الجمهورية الجزائرية الديمقراطية الشعبية
Republic of Algeria Ministry Democratic and Popular
وزارة التعليم العالي و البحث العلمي
of Higher Education and Scientific Research

University of Mohamed Khider- Biskra

Faculty of Science & Technology
Departement of Electrical Engineering
Sector: Electronic

Option : Telecommunication

Ref:.....

Memory End of Studies
For graduation

MASTER

Theme

Active Detection of Deformable Contours in Different Shapes for MR Images by Using B-Snake Model

Presented by:
Bougourzi Fares

Before the jury:

Mrs. Fedias Meriem	**Presented**
Dr. El Kcourd Kaouther	**Supervisor**
Mrs. Mdaoukh Sadia	**Examiner**

Academic Year: 2012 / 2013

الجمهورية الجزائرية الديمقراطية الشعبية
Republic of Algeria Ministry Democratic and Popular
وزارة التعليم العالي و البحث العلمي
of Higher Education and Scientific Research

University of Mohamed Khider- Biskra

Faculty of Science & Technology
Departement of Electrical Engineering
Sector: Electronics
Option：Telecommunication

MASTER

Theme

Active Detection of Deformable Contours on Different Shapes at RM Images by Using B-Snake Model

Presented by:

Bougourzi Fares

Favorable opinion of the supervisor:

Dr. El Kourd Kaouther *Signature*

Favorable opinion of the President of the Jury

Mrs. Fedias Meriem *Signature*

Stamp and signature

الجمهورية الجزائرية الديمقراطية الشعبية
Republic of Algeria Ministry Democratic and Popular
وزارة التعليم العالي و البحث العلمي
of Higher Education and Scientific Research

University of Mohamed Khider- Biskra

Faculty of Science & Technology
Departement of Electrical Engineering
Sector: Electronics

Option : Telecommunication

MASTER

Theme

Active Detection of Deformable Contours in Different Shapes for MR Images by Using B-Snake Model

Presented by: Bougourzi Fares

Supervisor by : Dr. El Kourd Kaouther

Abstract: Detection of the boundary of an object in medical images, then the segmentation from the background class, is a major challenge in medical image processing. Our work present an adaptive B-Snake model for object contour extraction. A cubic B-snake model is developed for extracting 2D deformable objects from medical images, with an adaptive control points insertion algorithm that is suggested to increase the flexibility of B-Snake to describe complex shape and different. And we using different energy function B-Snake (GVF(s)) is able not only to grow through long and thin concavities but also it does not miss the correct boundaries. Moreover, this algorithm avoids computationally expensive optimization stages with used just the external force (GVF). That is what is known as the B-Snake.

ملخصّ: تحديد حدود التشوه في الصور الطبية من الاساليب الاساسية في تجزئة التشوه من الخلفية و يشكل تحديا كبيرا في معالجة الصور الطبية. وفي عملناهذا قدمنا تكيف نظرية B-Snake مع B-Spline لاستخراج حدود التشوه. تم تطوير نموذج B-Snake مكعب لاستخراج كائنات تشوه 2D من الصور الطبية، مع نقاط السيطرة على التكيف خوارزمية الإدراج الذي اقترح لزيادة مرونة B-Snake لوصف شكل معقد من أشكال مختلفة.ولقد استعملنا معادلات مختلفة للطاقة ((GVF(s) هي قادرة ليس فقط على النمو من خلال التقعر طويلة ورقيقة ولكن أيضا لا يغيب عن الحدود الصحيحة. وعلاوة على ذلك، هذه الخوارزمية يتفوق من خلال تجنب عمليات مكلفة حسابيا وذلك باستعمال القوة الخارجية فقط (GVF).وذلك ما يعرف ب B-Snake.

Dedication

I would like to dedicate this humble work to my beloved mother and my generous father

To my dear second mother

To my lovely little sister, and all my venerable brothers

To my dear supervisor which was like a real mother

Who really helped and guided me in a very tough times through her experience tenderness and motivation

A special dedication to all my teachers that thought me in my life since the beginning

To all my dear friends and any person who helped and supported me during conducting this work

Thank’s

I would like to thank my Supervisor ***Dr. Elcourd Kaouther***, for her guidance. She has shown me what the role model of educator. Although she already has the experience of more than 17 years, she is always willing to learn new knowledge, and I am the beneficiary of the knowledge. She is not only a good teacher but also a good mother. I would thank **Dr. Bakhouche, Dr.Ouafi** and my friend **Salah Edinne** for their help, and my jury Mrs. **Fedias Meriem** the Presented and Mrs. **Mdaoukh Sadia** the Examination.

Abstract

Abstract

Detection of the boundary of an object in medical images, then the segmentation from the background class, is a major challenge in medical image processing.

Our work present an adaptive B-Snake model for object contour extraction. A cubic B-snake model is developed for extracting 2D deformable objects from medical images, with an adaptive control points insertion algorithm that is suggested to increase the flexibility of B-Snake to describe complex shape and different.

And we using different energy function B-Snake (GVF(s)) is able not only to grow through long and thin concavities but also it does not miss the correct boundaries. Moreover, this algorithm avoids computationally expensive optimization stages with used just the external force (GVF). That is what is known as the B-Snake.

Keywords: Object Boundary, Energy Function, B-Snake, B-Spline, GVF, 2D.

ملخص

تحديد حدود التشوه في الصور الطبية من الاساليب الاساسية في تجزئة التشوه من الخلفية و يشكل تحديا كبيرا في معالجة الصور الطبية.

وفي عملناهذا قدمنا تكيف نظرية B-Snakeمع B-Spline لاستخراج حدود التشوه. تم تطوير نموذج-B Snake مكعب لاستخراج كائنات تشوه 2D من الصور الطبية، مع نقاط السيطرة على التكيف خوارزمية الإدراج الذي اقترح لزيادة مرونة B-Snake لوصف شكل معقد من أشكال مختلفة.

.ولقد استعملنا معادلات مختلفة للطاقة ((GVF(s) هي قادرة ليس فقط على النمو من خلال التقعر طويلة ورقيقة ولكن أيضا لا يغيب عن الحدود الصحيحة. وعلاوة على ذلك، هذه الخوارزمية يتفوق من خلال تجنب عمليات مكلفة حسابيا وذلك باستعمال القوة الخارجية فقط (GVF).وذلك ما يعرف ب B-Snake.

كلمات دلالية: حدود التشوهة ,معادلات الطاقة ,B-Snake, B-Spline, 2D,GVF.

Résumé

Détection de la frontière d'un objet dans les images médicales et de segmentation à partir de la classe de base est un défi majeur dans le traitement d'images médicales.

Notre travail présente un modèle B-Snake adaptative pour l'extraction de contour de l'objet. Un modèle B-serpent cube est conçu pour extraire des objets déformables 2D à partir

Abstract

d'images médicales, avec un algorithme d'insertion des points de contrôle adaptatif qui est suggéré d'augmenter la flexibilité de B-Snake pour décrire une forme complexe et différent.

Et nous utilisons la fonction d'énergie différent B-Snake (GVF (s)) est capable non seulement de croître grâce à concavités longues et fines, mais aussi il ne rate pas les limites correctes. Par ailleurs, cet algorithme permet d'éviter les étapes d'optimisation onéreuse en calcul utilisée avec juste la force extérieure (GVF). C'est ce qui est connu comme le B-Snake.

Mots-clés: limite de l'objet, la fonction d'énergie, B-Snake, B-Spline, GVF, 2D.

List of Abbreviations

- MRI : Magnetic Resonance Imaging
- MRF: Markov Random Field.
- MFA: *Mean Field Annealing*.
- SA: Simulated Annealing.
- MR: Magnetic Resonance.
- GVF: *Gradient Vector Flow*.
- GGVF: Generalized Gradient Vector Flow.
- GVD: Gradient Vector Diffusion.
- PGVF: Poisson Gradient Vector.
- EPGVF: Edge Preserving Gradient Vector Flow.
- VFC: Vector Field Convolution.
- MMSE: Minimum Mean Square Error.
- 2D: 2 Dimentions.
- 3D: 3 Dimentions.

List of Figures

List of Figures

List of Figures

List of Tables

Summary

Summary

Introduction

Many works in the literature had as main goal to improve the snake model in its representation and its performances. Amini et al, [1] proposed dynamic programming (OP) algorithm to guide the snake curve throughout the minimum of energy, later, Kang et al, in [2] presented an enhanced OP algorithm. An alternate improvement was introduced by Menet et al. [3], the B-snake, and it consists in representing the curve by cubic B - splines, with his model they have improved the stability and the convergence speed of the contour. Kass [4] introduced Active Contour Model, or Snake, which is a deformable model for approaching object boundaries.

Active Contour Model, or Snake, which is a deformable model for approaching object boundaries, is a curve defined within an image domain which can move under the influence of internal forces from the curve itself and external forces from the image data. Once internal and external forces have been defined, the Snake can detect the desired object boundaries (or other object features) within an image. From the original philosophy of Snake, many techniques have been proposed as an alternative method for presenting a curve: Fourier descriptors [5], B-Splines [6, 7], auto-regressive model [8], moments [9], HMM models, and wavelets, etc. Among these methods, B-Spline representation of the curve stands for an advantage as such a formulation of a deformable model allows for the local control and a compact representation.

The important of the method of b-spline guide us to choose this subject and learn how the flexibility of the curve increases as more control points are added while it still has a small number of parameters compared to other model, Moreover, the internal force is not needed as the smoothness requirement has been implicitly built into the model.

The project, present an adaptive B-Snake model for object contour extraction. A cubic B-Snake model is developed for extracting 2D deformable objects from medical images, with an adaptive control point insertion algorithm that is suggested to increase the flexibility of B-Snake to describe complex shape. This method overcomes the problems that exist in other B-spline based model that have to decide beforehand or exhaustively search over a range of value for the number of control points. Hence, these methods are less flexible to describe unknown complex shapes. A minimum energy method which we called Minimum Mean Square Error (MMSE) is proposed for B-Snake to push it to the target boundary. The internal forces are not required in deforming B-Snake since the representation of B-Spline has guaranteed smoothness by hard implicit constraints. The proposed B-Snake model has been tested on object contour extraction

such as human brain ventric1e in Magnetic Resonance (MR) images. The experimental results demonstrate the capability of adaptive shape description and object contour extraction.

Our work divided into five chapters as the following:

Chapter 1: Definition of Segmentation and Active Contours

Chapter 2: Background of B-Spline method (to Access to cubic B-spline).

Chapter 3: Energy of image (Focus External Force GVF and present different forms), and study of different models deformable.

Chapter 4: Presentation of B-snake model.

Chapter 5: Conception and results.

In this work we faced difficulties to conversion the mathematic equations to real ideas then translate them to program, the difficulty to get some references important and the main difficulty was in calculate the GVF which are complex and it have a lot of cases , we like to note that we used the documents, we have chosen two titled:

- **"**Object Contour Extraction Using Adaptive B-Snake Model", by YUE WANG AND EAM KHWANG TEOH.

- "Deformable Models with Application to Human Cerebral Cortex Reconstruction from Magnetic Resonance Images", by Chenyang Xu.

CHAPTER I

SEGMENTATION AND ACTIVE CONTOURS

I.1.Introduction

Vision is the most advanced sense among the five senses of human beings, and plays the most important role in human perception. Although the sensitivity of human vision is limited within the visible band, imaging machines can operate on the images generated by sources that human vision cannot associate with. Thus, *machine vision* encompasses a wide and varied field of applications, even in areas where human vision cannot function, e.g. infrared (IR), ultraviolet (UV), X-ray, magnetic resonance imaging (MRI), ultrasound.

Although there is no clear distinction among image processing, image analysis, and computer vision, usually they are considered as hierarchies in the processing continuum. The low-level processing, which involves primitive operations such as noise filtering, contrast enhancement, and image sharpening, is considered as *image processing.* Note both its inputs and outputs are images. The mid-level processing, which involves segmentation and pattern classification, is considered as **image analysis** or image understanding. Note its input generally are images, but its outputs are attributes extracted from those images, e.g. edges, contours, and the identity of individual objects, called *class.* The high-level processing, which involves 'making sense' of an ensemble of recognized objects and performing the cognitive functions at the far end of the processing continuum, is considered as *computer vision*. We discuss the technologies used in the image analysis, and propose novel segmentation methods through this project.

I.2.Image Segmentation: background

There are two main approaches in image segmentation: *edge*- and *region*- based. Edge-based segmentation partitions an image based on discontinuities among sub-regions, while region-based segmentation does the same function based on the uniformity of a desired property within a sub-region. **In** this chapter, we briefly discuss existing image segmentation technologies as background.

