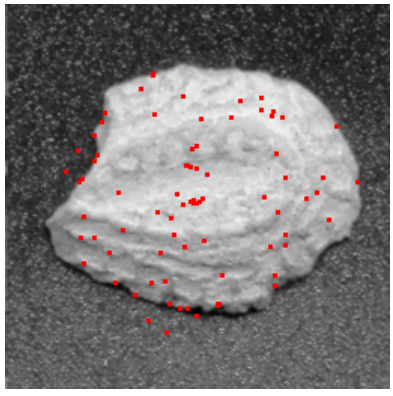
Cas n°4



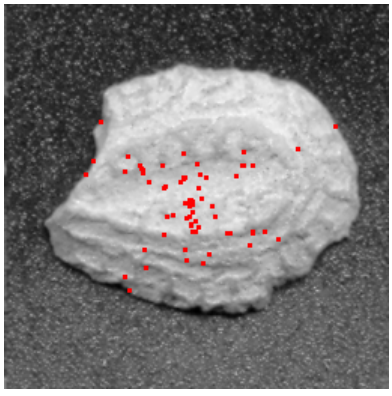
Cas n°5



Cas n°6

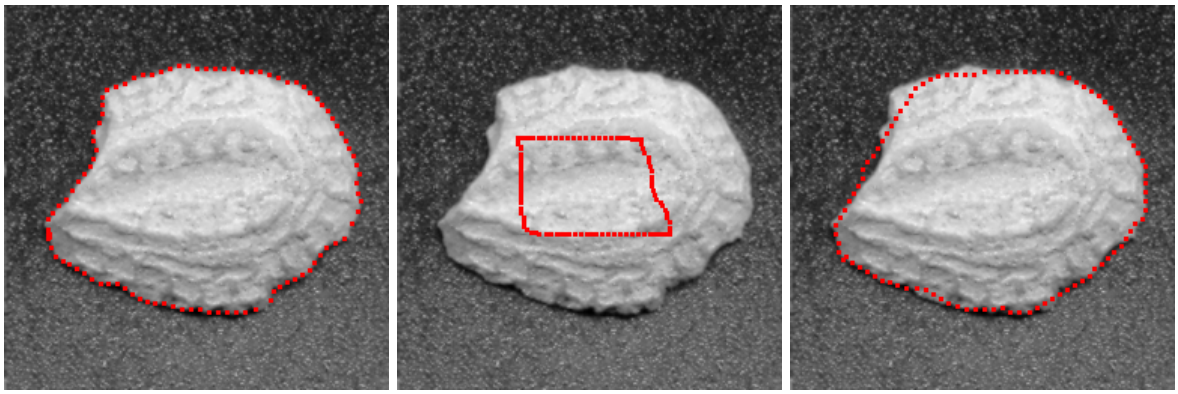


Cas n°7

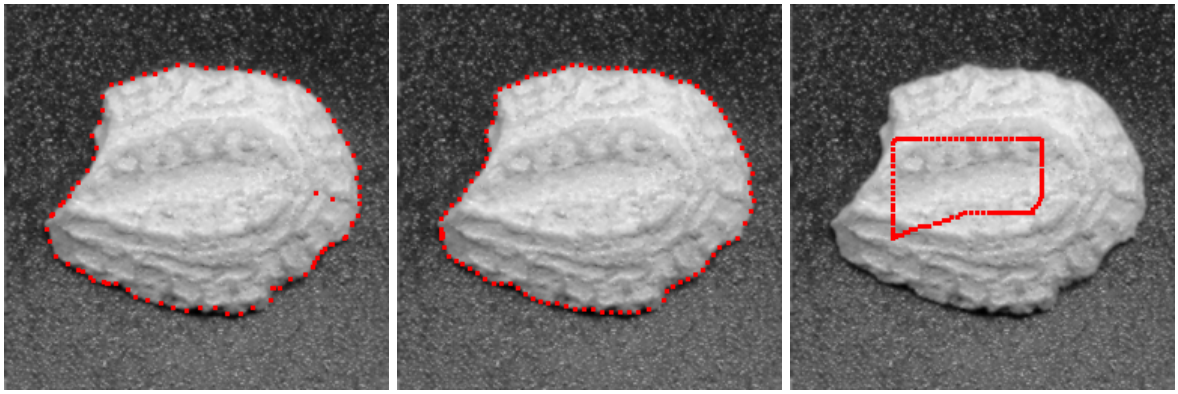


Cas n°8

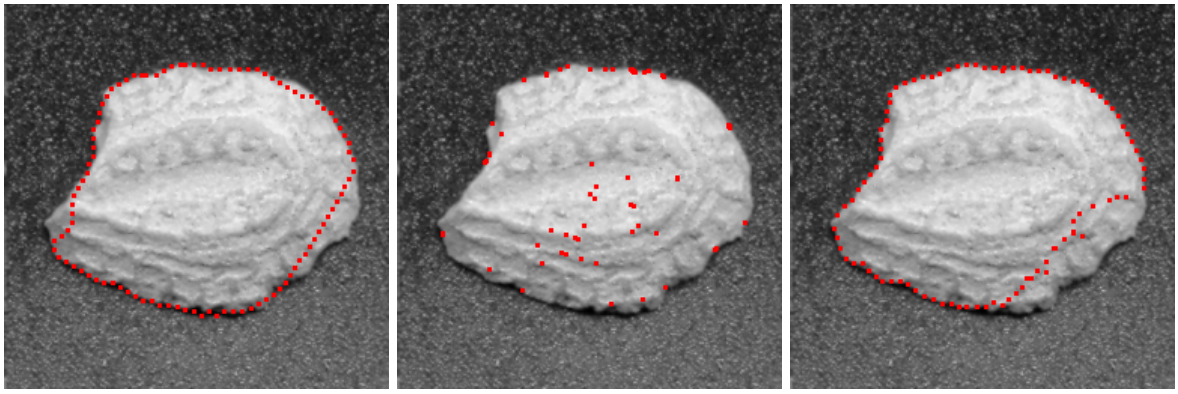
Résultats des tests effectués sur le CauryBroken, selon une modification globale des coefficients.



Cas n°0 Cas n°1 Cas n°2



Cas n°3 Cas n°4 Cas n°5



Cas n°6 Cas n°7 Cas n°8

Source : Création personnelle

Annexe 3

INSTALLATION D'IMAGE MAGICK POUR WINDOWS 98, NT, 2000 et XP POUR VISUAL C++ 6.0 : MAGICK++

Disposer du fichier ImageMagick.zip (41523Ko) ou bien télécharger :	ftp://ftp.simplesystems.org/pub/ImageMagick/ImageMagick-5.3.0.tar.gz
Décompresser le fichier dans un répertoire du type	c:\ImageMagick
Nous obtenons l'arborescence suivante :	un répertoire c:\ImageMagick contenant les répertoires : bzlib, coders, contrib, ...
Ouvrir le projet configure.dsw dans Visual C++ 6.0, le compiler et l'exécuter	c:\ImageMagick\VisualMagick\configure\configure.dsw
Pour chaque fenêtre du programme de configuration, laisser les paramètres proposés par défaut	Cliquer sur suivant, suivant...