I.2.1 Edge-based Segmentation

Edge-based segmentation looks for discontinuities in the intensity of an image.It is more likely edge detection or boundary detection rather than the literal meaning of image segmentation. An *edge* can be defined as the boundary between two regions with relatively distinct properties. The assumption of edge-based segmentation is that every sub-region in an image is sufficiently uniform so that the transition between two sub-

regions can be determined on the basis of discontinuities alone. When this assumption is not valid, region-based segmentation, usually provides more reasonable segmentation results.

Basically, the idea underlying most edge-detection techniques is the computation of a local derivative operator. The *gradient vector* of an image *I(x, y)*, given by

$$\nabla I = \begin{bmatrix} \partial I/\partial x \\ \partial I/\partial y \end{bmatrix} : \Omega \to \mathfrak{R}^2 \qquad \textbf{(Eq I.1)}$$

is obtained by the partial derivatives $\partial I/\partial x$ and $\partial I/\partial y$ *at* every pixel location. The local derivative operation can be done by convolving an image with kernels shown in Fig I.1.

-1
0
1

(a)

-1	0	1

(b)

Fig I.1: Examples of gradient kernels along: (a) vertical direction, (b) horizontal direction[1].

The magnitude of the first derivative

$$|\nabla I| = \sqrt{(\partial I/\partial x)^2 + (\partial I/\partial x)^2} \; : \; \Omega \to \mathfrak{R}^2 \qquad \textbf{(Eq I.2)}$$

determines the presence of edges in an image.

The *Laplacian* of an image function *I(x, y)* is the sum of the second-order derivatives, defined as; [10]

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \; : \; \Omega \to \Re. \quad . \qquad \textbf{(Eq I.3)}$$

The general use of the Laplacian is in fin ding the location of edges using its zero-crossings [11]. A critical disadvantage of the gradient operation is that the derivative enhances noise. As a second-order derivative, the Laplacian is even more sensitive to noise. An alternative is convolving an image with the Laplacian of a Gaussian (LoG) function [12], given by

$$LoG(x,y) = \frac{1}{-\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) : \Omega \to \mathfrak{R}^2 \qquad \textbf{(Eq I.4)}$$

where a two-dimensional Gaussian function with the standard deviation σ is defined as

$$G(x,y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) : \Omega \rightarrow \mathfrak{R}^2 \qquad \textbf{(Eq I.5)}$$

The LoG function produces smooth edges as the Gaussian filtering provides smoothing effect.

Sobel operation [13] is performed by convolving an image with kernels shown in Fig I.2. Sobel operators have the advantage of providing both a derivative and a smoothing effect. The smoothing effect is a particularly attractive feature of the Sobel operators compared to the gradient kernels shown in Fig I.2 because the derivative enhances noise.

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

(a)

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

(b)

Fig I.2: Sobel operators along: (a) vertical direction, (b) horizontal direction [13].

Canny edge detector [14] is based on the extrema of the first derivative of the Gaussian operator applied to an image. The operator first smoothes the image to eliminate noise, and then finds high gradient regions. After non-maximum suppression, the edges are finally determined by two thresholds, i.e. T_{min} and T_{max} as shown in tableI.1.

Canny edge detector is known as an optimal edge detector because it satisfies the criteria of low error rate, good localization of edge points, and a single response to a single edge pixel.

Edge detection by gradient operations generally work well only in the images with sharp intensity transitions and relatively low noise. Due to its sensitivity to noise, some smoothing operation is generally required as preprocessing, and the smoothing effect consequently blurs the edge information. However, the computational cost is relatively lower than other segmentation methods because the computation can be done by a local filtering operation, i.e. convolution of an image with a kernel. Edge-based active contour models, use the magnitude of gradient ∇I to determine the position of edges.

- If $|\nabla I(x,y)| > \tau_{max}$, then $I(x,y)$ is an edge pixel.
- If $\tau_{min} < |\nabla I(x,y)| < \tau_{max}$,
  - If there is a path from (x,y) to neighbor $(\aleph)$ and $|\nabla I(\aleph)| > \tau_{min}$, then $I(x,y)$ is an edge pixel.
  - Otherwise, $I(x,y)$ is a non-edge pixel.
- If $|\nabla I(x,y)| < \tau_{min}$, then $I(x,y)$ is a non-edge pixel.

Table I.1: Path searching in Canny edge detector [1]

I.2.2 Region-based Segmentation

Region-based segmentation looks for uniformity within a sub-region, based on a desired property, e.g. intensity, color, and texture. *Clustering* techniques encountered in pattern classification literature have similar objectives and can be applied for image segmentation [15].

Region growing is a technique that merges pixels or small sub-regions into a larger sub-region. The simplest implementation of this approach is *pixel aggregation* which starts with a set of seed points and grows regions from these seeds by appending neighboring pixels if they satisfy the given criteria. Fig I.3 shows a simple example of pixel aggregation.

Segmentation starts with two initial seeds, and then the regions grow if they satisfy a criterion such as:

$$|I(x,y) - I(seed)| < \tau \qquad (\boldsymbol{I.6})$$

Despite the simple nature of the algorithm, there are fundamental problems in region growing: the selection of initial seeds and suit able properties to grow the regions. Selecting initial seeds can be often based on the nature of applications or images. For example,

2	2	9
2	2	9
2	9	9

(a)

2	4	8
3	5	9
4	6	7

(b)

Figure I.3: Pixel aggregation: (a) original image with seeds underlined; (b) segmentation result with $T = 4$[10]

The ROI is generally brighter than the background in IR images. In this case, choosing bright pixels as initial seeds would be a proper choice.

Additional criteria that utilize properties to grow the regions lead region growing into more sophisticated methods, e.g. region competition. *Region competition,* merges adjacent sub- regions under criteria involving the uniformity of regions or sharpness of boundaries. Strong criteria tend to produce over-segmented results, while weak criteria tend to produce poor segmentation results by over-merging the sub-regions with blurry boundaries. An alternative of region growing is *split-and-merge*, which partitions an image initially into a set of arbitrary, disjointed sub-regions, and then merge and/or split the sub-regions in an attempt to satisfy the segmentation criteria.

Another common approach in region-based segmentation is characterizing statistical uniformity of sub-regions using parametric models, so called *statistical estimation.* With this approach, two sub-regions are considered to be uniform, and consequently merged, if they can be represented by a single instance of the model, i.e. if they have common parameter values within a threshold. In practice, the parameters of a sub-region cannot be observed directly but can only be inferred from the observed data and the knowledge of the imaging process. In statistical approaches, this inference is often made using *Bayes's rule* [16] and the conditional **PDF** $\boldsymbol{p}(\boldsymbol{I}(\boldsymbol{x},\boldsymbol{y})/\boldsymbol{\theta_m})$, which presents the conditional probability that certain data $\boldsymbol{I}(\boldsymbol{x},\boldsymbol{y})$ (or statistics derived from the data) will be observed, given that sub-region m has the parameter values of B_m. In typical statistical region merging algorithms stochastic estimates in the parameter space are obtained for different sub-regions, and merging decisions are based on the similarity of these parameters.

A limitation of most estimation-based segmentation methods is that they do not explicitly represent the uncertainty in the estimated parameter values and, therefore, are prone error when parameter estimates are poor. A *Bayesian probability of homogeneity* directly exploits all of the information contained in the statistical image models, instead of estimating parameter values. The probability of homogeneity is based on the ability to formulate a prior probability density on the parameter space, and measures homogeneity by taking the expectation of the data likelihood over a posterior parameter space.

Image segmentation is often approached by *edge-preserving smoothing* operations as well as the partitioning operation. Edge-preserving smoothing techniques can be classified roughly two approaches: Markov random field (MRF) including energy-based

methods and diffusion-based methods. Both approaches show similar restoration characteristics because the diffusion-based methods can be viewed as an energy-based method that uses only the prior energy term at a given temperature. Snyder et al. proposed an edge-preserving smoothing method for image segmentation based on the technology called *mean field annealing* (MFA), and the same segmentation method was extended to vector-valued images by Han et al. MFA is an energy-based method for finding the minimum of complex functions which typically have many minima. For the image segmentation problem, a proper energy function is defined intending to keep the edges and to smooth the rest of areas in the image. The segmentation is performed by minimizing the energy function using MFA. MFA approximates a stochastic algorithm called simulated annealing (SA), which has shown to converge to the global minimum, even for non-convex problems. Hiriyannaiah et al [17]. Derived MFA using the analogy to physics for the restoration of piecewise-constant images, and Bilbro et al [18]. Did the same job applying the MFA to the images with varying gray values.

Region-based approaches are generally less sensitive to noise, and usually produce more reasonable segmentation results as they rely on global properties rather than local properties, but their implementation complexity and computational cost can be often quite large. Statistical segmentation methods, both estimation-based and Bayesian-based, have been extended too many active contour models including the proposed models. Those active contour models based on statistical segmentation [10].

I.2.3 Other Segmentation Methods

The *watershed* [19] algorithm is a morphology-based segmentation method

It is based on the assumption that any gray-tone image can be considered as a topographic surface. If we flood this surface from its minima preventing the merge of the waters coming from different sources, the surface is eventually separated as two different sets: the catchment basins and the watershed lines. If we apply this transformation to the magnitude of image gradient $|\nabla I|$, the catchment basins correspond to the uniform sub-regions in the image and the watershed lines correspond to the edges. The flooding operation is simulated using morphological distance operators [20].

Fusions of different principles have produced good results. There have been a few approaches to integrate region- and edge-based segmentation, and also an approach to integrate region- and morphology-based segmentation called watersnakes [21].

Texture is another feature that we can use to determine the segmentation criteria.

Images can be considered as either a collection of pixels in the spatial domain or the sum of sinusoids of infinite extent in the spatial-frequency domain. Gabor observed that the spatial representation and the spatial-frequency representation are just opposite extremes of a continuum of possible joint space spatial-frequency representations [22]. In a joint space spatial-frequency representations for images, frequency is considered as a local phenomenon that can vary with position throughout the image. The human visual system is performing a form of local spatial-frequency analysis on the retinal image, and the analysis is done by a bank of band pass filters.

The same approach can be used to partition textured images in image analysis. Perceptually significant texture differences presumably correspond to differences in the local spatial frequency content using the space spatial-frequency paradigm. Texture segmentation is done by two steps: decomposing an image into a joint space/spatial-frequency representation with a bank of band pass filters and using this information to locate the regions of similar local spatial frequency content. The response of the filter bank generates a kind of multispectral images, where each band represents the response of the textured image at a particular spatial-frequency bandwidth. The multi-channel filtering has been implemented by the convolution of the image with a stack of two-dimensional Gabor filters or wavelets [10].

I.3 Active Contours

The technique of active contours has become quite popular for a variety of applications. Particularly image segmentation and motion tracking, during the last decade this methology is based upon the utilization of deformable contours which conform to various abject shapes and motions. This chapter provides a theoretical background of active contours and an overview of existing active contour methods.

There are two main approaches in active contours based on the mathematic implementation: *snakes* and *level sets.* Snakes explicitly move predefined snake points based on an energy, minimization scheme, while level set approaches move contours implicitly as a particular level of a function. As image segmentation methods, there are two kinds of active contour models according to the force evolving the contours: edge- and region-based. Edge-based active contours use an edge detector, usually based on the image gradient, to find the boundaries of sub-regions and to attract the contours to the detected boundaries. Edge-based approaches are closely related to the edge-based

segmentation discussed in section I.2.1. Region-based active contours use the statistical information of image intensity within each subset instead of searching geometrical boundaries. Region-based approaches are also closely related to the region-based segmentation discussed in section I.2.2.