Ouvrir le projet VisualDynamicMT.dsw dans Visual C++ 6.0, le compiler en mode DEBUG (menu Build->Set Active Configuration). Cette opération est assez longue (environ 10-15 min)	c:\ImageMagick\VisualMagick\VisualDynamicMT.dsw
Dans Visual C++ 6.0, aller dans le menu :	tools->options, onglet directories
Cette étape configure Visual C++ 6.0 en lui spécifiant les chemins afin d'utiliser ImageMagick	Les chemins suivants sont indiqués dans la continuité des exemples précédents. Ils doivent être adaptés si ImageMagick n'a pas été installé en c:\ImageMagick . Ne pas retaper les chemins à la main, utiliser le bouton "..." pour parcourir l'arborescence.
Dans include files, ajouter les chemins :	C:\IMAGEMAGICK\MAGICK++\LIB C:\IMAGEMAGICK
Dans library files, ajouter les chemins :	C:\IMAGEMAGICK\VISUALMAGICK\BIN C:\IMAGEMAGICK\VISUALMAGICK\LIB
Dans executable files, ajouter les chemins :	C:\IMAGEMAGICK\VISUALMAGICK\BIN C:\IMAGEMAGICK\VISUALMAGICK\LIB
Si l'installation d'ImageMagick a lieu sous Windows 98 ou Windows NT, ajouter au fichier c:\autoexec.bat le chemin:	%PATH%;c:\ImageMagick\VisualMagick\bin;