I.3.1 Snakes

The first model of active contour was proposed by Kass et al [23]. And named snakes due to the appearance of contour evolution. Let us define a contour parameterized by arc length s as

$$C(s) = \{(x(s), y(s)) : 0 \leq s \leq L\} : \mathfrak{R} \rightarrow \Omega\,, \qquad \textbf{(Eq I.7)}$$

Where L denotes the length of the contour C, and Ω denotes the entire domain of an image $I(x, y)$. the corresponding expression in a discrete domain approximates the continuous expression as

$$C(s) \approx C(n) = \{(x(n), y(n)) : 0 \leq n \leq N, s = 0 + n\Delta s\} \qquad (\textbf{Eq I.8})$$

where $L = N\Delta s$ an energy function $E(C)$ can be defined on the contour such as

$$E(C) = E_{int} + E_{ext} \qquad \textbf{(Eq I.9)}$$

where E_{int} *and* E_{ext} respectively denote the *internal energy* and *external energy*
The internal energy function determines the regularity, i.e. smooth shape, of the contour. A common choice for the internal energy is a quadratic functional given by

$$\boldsymbol{C(s) = \int_0^L (\alpha |C'(s)|^2 + \beta |C''(s)|^2)\, ds} \qquad (\textbf{Eq I.10})$$

$$\approx \sum_{n=0}^{N} (\alpha |C'(n)|^2 + \beta |C''(n)|^2)\Delta s$$

Here α controls the tension of the contour, and β controls the rigidity of the contour .The external energy determines the criteria of contour evolution depending on the image $I(x, y)$ and can be defined as

$$E_{ext} = \int_0^l E_{img}(C(s))ds \approx \sum_{n=0}^{N} E_{img}(C(n))\Delta s \qquad (\textbf{Eq I.11})$$

where $\boldsymbol{E_{img}(x, y)}$ denotes a scalar function defined on the image plane, so the local minimum of $\boldsymbol{E_{img}}$ attracts the snakes to edges. A common example of the edge attraction function is a function of image gradient, given by

$$E_{img}(x, y) = 1/\mu|\nabla G_\sigma * I(x, y)| : \Omega \to R \qquad \textbf{(Eq I. 12)}$$

where $G_\sigma$denotes a Gaussian smoothing filter with the standard deviation σ , and μ is a suitably chosen constant. Solving the problem of snakes is to find the contour C that minimizes the total energy term *E* with the given set of weightsα, βandμ. In numerical experiments, a set of snake points residing on the image plane are defined in the initial stage, and then the next position of those snake points are determined by the local minimum *E*. The connected form of those snake points is considered as the contour.
Fig I.4 shows an example of classic snakes. There are about 70 snakes points in the image, and the snake points form a contour [10]

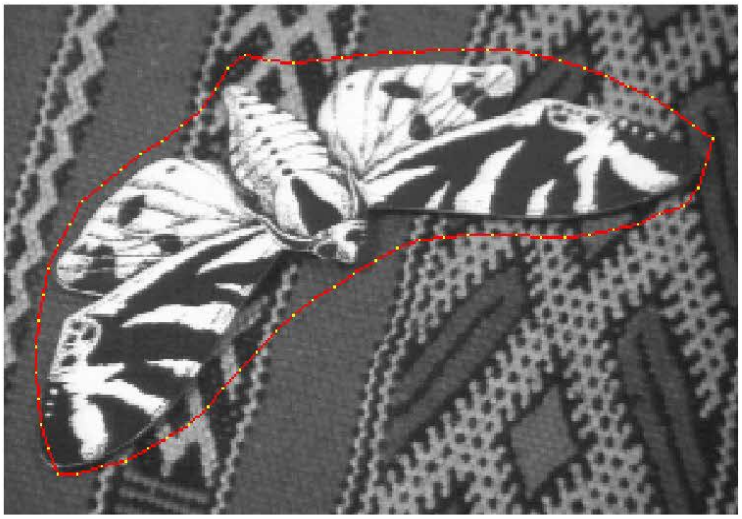


Fig I.4: An example of classic snakes [1]

around the moth. The snakes points are initially placed at further distance from the boundary of the object, i.e. the moth. Then, each point moves towards the optimum coordinates, where the energy function converges to the minimum. The snakes points eventually stop on the boundary of the object. More details of the Snakes model is respectively discussed in next chapters.

The classic snakes provide an accurate location of the edges only if the initial contour is given sufficiently near the edges because they make use of only the local information along the contour. Estimating a proper position of initial contours without prior knowledge is a difficult problem. Also, classic snakes cannot detect more than one boundary simultaneously because the snakes maintain the same topology during the evolution stage. That is, snakes cannot split to multiple boundaries or merge from multiple initial contours. Level set theory has given a solution for this problem[10].

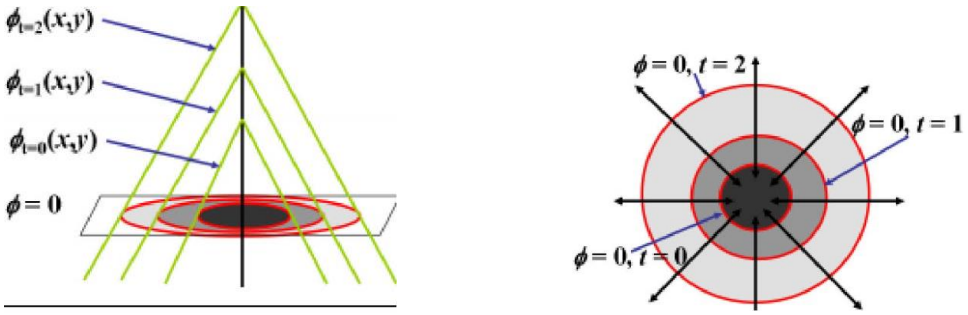
I.3.2 Level Set Methods

Level set theory, a formulation to implement active contours, was proposed by Osher and Sethian [24]. They represented a contour implicitly via a two-dimensional Lipschitz-continuous function $\phi(x,y): \Omega \rightarrow \mathfrak{R}$ defined on the image plane. The function

$\phi(x,y)$ is called level set function, and the particular level, usually the zero level, of is defined as the contour , such as

$$C \equiv \{(x,y): \phi(x,y) = 0\}, \forall (x,y) \in \boldsymbol{\Omega} \qquad (\boldsymbol{Eq\ I.12})$$

where Ω denote the entire image plane. Fig I.5 shows the evolution of level set function $\phi(x,y)$, and Fig I.5 (b) shows the propagation of the corresponding contours C. As the Level **set**

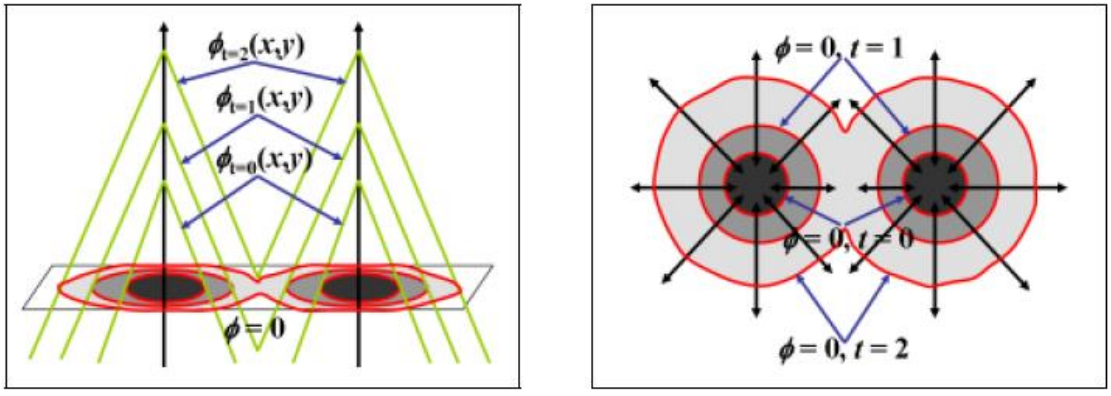


(a) (b)

Fig I.5: Level set evolution and the corresponding contour propagation:(a) topological view of level set $\phi(x,y)$ evolution, (b) the changes on the zero level set C: $\phi(x,y) = 0$ [10].

The deformation of the contour is generally represented in a numerical form as a PDE. A formulation of contour evolution using the magnitude of the gradient of $\phi(x,y)$ was initially proposed by Osher and Sethian.

An outstanding characteristic of level set methods is that contours can split or merge as the topology of the level set function changes. Therefore, level set methods can detect more than one boundary simultaneously, and multiple initial contours can be placed. Fig I.6 (a) shows an example of the topological changes on a level set function, while Fig I.6 (b) shows how the initially separated contours merge as the topology of level set function varies. This flexibility and convenience provide a means for an autonomous segmentation by using a predefined set of initial contours. The computational cost of level set



(a) (b)

Fig I.6: The change of topology observed in the evolution of level set function and the propagation of corresponding contours: (a) the topological view of level set $\phi(x,y)$ evolution, (b) the changes on the zero level set C: $\phi(x,y) = 0$[10]

Methods is high because the computation should be done on the same dimension as the image plane n. Thus, the convergence speed is relatively slower than other segmentation methods, particularly local filtering based methods. The high computational cost can be compensated by using multiple initial contours. The use of multiple initial contours increases the convergence speed by cooperating with neighbor contours quickly.

Level set methods with faster convergence, called fast marching methods, have been studied intensively for the last decade. Because of these attractive properties, we implement the proposed active contour model using the level set method.

I.3.3 Edge-based Active Contours

Edge-based active contours are closely related to the edge-based segmentation. Most edge based active contour models consist of two parts: the regularity part, which determines the shape of contours, and the edge detection part, which attracts the contour towards the edges.

I.3.4 Region-based Active Contours

Most region-based active contour models consist of two parts: the regularity part, which determines the smooth shape of contours, and the energy minimization part, which searches for uniformity of a desired feature within a subset. A nice characteristic of region-based active contours is that the initial contours can be located anywhere in the image as region-based segmentation relies on the global energy minimization rather than local energy minimization. Therefore, less prior knowledge is required than edge-based active contours [10].

Conclusion

From this chapter we have present the relationship between segmentation of image and active contour, where we have given two main approaches based on the mathematic implementation: *snakes* and *level sets.* Snakes (snake & b-snake) explicitly move predefined snake points based on an energy, minimization scheme, while level set approaches move contours implicitly as a particular level of a function.

In the next chapters we are going to present with detail b-snake method.

CHAPTER II

BACKGROUND OF B-SPLINE

II.1 Introduction

A parametric deformable contour is constructed as a weighted sum of some bases function and the expansion coefficients constitute the parameter vector. From many basis to choose, Fourier and B-Spline basis have been most frequently used in computer vision.

Each parameterization has particular properties that make it suitable for different purposes. Some restrictions that may be comprised within some basis representations (such as smoothness or continuity) can be convenient because the necessary constraints are built directly into the representation and they do not have to be regularized subsequently. It is also desirable for the parameterization to be concise that determines the complexity of the optimization process.

Before considering any of these bases, let's define mathematically some of the terms we have used so far and introduce an appropriate notation.

II.2 The Choice of Curve Representation

There are three ways of expressing a curve which are normally used. These are the following:

1. explicit: $y = f(x)$

2. implicit : $f(x, y) = 0$

3. parametric : $x = x(t), y = y(t)$

The difficulties associated with (i) are that it is impossible to get multiple values of y for a single x, so that closed curves such as circles and ellipses, must be represented by multiple curve segments. Further, it is difficult to represent a curve with the vertical tangent, since

a slope of infinity is difficult to represent. Since the curve representing the snake often has both of those properties, the explicit representation is not well suited for it.

The problems with the implicit representation are the following: the equation may have more than one solution. For example, the implicit equation of a circle is $x^2 + y^2 = 0$. Should we want to model a half circle, we would have to add constraints such as $x \geq 0$ which cannot be contained within the explicit equation. Furthermore, if two implicitly defined segments are joined together, it may be difficult to determine if their tangent directions agree at the point.

The parametric representation for curves x = x (t), y = y (t) overcomes the problems caused by implicit or explicit forms. We can get multiple values of y for a single x on the plot since we do not plot y against x but we plot both x and y against a third variable t, which is not shown on the plot.

The parametric curve form solves the slope of infinity problem too, since it replaces the use of geometric slopes with parametric tangent vector which is never infinite [25].

II.3 Definition of the Parametric Curve Representing the Snake

Let $\theta \in \Theta$ denote the parameter vector of a closed contour where Θ is the space of all configurations and deformations of the closed curves. The number of elements in the parameterization vector is the order of parameterization. In Fourier representation it is the number of Fourier descriptors (coefficients), in Spline representation it is the number of control points. The curve represented by such a vector is a continuous, periodic, vector function v(s) = [x(s), y(s)], of period L (in Fourier representation L= 2π, in B-Spline representation L is an arbitrary real number); its N-point discrete version is a (N × 2) vector

$$v = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{N-1} \end{pmatrix} = \begin{pmatrix} x_0 & y_0 \\ x_1 & y_1 \\ \vdots & \vdots \\ x_{N-1} & y_{N-1} \end{pmatrix} \qquad \textbf{(Eq II.1)}$$

where $v_i = v[iL/(N-1)]$ for $i = 0,1,2,\dots,N-1$

Now, given a parameter vector, we construct a deterministic operator V, defined on Θ, that maps θ into v, i.e. we write $= V\theta$.[25]

II.4 Arc Length as the Curve Parameter

A single parametric curve can be represented by more than one vector function depending on the choice of the parameter.

Suppose that C is a piecewise-smooth curve given by a vector function

$$v(t) = [\mathrm{x(t)}, \mathrm{y(t)}] = \mathrm{x(t)}\,.\mathrm{i} + \mathrm{y(t)}.\mathrm{j} \qquad \textbf{(Eq II.2)}$$

where i i and j j are unit vectors along the x x and y y axis, respectively. We define the arclength function of C by

$$l(s) = \int_a^s |v'(u)|\, du$$

(Eq II.3)

$$= \int_a^s \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dx}{du}\right)^2}\, du$$

where u is a dummy integration variable.

Thus,$l(s)$) is the length of the part of C between $v(a)$ and $v(s)$. It is often useful to parameterize a curve with respect to arc length, because arc length arises naturally from the shape of the curve and does not depend on particular coordinate system [25].

II.5 Spline Basis

One of the most frequently used curve representations in the computer vision is the B-spline representation. It is important to distinguish between splines and B-splines.

B-splines are polynomial functions with minimal support. Splines are linear combinations of B-splines. In the literature, splines are usually defined as a divided difference of a truncated power function. For computational purposes [25].

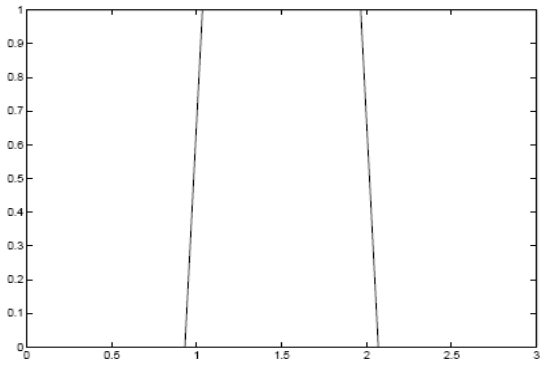
II.5.1 B-Splines

The starting point for all splines is a non-decreasing knot sequence $s = s(i)$. B-splines of order 0 are the characteristic functions of this partition

$$B_{i,0}(s) := \begin{cases} 1, & if\ \ s_i < s < s_{i+1} \\ 0, & otherwise \end{cases} \qquad \textbf{(Eq II.4)}$$

where $:=$ denotes "equality by definition". The only constraint is that these B-splines should form a partition of unity, i.e

$$\sum_i B_{i,0}(s) = 1 \text{ for all s.} \qquad \textbf{(Eq II.5)}$$



FigII.1: First order B-spline [25]

In particular

$$s_i = s_{i+1} \text{ implies } B_{i,0} = 0 \qquad \textbf{(Eq II.6)}$$

From these first-order B-splines we obtain higher-order B-splines by recurrence:

$$B_{i,k} := w_{i,k}\, B_{i,k-1} + (1 - w_{i+1,k}) B_{i+1,k-1} \qquad \textbf{(Eq II.7)}$$

with
$$w_{i,k}(s) := \begin{cases} \frac{s - s_i}{s_{i+k-1} - s_i}, & if\ s_i \neq s_{i+k-1} \\ 0, & otherwise \end{cases} \qquad \textbf{(Eq II.8)}$$

Thus, the second-order B-spline is given by

$$B_{i,1} := w_{i,2}\, B_{i,0} + (1 - w_{i+1,2}) B_{i+1,0} \qquad \textbf{(Eq II.9)}$$

and so consists in general of two nontrivial linear pieces which join continuously to form a piecewise linear function which vanishes outside the interval $[s_i, s_{i+2}]$.

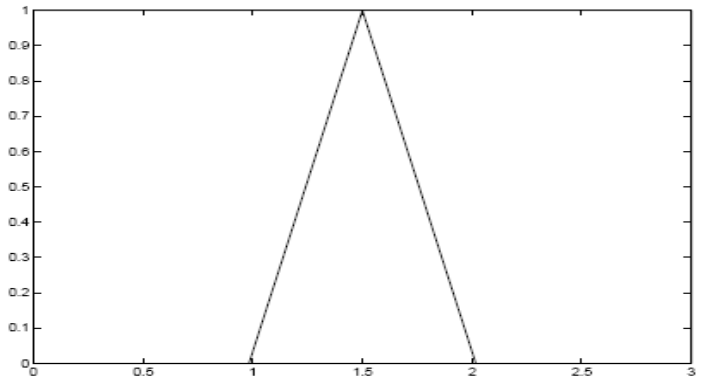


Figure II.2: Second order B-spline [25]

The third order B-spline is given by

$$B_{i,3} = w_{i,3}B_{i,2} + (1 - w_{i+1,3})B_{i+1,2}$$

$$= w_{i,3}w_{i,2} + (w_{i,3}(1 - w_{i+1,2})(1 - w_{i+1,3})w_{i+1,2})B_{i+1,0} + \quad \textbf{(Eq II. 10)}$$

$$+ (1 - w_{i+1,3})(1 - w_{i+2,2})B_{i+2,0}$$

So, $B_{i,3}$ consists of 3 (nontrivial) quadratic pieces.

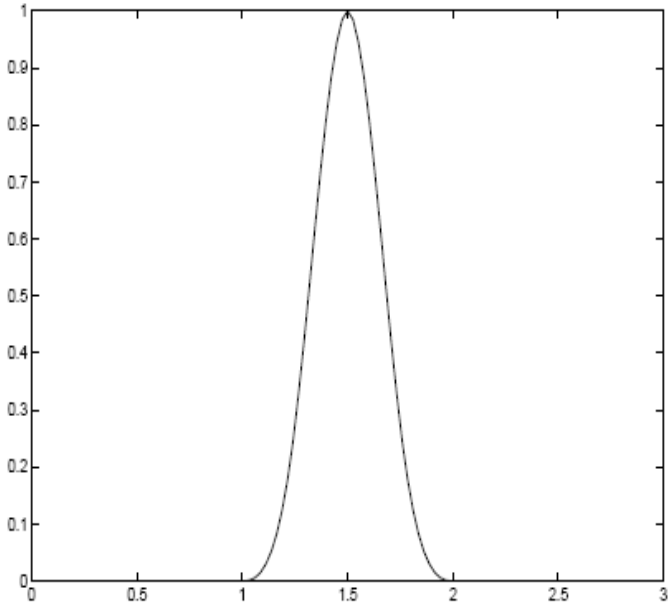


Fig II.3: Third order B-Spline [25]

After $K - 1$ steps of the recurrence, we obtain$B_{i,k}$ in the form

$$B_{i,k} = \sum_{j=1}^{i+k-1} b_{j,k}B_{j,0} \qquad \textbf{(Eq II.11)}$$

with each b_{jk} polynomial of degree < k since it is the sum of products of k -1 polynomials.

Many properties of B-splines are derived most easily by considering not just one B-spline but the linear span of all B-splines of given order k for a given knot sequence s and this brings us to splines [25].

II.5.2 Splines

A spline of order k with knot sequence s is, by definition, a linear combination of the B-splines $B_{i,k}$ associated with that knot sequence. We denote by

$$S_{k,s} := \sum_i B_{i,k}\ a_i :\ a_i \in \boldsymbol{R}\ 2 \qquad \textbf{(Eq II.12)}$$

the collection of all such splines. We will pay special attention to the following knot sequence

$$Z: = (...,-2, -1, 0, 1, 2 ...) \qquad \textbf{(Eq II.13)}$$

which is the set of all integers.

The spline associated with this knot sequence is called a cardinal spline. Because of the uniformity of the knot sequence Z, formulae involving cardinal B-splines are often much simpler than corresponding formulae for general B-splines.

To begin with, all cardinal B-splines (of a given order) are translates of one another.With the natural indexing s_i := i, for all i , we have

$$N_k : = N_k\ (s) := B_{0,k} \qquad \textbf{(Eq II.14)}$$

and

$$B_{i,k} = N_k\ (s - i) \qquad \textbf{(Eq II.15)}$$

The recurrence relation (9) simplifies as follows

$$(k - 1)\ N_k\ (s) = sN_{k-1}\ (s) + (k - s)\ N_{k-1}\ (s - 1) \qquad \textbf{(Eq II.16)}$$

II.2.5.3 Cubic B-Splines and Polygon Approximation

Now, we will take a closer look at the B-spline theory that is relevant for our snake applications.

The reason for choosing B-splines and not so called natural splines which we are familiar with from the classical interpolation theory, is the so called local control property of the B-splines. The polynomial coefficients of the natural splines are dependent on all N control points. Their calculation involves inserting an N + 1 ×N + 1 matrix. This has two disadvantages: moving one control point affects the entire curve and the computation time needed to invert the matrix can interfere with rapid interactive reshaping of a curve. B-splines, on the other hand, consist of curve segments depending on just a few control points. This is called the local control.

Thus, moving a control point affects only a small part of the curve. B-splines have the same continuity as the natural splines but do not interpolate their control points.

Therefore, we speak of polygon approximation and not of interpolation of control points.

The first step would be to choose the order of the basis splines in order to achieve the desired smoothness and still maintain reasonable computational efficiency. We choose cubic splines, which means the splines of order 3:$B_{i,3}(t)$. The reasons are following:

1. The lower-degree polynomials give too little flexibility in controlling the shape of the curve. The 1: order splines (straight lines) do not give satisfactory smoothness of the approximating curve. The 2: order splines (quadratic curves) give a smooth curve but a problem arises at the points where different curve segments join. In order to understand this problem we introduce a new criterion:

Definition 1: *Let* Q_i(s) *denote a curve segment. If the direction and the magnitude of* $\frac{d^n}{dt^n}Q_i$(s) *and* $\frac{d^n}{dt^n}Q_{i+1}$(s) *are equal at the join point, the curve consisting of these two segments is called* C^n *continuous.*

The 2: order splines are $C^0$and C^1continuous, which does not insure a satisfactory continuity at the joint points. The problem is solved by using cubic splines which are C^0,$C^1$and C^2 continuous.

2. The higher-degree polynomials are more time-consuming in the computational process and can introduce unwanted wiggles. The curve might “wiggle”back and forth in ways that are difficult to control.

3. We say that cubic splines give “satisfactory” continuity because it is found that the eye cannot detect a geometric discontinuity of degree higher than two and it is sufficient in practice to consider splines of degree three.

In the rest of the chapter we will introduce the mathematical formalism for the snake curve as a weighted sum of B-splines. Parametric spline curves

$$v[s] = (x[s],y[s]) \qquad \textbf{(Eq II.17)}$$

have coordinates x[s] and y[s], each of which is a spline function of the curve parameters, 0≤s ≤ 1. Since$B_{i,4}$, the cardinal cubic B-spline, has been used in the snake model throughout the literature, we will from now on denote it as Bi and use it in further work. We will now define a spline curve in the plane representing the snake.

For each basis function B_ia control point qi = [q_i^x,q_i^yi] is defined and the snake curve is a weighted sum of control points

$$v[s] = \sum_{i=0}^{N} B_i\,[s]\,\text{qi} \qquad \textbf{(Eq II.18)}$$

which becomes a smooth curve that follows approximately "the control polygon" defined by linking control points by lines.

Fig II.4: Potential field of the image in Fig [25]

Now we introduce a more compact notation for the control points by defining a space of control vectors K_Q consisting of

$$Q = \begin{pmatrix} Q^x \\ Q^y \end{pmatrix} \quad \text{where} \quad \begin{pmatrix} q_0^x \\ \vdots \\ q_{N-1}^x \end{pmatrix}^T \qquad \textbf{(Eq II.19)}$$

Then the coordinate functions can be written as

$$\text{x[s]} = \text{B[s]}^{\text{T}} Q^x \qquad \textbf{(Eq II.20)}$$

and

$$\text{y[s]} = \text{B[s]}^{\text{T}} Q^y \qquad \textbf{(Eq II.21)}$$

where $B[s] = (B_0[s], B_1[s], \dots, B_{N-1}[s])^T$ is a vector of B-spline functions. The parametric curve v[s] becomes then

$$v[s] = U[s]Q \quad \text{for } 0 \leq s \leq 1 \qquad \textbf{(Eq II.22)}$$

where

$$U[s] = I2 \otimes B[s]^T = \begin{pmatrix} B[s]^T & 0 \\ 0 & B[s]^T \end{pmatrix} \qquad \textbf{(Eq II.23)}$$

where $\otimes$ denotes the Kronecker product.

Comparing this result with our definition of the snake curve in the section 4.2, we see that we can interpret the set of control points defining the spline as the parameter vector

$$\theta = (q_0^T q_1^T \dots q_2^T) \qquad \textbf{(Eq II.24)}$$

describing the shape of the curve. The matrix operator V is in this case

$$v = V\theta = B\theta \qquad \textbf{(Eq II.25)}$$

where the elements of the matrix B are given by $[B]_{ij} = B_i(2 \times j = (N - 1))$. We can rewrite the expression (4.18) further. Restricting B_i[s] to the interval $[si, si + 1]$, we can identify it with a polynomial in s of degree $\leq$ k. We will always assume in this section that the interval $[si, si + 1]$ is in fact $[0, 1]$; this can be obtained with an affine change of variable.