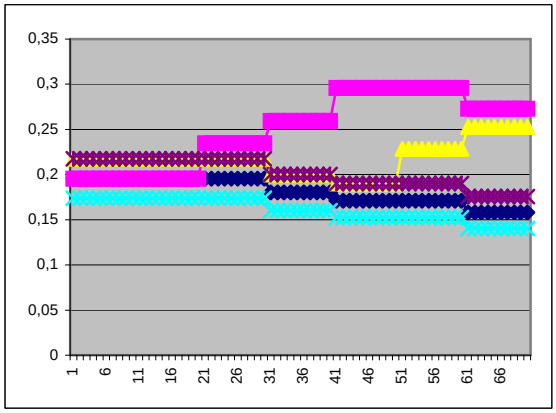
Si l'installation d'ImageMagick a lieu sous Windows XP, aller dans le panneau de configuration, performance et maintenance, système, avancé et variables d'environnements puis modifier la variable PATH en y ajoutant le premier chemin indiqué ci-contre : Créer ensuite une nouvelle variable d'environnement MAGICK_HOME avec la valeur du second champ indiqué ci-contre : Suivre la même procédure pour Windows 2000 sauf que système se trouve directement dans le panneau de configuration	%PATH%;c:\ImageMagick\VisualMagick\bin; c:\ImageMagick\VisualMagick\bin
L'installation est terminée. Il est conseillé de redémarrer l'ordinateur pour s'assurer de la prise en compte des nouvelles variables d'environnement	Pour obtenir de l'aide sur l'utilisation d'ImageMagick, ouvrir le fichier c:\ImageMagick\ImageMagick.html

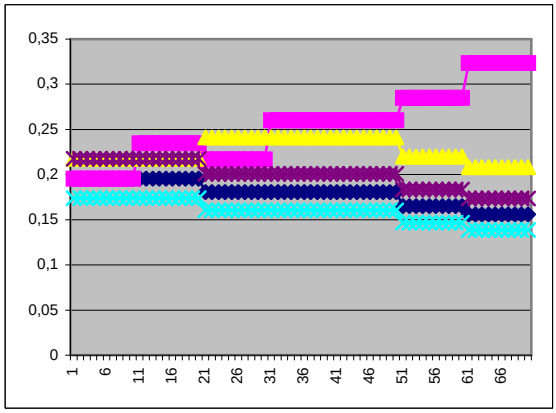
Source : www.ImageMagick.com

Annexe 4

Visualisation des différents modes d'adaptation des coefficients d'énergie

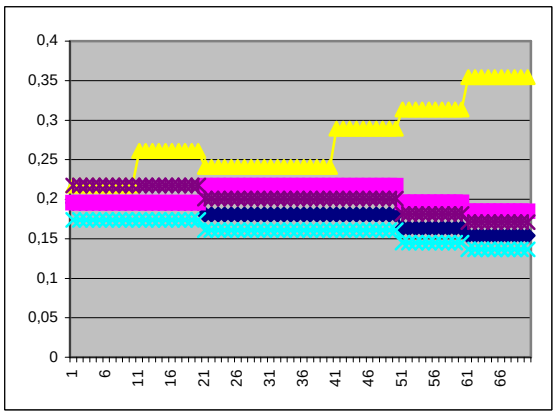
1. Augmentation locale de 20 % par paliers de 10 itérations :