Each $B_i[s]$ is a polynomial in s which may be written as its Taylor series at 0. We may therefore write

$$(B_{i-k}[s], \_\_\_, B_i[s]) = (1, s, \dots, s^k)M \qquad \textbf{(Eq II.26)}$$

where M is a (k + 1) × (k + 1) matrix. The curve v is thus represented in the following matrix form

$$x[s] = (1, s, \dots, s^k)M \begin{pmatrix} q_0^x \\ \vdots \\ q_{N-1}^x \end{pmatrix}^T \qquad \textbf{(Eq II. 27. a)}$$

and

$$y[s] = (1, s, \dots, s^k) M \begin{pmatrix} q_0^y \\ \vdots \\ q_{N-1}^y \end{pmatrix}^T \qquad (\textbf{Eq II. 27. b})$$

Now let us assume that we have a set of control points (xi, yi). First we parameterize the set:

$$x[s_i] = x_i \qquad y[s_i] = t_i; \qquad (\textbf{Eq II. 28})$$

where s_i, $0 \leq i \leq N - 1$ are evenly spaced points - integers - on the real axis.

We parameterize the curve for the reasons explained in the beginning of the chapter, and knots are chosen to be evenly spaced since it leads to cardinal splines which are easier and less computational costly to generate than the non-uniform splines.

Next, we generate the splines using the recurrence formulae **(Eq II.7).** We notice that each spline of order 1 is locally controlled by three knots, splines of order 2 are locally controlled by four knots and splines of order 3 are controlled by five knots.

We begin by computing the expression of $B_3[s]$on each interval$[s_i, s_{i+1}]$, $0 \leq i \leq 3$

and the knots are 0, 1, 2, 3, and 4.

$$B_3[s] = B_{0,3}[s] = \frac{1}{3}(sB_{0.2}[s] + (4 - s)B_{1,2}[s])$$

$$B_{0,2}[s] = \frac{1}{2}(sB_{0,1}[s] + (3 - s)B_{1,1}[s])$$

$$B_{1,2}[s] = \frac{1}{2}((s - 1)B_{1,1}[s] + (4 - s)B_{2,1}[s])$$

$$B_{0,1}[s] = sB_{0,0}[s] + (2 - s)B_{1,0}[s]$$

$$B_{1,1}[s] = (s - 1)B_{1,0}[s] + (3 - s)B_{2,0}[s] \qquad (\textbf{Eq II. 29})$$

$$B_{2,1}[s] = (s - 2)B_{2,0}[s] + (4 - s)B_{0,3}[s]$$

which gives

$$B[s] =: \begin{cases} \frac{1}{6}s^3, & for\ 0 \le s < 1 \\ \frac{1}{6}(-3s^3 + 12s^2 - 12s + 4), & for\ 1 \le s < 2 \\ \frac{1}{6}(3s^3 - 24s^2 + 60s - 44), & for\ 2 \le s < 3 \\ \frac{1}{6}((4 - s)^3) = \frac{1}{6}(-s^3 + 12s^2 - 48s + 64), & for\ 3 \le s < 4 \end{cases} \quad (\textbf{Eq II. 30})$$

On the interval $[0, 1]$, the function $x[s]$ reduces than to (say for $x[s]$ and correspondingly for $y[s]$)

$$B[s + 3]q_{-3}^x + B[s + 2]q_{-2}^x + B[s + 1]q_{-1}^x + B[s]q_0^x$$

$$= \frac{1}{6}[(1 - s)^3 q_{-3}^x + (3(s + 2)^3 - 24(s + 2)^2 + 60(s + 2) - 44)q_{-2}^x$$

$$+ (-3(s + 1)^3 + 12(s + 1)^2 - 12(s + 1) + 4)q_{-1}^x + s^3 q_0^x \,] \quad (\textbf{Eq II. 31})$$

$$= \frac{1}{6}[(-s^3 + 3s^2 - 3s + 1)q_{-3}^x + (3s^3 - 6s^2 + 4)q_{-2}^x$$

$$+ (-5s^3 + 3s^2 + 3s + 1)q_{-1}^x + s^3 q_0^x \,]$$

and correspondingly for $y[s]$. It gives the following matrix representation

$$x[s] = \frac{1}{6}(1, s, s^2, s^3)\begin{pmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & 1 & -3 \end{pmatrix}\begin{pmatrix} q_{-3}^x \\ q_{-2}^x \\ q_{-1}^x \\ q_0^x \end{pmatrix} \quad (\textbf{Eq II. 32})$$

The portion of the curve x[s] (and correspondingly y[s]) corresponding to $i \le s + 1 \le i + 1$

is written with the same matrix as [25]

$$x[s]\frac{1}{6}(1, s, s^2, s^3)\begin{pmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & 1 & -3 \end{pmatrix}\begin{pmatrix} q_{i-3}^x \\ q_{i-2}^x \\ q_{i-1}^x \\ q_i^x \end{pmatrix} \quad (\textbf{Eq II. 33})$$

II.6 Conclusion

When constructing a snake, it is convenient to choose a set of control points on the image and then generate the snake as an approximation to the polygon which arises by connecting the point with straight lines. The question is why to choose the spline representation?

-Splines are polynomial, and being polynomial they can be evaluated efficiently on computer

-They are piecewise polynomial which makes them very flexible.

-They can be represented as a linear combination of B-splines and this representation provides geometric information and insight.

The snake curve is closed which implies periodicity. This is easily achieved by treating the parameter t as periodic.

-In the snake model it is important that the curve is smooth. We impose constraints

-On the curve in the energy model in order to give it flexibility and smoothness. But the splines minimize the deformation-like energy so the search for the solution-curve may be confined to a set of such functions under the action of the image potential. The choice of the cubic-spline as basis spline is a pragmatic approach. The linear spline does not give a smooth function since the first derivative of the function is not continuous. The cubic spline gives satisfactory smoothness while it still is computationally relatively efficient.

-Splines preserve the shape which means that a spline has the same shape as its control polygon, or more precisely: A spline crosses any straight line no more often than does its control polygon. In particular, if the control polygon is monotone (convex), then so is the spline.

All these properties make B-splines a very popular choice for curve representation.

CHAPTER III

EXTERNAL FORCE GVF

III.1 Introduction

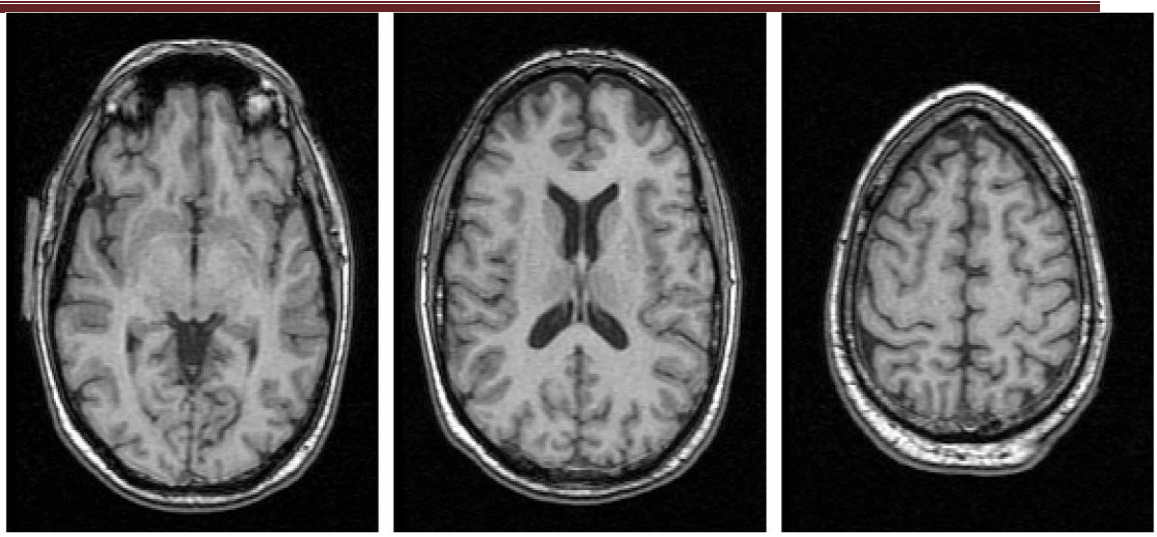
Over the last decade, there has been increasing research activity on deriving a mathematical description of object boundaries from images. This task, also known as *boundary mapping,* is a fundamental step for many active research areas **in** image analysis, computer vision, and medical imaging. Boundary mapping is aimed at helping us to augment our understanding of and to form conclusions about various properties of objects of interest **in** images. The applications of boundary mapping include image segmentation, motion tracking, shape modeling, object recognition, and image registration and warping. For obtain that, we need to study the external force by using the gradient which is the main study for this chapter.

-Back ground of Boundary mapping

Mapping object boundaries from images is a difficult task due to the tremendous variability of object shapes and diverse image sources. For example, one important task **in** medical imaging is the boundary mapping of the brain cortex from 3-D magnetic resonance (MR) images (Fig III.1), where we are facing highly convoluted shape of brain cortex, imaging noise, sampling artifacts, and large-scale image data set. Imaging noise and sampling artifacts especially may cause the boundaries of objects of interest to be indistinct and disconnected. How to integrate these boundaries into a coherent and consistent mathematical description is a challenging problem that a boundary mapping technique has to address.

Boundary mapping methods abound **in** the literature of image analysis, computer vision, and medical imaging. Edge detection and linking, region growing, and relaxation labelling are among the most widely used "classical" boundary mapping techniques. Edge detection and linking is a two-step technique. First, an edge detector is applied to an image to identify boundary elements through detecting intensity discontinuities. Then, an edge linking algorithm is used to link the boundary elements together to obtain a parameterized curve or surface representation. Region growing is a region-based technique that usually starts with a set of "seed" points and from these grows regions by merging neighboring pixels or voxels that share similar properties.

Relaxation labeling is a technique for segmenting objects through a class of locally cooperative and parallel processes based on the intensity difference among neighboring pixels or voxels. Further information about these classical boundary mapping methods can be found in most image analysis and computer vision textbooks.



(a) (b) (c)

Fig (III.1) sample image slices from acquired 3-D MRI data set [26]

One limitation of these classical methods is that they only consider local information, so that incorrect assumptions may be made during the boundary integration process causing generation of infeasible object boundaries. Furthermore, these methods usually generate results that are constrained by the resolution of the images and do not necessarily lead to accurate results. To address these problems, we have explored a boundary mapping technique called *deformable models* which is based on the work of Kass et al [4]. Various names have been used to refer deformable models in the literature. In 2-D, deformable models are usually referred as snakes, active contours, balloons, and deformable contours. In 3-D, they are usually referred as active surfaces and deformable surfaces. In this thesis, we shall refer 2-D deformable models as deformable contours and 3-D deformable models as deformable surfaces.

III.2 Deformable Models

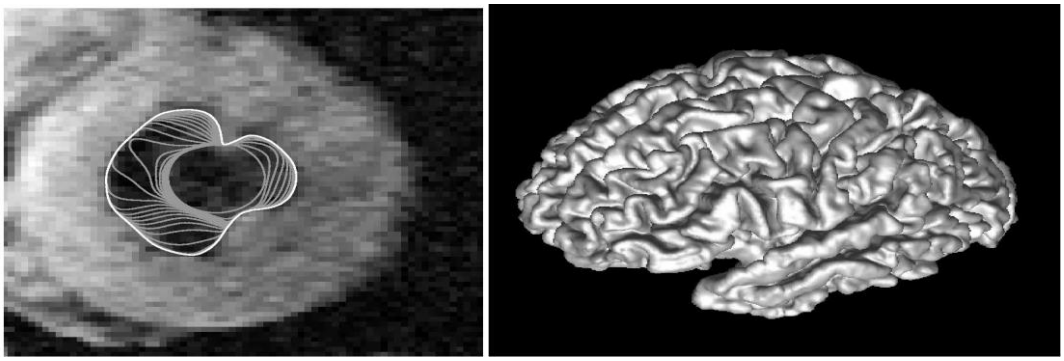
Deformable models are elastic curves or surfaces defined within an image domain that can move under the influence of internal forces coming from within the curve or surface itself and external forces computed from the image data. The internal and external forces are defined so that the deformable model will conform to an object boundary or other desired features within an image. Fig. III.2 shows two examples of using both a deformable contour and a deformable surface to reconstruct anatomical boundaries from MR images. The results shown in this figure are obtained using the deformable models developed in this section.

Mathematically, deformable models are represented as parameterized manifolds (curves or surfaces) x (u), where u is the parameter of the manifold. The shape of the manifold is typically

determined by a variational formulation whose general form is the following: find the x(u) that minimize the energy functional

$$\mathcal{E} = \mathcal{E}_{\text{int}} + \mathcal{E}_{\text{ext}} = \int_{\mathbf{U}} \mathbf{E}_{\text{int}}(\mathbf{x}(\mathbf{u})) + \mathbf{E}_{\text{ext}}(\mathbf{x}(\mathbf{u}))\, d\mathbf{u}. \qquad \textbf{(Eq III.1)}$$

This functional can be viewed as a representation of the energy of the manifold, and the final shape of the manifold corresponds to the minimum of this energy. The first term $\mathcal{E}_{int}$ prescribes *a priori* knowledge about the model such as its material properties (elasticity and rigidity). It can be used to characterize the deformation of a membrane or a thin-plate, for example. The second term $\mathcal{E}_{ext}$ is usually derived from image data and takes a minimum when the deformable model lies in the feature of interest such as object boundary. More discussion about deformable models is provided in next sections.



(a) (b)

Fig III.2: (a) A 2-D example of using a deformable contour to extract the inner wall of the left ventricle of a human heart from an MR image. A sequence of deformable contours (plotted **in** a shade of gray) and the final converged result (plotted **in** white). (b) A 3-D example of using a deformable surface to reconstruct the brain cortical surface from a 3-D MR Image [26].

III.3 Deformable Contours

A traditional deformable contour is a curve $x(s) = [x(s), y(s)], s \in [0,1]$, that moves through the spatial domain of an image to minimize the energy functional

$$E = \int_0^1 \frac{1}{2}\left(\alpha|x'(s)|^2 + \beta\left|x''(s)\right|^2\right) + E_{ext}(x(s))ds \qquad \textbf{(Eq III.2)}$$

where α and β are weighting parameters that control the contour's tension and rigidity, respectively, and $x'(s)$ and $x''(s)$ denote the first and second derivatives of $x(s)$ with respect to

s. The external energy E_{ext} is a function derived from the image so that it takes on its smaller values at the features of interest, such as boundaries. Given a gray-Ievel image $I(x,y)$, viewed as a function of continuous position variables(x,y), typical external energies designed to lead a deformable contour toward step edges are [4]:

$$E_{ext}^{(1)}(x,y) = -|\nabla I(x,y)|^2 \quad \textbf{(Eq III.3)}$$

$$E_{ext}^{(2)}(x,y) = -|\nabla(G_\sigma(x,y) * I(x,y))|^2 \quad \textbf{(Eq III.4)}$$

where $(G_\sigma(x,y)$ is a two-dimensional Gaussian function with standard deviation α and ∇ is the gradient operator. If the image is a line drawing (black on white), then appropriate external energies include [28]:

$$E_{ext}^{(3)}(x,y) = I(x,y) \quad \textbf{(Eq III.5)}$$

$$E_{ext}^{(4)}(x,y) = G_\sigma(x,y) * I(x,y) \quad \textbf{(Eq III.6)}$$

It is easy to see from these definitions that larger's will cause the boundaries to become blurry. Such large's are often necessary, however, in order to increase the capture range of the deformable contour.

The problem of finding a parameterized curve $x(s)$ that minimizes E is known as the variational problem [21]. It has be shown that the curve $x(s)$ that minimizes E must satisfy the following Euler equation [4] [28].

$$\alpha x''(s) - \beta x''''(s) - \nabla Eext = 0 \quad \textbf{(Eq III.7)}$$

Various choices of boundary conditions for $x(s)$ may be used, we use periodic boundary condition $x(0) = x(l)$ since we deal only with closed contours.

To gain some insight about the physical behaviour of deformable contours, we can view **Eq (III.7)** as a force balance equation

$$\mathbf{F}_{\text{int}} + \mathbf{F}_{\text{ext}}^{(\text{p})} = 0 \quad \textbf{(Eq III.8)}$$

where $F_{int} = \alpha x''(s) - \beta x''''(s)$ and $F_{ext}^{(p)} = -\nabla E_{ext}$. The internal force F_{int} discourages stretching and bending while the external potential force $F_{ext}^{(p)}$ pulls the contour towards the desired image edges.

Euler equation **(Eq III.7)** provides the necessary condition for any curve that minimize the energy functional E. In general, however, since the energy functional E is non-convex, the Euler equation has many solutions that correspond to the local minimum of E [27] [28]. Although a global minimum can be found using several existing global optimization techniques such as graduated non-convexity algorithms [30], and genetic algorithms [31] [32], they are generally much more computational intensive than local minimization techniques such as gradient descent methods. Will use gradient descent methods to find the minimum. One of the consequence of using gradient descent methods is that a good initialization is required to obtain a satisfactory solution.

To find a solution to **(Eq III.6)** the deformable contour is made dynamic by treating x as function of time t as well as *s* - i.e., *x(s, t)*. We note that adding a time directive term of x is equivalent to applying gradient descent algorithm to find the local minimum of **(Eq III.1).** Then, the partial derivative of x with respect to t is then set equal to the left hand side of **(Eq III.6)** as foIlows

$$x_t(s,t) = \alpha x''(s,t) - \beta x''''(s,t) - \nabla E_{ext} \qquad \textbf{(Eq III.8)}$$

When the solution $x(s,t)$ stabilizes, the term $x_t(s,t)$ vanishes and we achieve a solution of **Eq (III.7**). A numerical solution to **Eq (III.8**). Can be found by discretizing the equation and solving the discrete system iteratively. note that most deformable contour implementations use either a parameter that multiplies $x(t)$ in order to control the temporal step-size, or a parameter that multiplies E_{ext}, which permits separate control of the external force strength. In this thesis, we normalize the external forces so that the maximum magnitude is equal to one, and use a unit temporal step-size for all the experiments.

III.4 Deformable Surfaces

A traditional deformable surface is a surface $x(u) = [x(u), y(u), z(u)]$, $u = (u^1, u^2) \in [0,1]\, x\, [0,1]$, That moves through the spatial domain of a 3-D image to minimize an energy functional [33,34]. A typical example of such an energy functional is

$$E = \int \frac{1}{2}(\alpha \sum_{i=1}^{2} |\mathbf{x}_i|^2 + \beta \sum_{i,j=1}^{2} |\mathbf{x}_{ij}|^2) + E_{\text{ext}}(\mathbf{x})d\mathbf{u} \qquad \textbf{(Eq III.9)}$$

where α and β are weighting parameters that control the surface's tension and rigidity, x_i and $x_{i,j}$ denote the first and second partial derivatives of x with respect to u^i, and $E_{ext}(x)$ is the external energy function derived from the image that can be defined similarly as that of deformable contours. It can also be shown that the deformable surface minimizing the above energy functional can be obtained by fin ding the steady state solution of the following dynamic equation:

$$\mathbf{x}_t = \mathbf{F}_{\text{int}} + \mathbf{F}_{\text{ext}} \qquad \textbf{(Eq III.10)}$$

where the internal forces are given by $F_{int} = \alpha\nabla_u^2 x - \beta\nabla_u^2(\nabla_u^2 x)$ and the external forces are F $F_{ext} = -\nabla E_{ext}$The symbol $\nabla_u^2 = \frac{\partial^2}{(\partial u^{2})^2} + \frac{\partial^2}{(\partial u^{1})^2}$ is the Laplacian operator. Note that **in** Eq. (III.10) an auxiliary variable time t is introduced to make deformable surface x dynamic.

The comments made **in** Section III.3 about the global versus local nature of the solution of the deformable contours also applies to deformable surfaces.

III.5 Related Work

Although the name *deformable models* or *snakes* first appeared **in** work by Terzopoulos and his collaborators [4], the ideas of deforming an elastic template date back much further to the work of Fischler and Elschlager's springloaded templates [35] (1973) and Widrow's rubber mask technique [28] (1973). However, the popularity of deformable models to date is mostly credited to the work of "Snakes" by Kass, Witkin, and Terzopoulos [4] (1987).

There are basically two types of deformable models discussed **in** the literature: *parametric deformable models* and *geometric deformable models* [4,28]

Geometric deformable models, based on the theory of curve evolution and geometric flows, represents curves and surfaces implicitly as a level set of an evolving scalar function. Parametric deformable models, on the other hand, represents curves and surfaces explicitly in its parametric forms. In this thesis, we shall focus on parametric deformable models, although we expect our results to have applications in geometric deformable models as well.

Parametric deformable models synthesize parametric curves or surfaces within an image domain and allow them to move towards desired features, usually edges. Typically, the model is drawn toward the edges by *potential forces*, which are defined to be the negative gradient of a potential function. Additional forces, such as pressure forces together with the potential forces

comprise the external forces. There are also internal forces designed to hoId the model together (elasticity forces) and to keep it from bending excessively (bending forces).

There are two key difficulties with parametric deformable models. First, the initial model must, in general, be close to the true boundary or else it will likely converge to the wrong result. Several methods have been proposed to address this problem including multi-resolution methods, pressure forces, and distance potentials. The basic idea is to increase the capture range of the external force fields and to guide the model toward the desired boundary. The second problem is that deformable models have difficulties progressing into boundary concavities. Although pressure forces, control points, domain-adaptively, directional attractions, and the use of solenoidal fields have been proposed, these methods solve only one problem or both problems but with the priee of creating new difficulties. For example, multi-resolution methods have addressed the issue of capture range, but specifying how the deformable model should move across different resolutions remains problematic.

Another example is that of pressure forces, which can push a deformable model into boundary concavities, but cannot be too strong or "weak" edges will be overwhelmed. Pressure forces must also be initialized to push out or push in, a condition that mandates careful initialization [4].

The next title display difference types of GVF.

III.6 Types of GVF

There are three famous types of gradient vector flow (GVF):

1. Gradient vector flow: A new deformable model formulation for boundary mapping is developed. This new model uses a new type of external force called *gradient vector flow* (GVF). The GVF external force has three advantages compared to traditional external forces. First, it has a large capture range and allows deformable models to be initialized far away from the object boundary. Second, it can attract deformable models to move into boundary concavities where conventional methods have difficulties. Third, the GVF formulation is applicable to any dimension allowing it to be applied in a wide range of applications. This method is applied to both simulated images and real MR images.

2. Generalized gradient vector flow: Based on the GVF formulation, a generalization is developed to generate a family of vector fields that share similar properties as those of the GVF vector field. This generalization, called generalized gradient vector flow (GGVF), allows selection of external forces that are superior to the GVF external forces for certain applications. An important property called GGVF medialness, which results in an effective method for reconstructing the central layer of thick boundaries.

3. Gradient vector diffusion: proposed a new type of anisotropic diffusion equations to obtain the gradient vector field, which not only provides us a good guess of the initial snakes but also generates an external force on each pixel in the image domain. Unlike the diffusion equations, the diffusion scheme is based on the magnitudes and orientations of the vectors. This strategy greatly improves the behaviors of the vector diffusion when dealing with boundary concavities or "gaps" that Xu's [17] method and even its improved version cannot solve efficiently. The proposed gradient vector diffusion (GVD) scheme, coupled with the idea of multiple-snake, can correctly obtain an initial segmentation, and then a region merging approach is used to find the desired segmentation of the entire image .

III.7 Gradient Vector Flow Deformable Models

It is known that traditional deformable models have problems associated with initialization and poor convergence to boundary concavities. In this chapter, we present a new class of external forces for deformable models that addresses both problems listed above. These fields, which we call *gradient vector flow* (GVF) fields, are dense vector fields derived from images by minimizing a certain energy functional in a variational framework. The minimization is achieved by solving a pair of decoupled linear partial differential equations which diffuses the gradient vectors of a gray-leval or binary edge map computed from the image. We call the deformable models that uses the GVF field as its external force a *GVF deformable model*. The GVF deformable model is distinguished from nearly all previous deformable formulations in that its external forces cannot be written as the negative gradient of a potential function. Because of this, it cannot be formulated using the standard energy minimization framework; instead, it is specified directly from a force balance condition.

GVF can be defined in any dimension; however, in this chapter, we focus our attention on two-dimensional problems for convenience. We shall refer 2-D deformable models as deformable contours and those that use GVF as their external forces *GVF deformable contours*. We note that all the discussions on GVF deformable contours are valid to GVF deformable models in general.