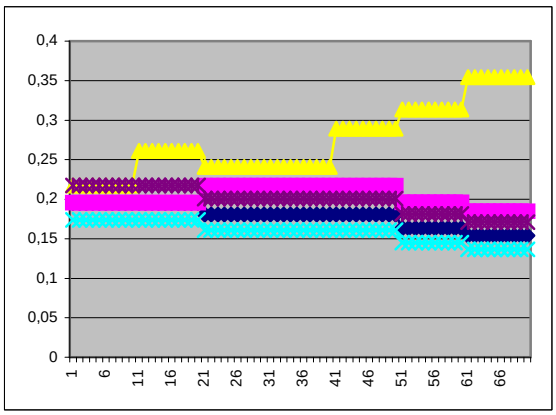




Point N° 28 et 29

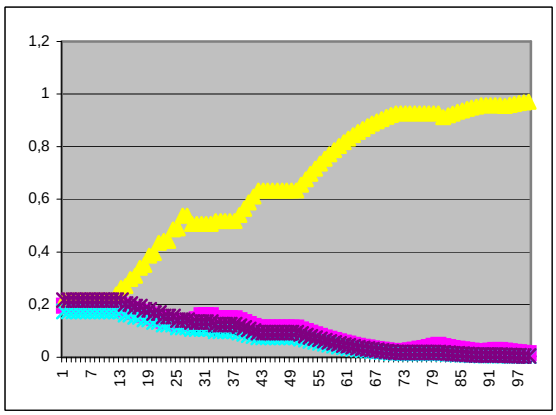
2. Augmentation globale de 20 % par paliers de 10 itérations :

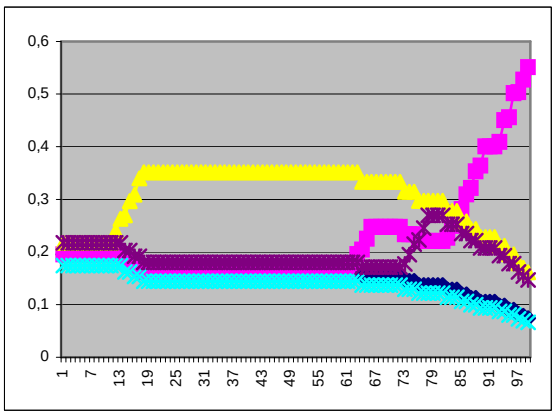




Point N° 28 et 29

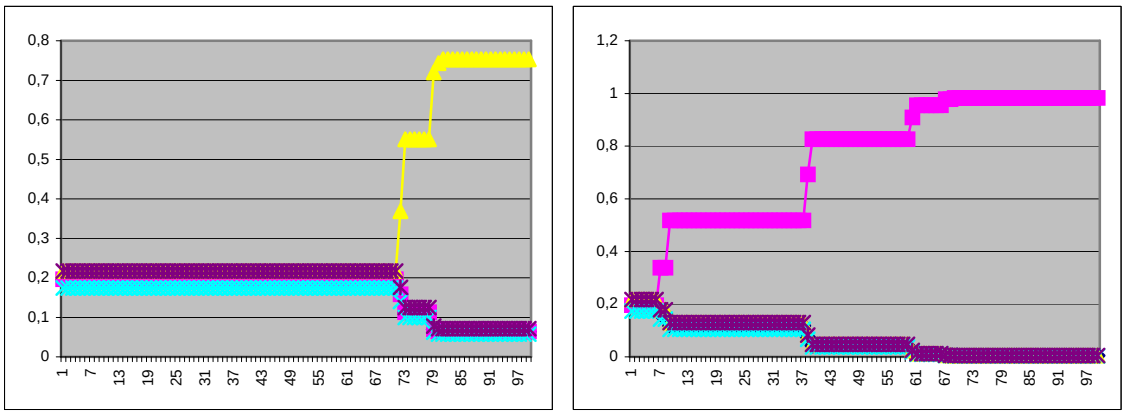
3. Augmentation locale de 10 % (calcul sur 10 itérations) sans palier :





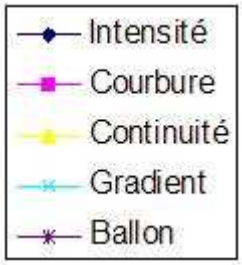
Point N° 28 et 29

4. Augmentation locale suivant la variation d'énergie (calcul sur 10 itérations) :



Point N° 28 et 29

Légende :



Tests effectués avec une normalisation des coefficients (la somme des valeurs absolues doit tendre vers 1).

Source : Création personnelle