Particular advantages of a GVF deformable contour over a traditional deformable contour are its insensitivity to initialization and its ability to move into boundary concavities. As we show in this chapter, its initializations can be inside, outside, or across the object's boundary. Unlike the balloon model, the GVF deformable contour does not need prior knowledge about whether to shrink or expand towards the boundary. The GVF deformable contour also has a large capture range, which means that, barring interference from other objects, it can be initialized far away from the boundary. This increased capture range is achieved through a diffusion process that does not blur the edges themselves, so multi-resolution methods are not

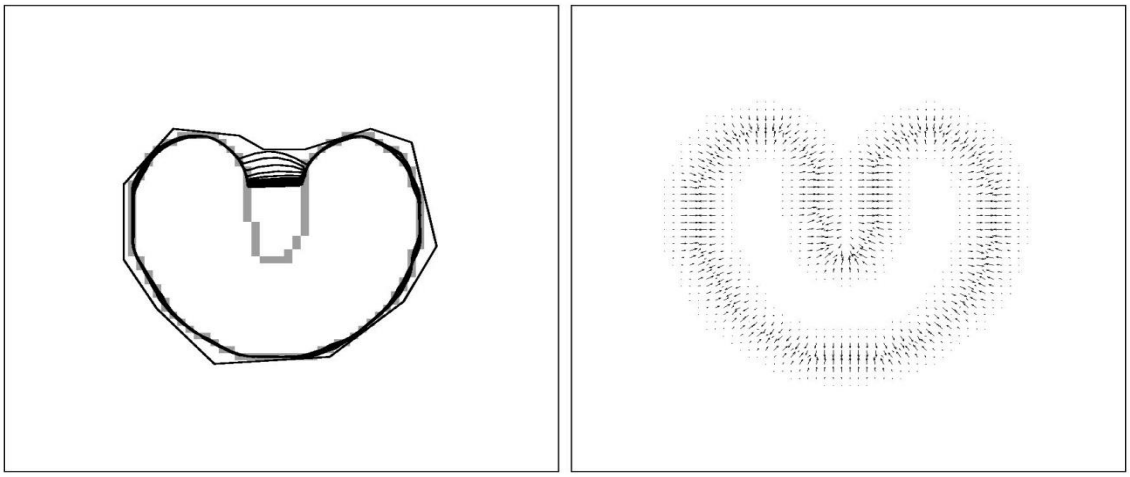
needed. The external force model that is closest in spirit to GVF is the distance potential forces of Cohen and Cohen. Like GVF, these forces originate from an edge map of the image and can provide a large capture range. We show, however, that unlike GVF, distance potential forces cannot move a deformable contour into boundary concavities. We believe that this is a property of all conservative forces which characterize nearly all deformable contour external forces, and that exploring non-conservative external forces, such as GVF, is an important direction for future research in deformable contour models.

III.7.1 Behavior of Traditional Deformable Contours

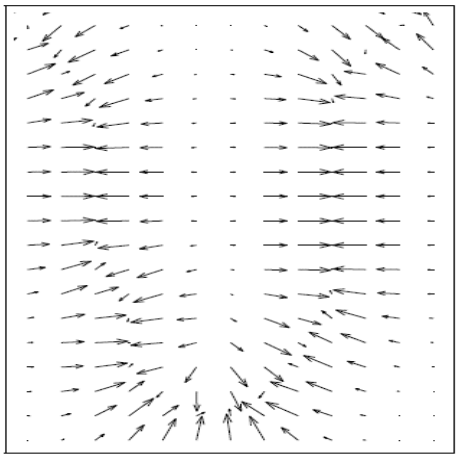
An example of the Behavior of a traditional deformable contour is shown in Fig. III.3. Fig. III.3.a shows a 64 x 64-pixelline-drawing of a U-shaped object (shown in gray) having a boundary concavity at the top. It also shows a sequence of curves (in black) depicting the iterative progression of a traditional deformable contour $(\alpha = 0.6, \beta = 0.0)$ initialized outside the object but within the capture range of the potential force field. The potential force field $F_{ext}^{(p)} = \nabla E_{ext}^{(4)}$ (defined in **(Eq III.5))** where $\sigma = 1.0$ pixel is shown in Fig.III.3.b. We note that the final solution in Fig. III.3.a solves the Euler equations of the deformable contour formulation, but remains split across the concave region.

The reason for the poor convergence of this deformable contour is revealed in Fig. III.1c, where a close-up of the external force field within the boundary concavity is shown. Although the external forces correctly point toward the object boundary, within the boundary concavity the forces point horizontally *in opposite directions.* Therefore, the deformable contour is pulled apart toward each of the "fingers" of the V-shape, but not made to progress downward into the concavity. There is no choice of α and β that will correct this problem.

Another key problem with traditional deformable contour formulations, the problem of limited capture range, can be understood by examining Fig. III.3.b. In this figure, we see that the magnitude of the external forces die out quite rapidly away from the object boundary. Increasing σ in **(Eq III.6)** will increase this range, but the boundary localization will become less accurate and distinct, ultimately obliterating the concavity itself when σ becomes too large.



(a) (b)



(c)

Fig III.3: (a) The convergence of a deformable contour using (b) traditional potential forces, (c) shown close-up within the boundary concavity. [26]

Cohen and Cohen [33] proposed an external force model that significantly increases the capture range of a traditional deformable contour. These external forces are the negative gradient of a potential function that is computed using a Euclidean (or chamfer) distance map. We refer to these forces as *distance potential forces* to distinguish them from the traditional potential forces defined in Section 2.1. Fig. III.4 shows the performance of a deformable contour using distance potential forces. Fig. III.4.a shows both the V-shaped object (in gray) and a sequence of contours (in black) depicting the progression of the deformable contour from its initialization far from the object to its final configuration. The distance potential forces shown in Fig. III.4.b have vectors with large magnitudes far away from the object, explaining why the capture range is large for this external force model.

As shown in Fig. III.4.a, this deformable contour also fails to converge to the boundary

concavity. This can be explained by inspecting the magnified portion of the distance potential forces shown in Fig. III.4.c. We see that, like traditional potential forces, these forces also point horizontally in opposite directions, which pulls the deformable contour apart but not downward into the boundary concavity. We note that Cohen and Cohen's modification to the basic distance potential forces, which applies a nonlinear transformation to the distance map, does not change the direction of the forces, only their magnitudes. Therefore, the problem of convergence to boundary concavities is not solved by distance potential forces.

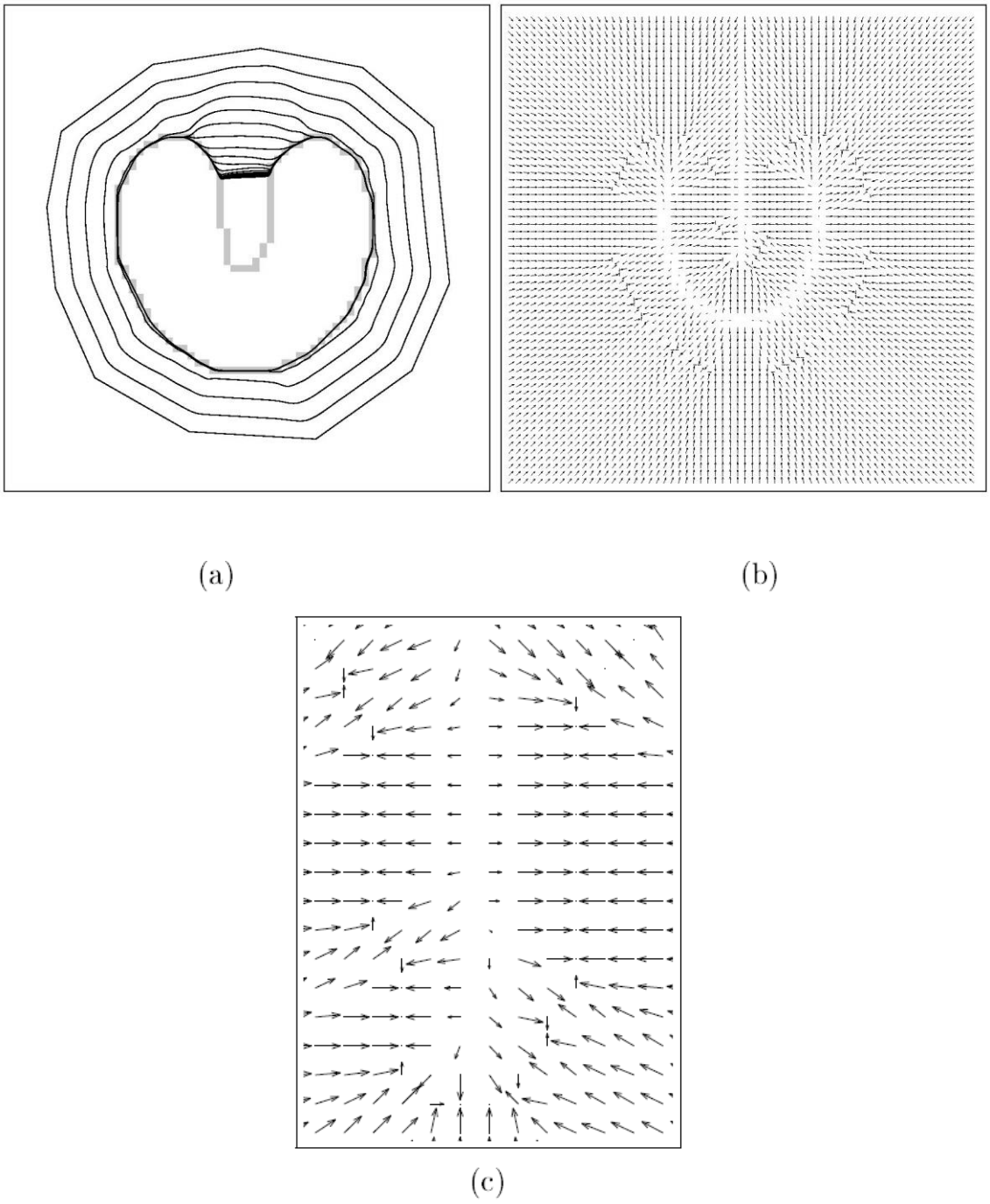


Fig III.4: (a) The convergence of a deformable contour using (b) distance potential forces, (c) shown close-up within the boundary concavity. [26]

III.7.2 Generalized Force Balance Equations

The deformable contour solutions shown in Figs. III.3.a and III.4 both satisfy the Euler equations **(Eq III.5)** for their respective energy model. Accordingly, the poor final configurations can be attributed to convergence to a local minimum of the objective function **(Eq III.2) .**Several researchers have sought solutions to this problem by formulating deformable contours directly from a force balance equation in which the standard external force $F_{ext}^{(p)}$is replaced by a more general external force $F_{ext}^{(g)}$ as follows

$$\mathbf{F}_{\text{int}} + \mathbf{F}_{\text{ext}}^{(g)} = 0 \qquad \textbf{(Eq III.11)}$$

The choice of $F_{ext}^{(g)}$can have a profound impact on both the implementation and the behavior of a deformable contour. Broadly speaking, the external forces $F_{ext}^{(g)}$can be divided into two classes: static and dynamic. Static forces are those that are computed from the image data, and do not change as the deformable contour progresses. Standard deformable contour potential forces are static external forces. Dynamic forces are those that change as the deformable contour deforms.

Several types of dynamic external forces have been invented to try to improve upon the standard deformable contour potential forces. For example, the forces used in multi-resolution deformable contours [37] and the pressure forces used in balloons [28] are dynamic external forces. The use of multi-resolution schemes and pressure forces, however, adds complexity to a deformable contour's implementation and unpredictability to its performance. For example, pressure forces must be initialized to either push out or push in, and may overwhelm weak boundaries if they act too strongly. Conversely, they may not move into boundary concavities if they are pushing in the wrong direction or act too weakly.

There is a type of *static external force,* one that does not change with time or depend on the position of the deformable contour itself. The underlying mathematical premise for this force comes from the Helmholtz theorem [38], which states that the most general static vector field can be decomposed into two components: an irrotational (curl-free) component and a solenoidal (divergence free) component. An external potential force generated from the variational formulation of a traditional deformable contour must enter the force balance equation **(Eq III.6)**as a static irrotational field, since it is the gradient of a potential function. Therefore, a more general static field $F_{ext}^{(g)}$can be obtained by allowing the possibility that it comprises *both* an irrotational component and a solenoidal component. In [39] explored the idea of constructing a separate solenoidal field from an image, which was then added to a standard irrotational field. In the

following section, we pursue a more natural approach in which the external force field is designed to have the desired properties of both a large capture range and the presence of forces that point into boundary concavities. The resulting formulation produces external force fields that can be expected to have both irrotational and solenoidal components [26].

III.7.3 Gradient Vector Flow Deformable Contour

Our overall approach is to use the force balance condition **(Eq III.7)** as a starting point for designing a deformable contour. We define below a new static external force field

$F_{ext}^{(g)} = v(x,y)$, which we call the *gradient vector flow* (GVF) field. To obtain the corresponding dynamic deformable contour equation, we replace the potential force $-\nabla E_{ext}$ in **(Eq III.8)** with $v(x, y)$, yielding

$$x_t(s,t) = \alpha x''(s,t) - \beta x''''(s,t) + v \qquad \textbf{(Eq III.12)}$$

We call the parametric curve solving the above dynamic equation a G*VF deformable contour*. It is solved numerically by discretization and iteration, in identical fashion to the traditional deformable contour.

Although the final configuration of a GVF deformable contour will satisfy the force-balance equation **(Eq III.7)**, this equation does not, in general, represent the Euler equations of the energy minimization problem in (III.5). This is because $v(x,y)$ will not, in general, be an irrotational field. The loss of this optimality property, however, is well-compensated by the significantly improved performance of the GVF deformable contour.

III.7.3.1 Edge Map

To define the GVF field, begin by defining an *edge map* $f(x,y)$ derived from the image $I(x,y)$ having the property that it is larger near the image edges." We can use any gray-level or binary edge map defined in the image processing literature [40]; for example, we could use

$$f(x,y) = -E_{ext}^{(i)}(x,y) \qquad \textbf{(Eq III.13)}$$

where $E_{ext}^{(i)}(x,y)$, $i = 1, 2, 3, or\ 4$, is the external energy defined in **(Eq III.2)**and **(Eq III.6)**. Three general properties of edge maps are important in the present context.

First, the gradient of an edge map ∇f has vectors pointing toward the edges, which are normal to the edges at the edges. Second, these vectors generally have large magnitudes only in the

immediate vicinity of the edges. Third, in homogeneous regions, where $I(x, y)$ is nearly constant, ∇f is nearly zero.

Now consider how these properties affect the behavior of a traditional deformable contour when the gradient of an edge map is used as an external force. Because of the first property, a deformable contour initialized close to the edge will converge to a stable configuration near the edge. This is a highly desirable property. Because of the second property, however, the capture range will be very small, in general. Because of the third property, homogeneous regions will have no external forces whatsoever.

These last two properties are undesirable. Our approach is to keep the highly desirable property of the gradients near the edges, but to extend the gradient map farther away from the edges and into homogeneous regions using a computational diffusion process. As an important benefit, the inherent competition of the diffusion process will also create vectors that point into boundary concavities.

III.7.3.2 Gradient Vector Flow

We define the gradient vector flow to be the vector field $v(x, y) = [u(x, y), v(x, y)]$ that minimizes the energy functional

$$\mathcal{E} = \iint \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |\mathbf{v} - \nabla f|^2 dxdy \qquad \textbf{(Eq III.15)}$$

This variational formulation follows a standard principle, which of making the result smooth when there is no data. In particular, we see that when $|\nabla f|$ is small, the energy is dominated by sum of the squares of the partial derivatives of the vector field, yielding a slowly-varying field. On the other hand, when $|\nabla f|$ is large, the second term dominates the integrand, and is minimized by setting $\boldsymbol{v} = \nabla f$. This produces the desired effect of keeping $\boldsymbol{v}$ nearly equal to the gradient of the edge map when it is large, but forcing the field to be slowly-varying in homogeneous regions. The parameter μ is a regularization parameter governing the trade-off between the first term and the second term in the integrand. This parameter should be set according to the amount of noise present in the image (more noise, increase μ).

We note that the smoothing term - the first term within the integrand of **(Eq III.14)** - is the same term used by Horn and Schunck in their classical formulation of optical flow [41]. It has recently been shown that this term corresponds to an equal penalty on the divergence and curl of the vector field. Therefore, the vector field resulting from this minimization can be expected to be neither entirely irrotational nor entirely solenoidal.

Using the *calculus of variations,* it can be shown that the GVF field can be found by solving the following Euler equations

$$\mu\nabla^2 u - (u - f_x)\left(f_x^2 + f_y^2\right) = 0 \qquad \textbf{(Eq III.15.a)}$$

$$\mu\nabla^2 v - \left(v - f_y\right)\left(f_x^2 + f_y^2\right) = 0 \qquad \textbf{(Eq III.15.b)}$$

where ∇^2 is the Laplacian operator. These equations provide further intuition behind the GVF formulation. We note that in a homogeneous region (where $I(x, y)$ is constant), the second term in each equation is zero because the gradient of $f(x, y)$ is zero. Therefore, within such a region, u and v are each determined by Laplace's equation, and the resulting GVF field is interpolated from the region's boundary, reflecting a kind of competition among the boundary vectors. This explains why GVF yields vectors that point into boundary concavities.

III.7.3.3 Numerical Implementation

Equations (3.5a) and (3.5b) can be solved by treating u and v as functions of time and solving

$$u_t(x, y, t) = \mu\nabla^2 u(x, y, t) - (u(x, y, t) - f_x(x, y))(f_x(x, y)^2 + f_y(x, y)^2) \qquad \textbf{(Eq III.16.a)}$$

$$v_t(x, y, t) = \mu\nabla^2 v(x, y, t) - (v(x, y, t) - f_y(x, y))(f_x(x, y)^2 + f_y(x, y)^2) \qquad \textbf{(Eq III.16.b)}$$

The steady-state solution of these linear parabolic equations is the desired solution of the Euler equations **(Eq III.15.a)** and **(Eq III.15.b).** Note that these equations are decoupled, and therefore can be solved as separate scalar partial differential equations in u and v. The equations in **(Eq III.16)** are known as *generalized diffusion equations,* and are known to arise in such diverse fields as heat conduction, reactor physics, and fluid flow. Here, they have appeared from our description of desirable properties of deformable contour external force fields as represented in the energy functional of **(Eq III.14).**

For convenience, we rewrite Equations **(Eq III.16)** as follows

$$U_t(x, y, t) = \mu\nabla^2 u(u, y, t) - b(x, y)\, u(x, y, t) + c^1(x, y) \qquad \textbf{(Eq III.17.a)}$$

$$V_t(x, y, t) = \mu\nabla^2 v(v, y, t) - b(x, y)\, v(x, y, t) + c^2(x, y) \qquad \textbf{(Eq III.17.b)}$$

where

$$b(x, y) = f_x^2(x, y) + f_y^2(x, y)$$

$$c^1(x, y) = b(x, y)\, f_x(x, y)$$

$$c^2(x, y) = b(x, y)\, f_y(x, y)$$

Any digital image gradient operator can be used to calculate *fx* and *fy*. In the examples shown in this paper, we use simple central differences. The coefficients *b(x, y), c^1(x, y),* and *c^2(x, y),* can then be computed and fixed for the entire iterative process.

To set up the iterative solution, let the indices i, j, and $\boldsymbol{ti}$ correspond to *x*, *y*, and t, respectively, and let the spacing between pixels be Δx and Δy and the time step for each iteration be Δt. Then the required partial derivatives can be approximated as:

$$u_t = \frac{1}{\Delta t} u_{i,j}^{n+1} - u_{i,j}^{n}$$

$$v_t = \frac{1}{\Delta t} v_{i,j}^{n+1} - v_{i,j}^{n}$$

$$\nabla^2 u = \frac{1}{\Delta x \Delta y}(u_{i+1,j} + u_{i,j+1} + u_{i-1,j} + u_{i,j-1})$$

$$\nabla^2 v = \frac{1}{\Delta x \Delta y}(v_{i+1,j} + v_{i,j+1} + v_{i-1,j} + v_{i,j-1})$$

Substituting these approximations into **(Eq III.17)** gives our iterative solution to GVF:

$$u_{i,j}^{n+1} = (1 - b_{i,j}\Delta t)u_{i,j}^{n} + r(u_{i+1,j}^{n} + u_{i,j+1}^{n} + u_{i-1,j}^{n} + u_{i,j-1}^{n} - 4u_{i,j}^{n}) + c_{i,j}^{1}\Delta t \qquad \textbf{(Eq III.18.a)}$$

$$v_{i,j}^{n+1} = (1 - b_{i,j}\Delta t)v_{i,j}^{n} + r(v_{i+1,j}^{n} + v_{i,j+1}^{n} + v_{i-1,j}^{n} + v_{i,j-1}^{n} - 4v_{i,j}^{n}) + c_{i,j}^{2}\Delta t \qquad \textbf{(Eq III.18.b)}$$

Where

$$r = \frac{\mu \Delta t}{\Delta x \Delta y} \qquad \textbf{(EqIII.19)}$$

Convergence of the above iterative process is guaranteed by a standard result in the theory of numerical methods [42]. Provided that b, c^1, and c^2 are bounded, **(Eq III.18)** is stable whenever the Courant-Friedrichs-Lewy step-size restriction $\mu \leq \frac{1}{4}$is maintained. Since normally Δx,Δy and μ are fixed, using the definition of r in **(Eq III.19)** we find that the following restriction on the time-step Δt must be maintained in order to guarantee convergence of GVF:

$$\Delta t \leq \frac{\Delta x \Delta y}{4\mu} \qquad \textbf{(Eq III.20)}$$

The intuition behind this condition is revealing. First, convergence can be made to be faster on coarser images - i.e., when Δx and Δy are larger. Second, when μ is large and the GVF is expected to be a smoother field, the convergence rate will be slower (since Δt must be kept small). [26]

III.7.4 Generalized Gradient Vector Flow

GVF has many desirable properties as an external force for deformable models. It still has difficulties, however, forcing a detection into long, thin boundary indentations. That this difficulty could be caused by excessive smoothing of the field near the boundaries, governed by the coefficient μ in eq:

$$v_t = \mu \nabla^2 v - (v - \nabla f)|\nabla f|^2 \qquad \textbf{(Eq III.21)}$$

We reasoned that introducing a spatially varying weighting function, instead of the constant μ, and decreasing the smoothing effect near strong gradients, could solve this problem. In the following formulation, which we have termed *generalized GVF* (GGVF), we replace both μ and $|\nabla f|^2$ in **(Eq III.21)**.

We define GGVF as the equilibrium solution of the following vector partial differential equation:

$$v_t = g(|\nabla f|)\nabla^2 v - h(|\nabla f|)(v - \nabla f) \qquad \textbf{(Eq III.22)}$$

The first term on the right is referred to as the *smoothing term* since this term alone will produce a smoothly varying vector field. The second term is referred as the *data term* since it encourages the vector field v to be close to ∇f computed from the data. The weighting functions g (.) and $h(.)$ apply to the smoothing and data terms, respectively. Since the se weighting functions are dependent on the gradient of the edge map which is spatially varying, the weights themselves are spatially varying, in general. Since we want the vector field v to be slowly varying (or smooth) at locations far from the edges, but to conform to $|\nabla f|$near the edges, g (.) and $h(.)$ should be monotonically non-increasing and non-decreasing functions of $|\nabla f|$, respectively.

The above equation reduces to that of GVF when

$$g(|\nabla f|) = \mu \qquad \textbf{(Eq III.23)}$$

$$h(|\nabla f|) = |\nabla f|^2 \qquad \textbf{(Eq III.24)}$$

Since $g(.)$ is constant here, smoothing occurs everywhere; however, $h(.)$ grows larger near strong edges, and should dominate at the boundaries. Thus, GVF should provide good edge localization. The effect of smoothing becomes apparent, however, when there are two edges in close proximity, such as when there is a long, thin indentation along the boundary. In this situation, GVF tends to smooth between opposite edges, losing the forces necessary to drive an active contour into this region.

To address this problem, weighting functions can be selected such that $g(.)$ gets smaller as $h(.)$ becomes larger. Then, in the proximity of large gradients, there will be very little smoothing, and the effective vector field will be nearly equal to the gradient of the edge map. There are many ways to specify such pairs of weighting functions. In this section, we use the following weighting functions for GGVF

$$g(|\nabla f|) = \exp(-\frac{|\nabla f|}{K}) \qquad \textbf{(Eq III.25)}$$

$$h(|\nabla f|) = 1 - g(|\nabla f|) \qquad \textbf{(Eq III.26)}$$

The GGVF field computed using this pair of weighting functions will conform to the edge map gradient at strong edges, but will vary smoothly away from the boundaries. The specification of *K* determines to soe extent the degree of tradeoff between field smoothness and gradient conformity.

As in GVF, the partial differential equation **(Eq III.22)** specifying GGVF, can be implemented using an explicit finite difference scheme, which is stable if the time stepΔt and the spatial sample intervals Δx and Δt satisfy

$$\Delta t \leq \frac{\Delta x \Delta y}{4G_{max}}$$

where G_{max} is the maximum value of $g(.)$ over the range of gradients encountered in the edge map image. While an implicit scheme for the numerical implementations of **(Eq III.25)** would be unconditionally stable and therefore not need this condition, the explicit scheme is faster. Still faster methods for example, the multigrain method are possible [43].

Resultat the Xu's Method and the GGVF:

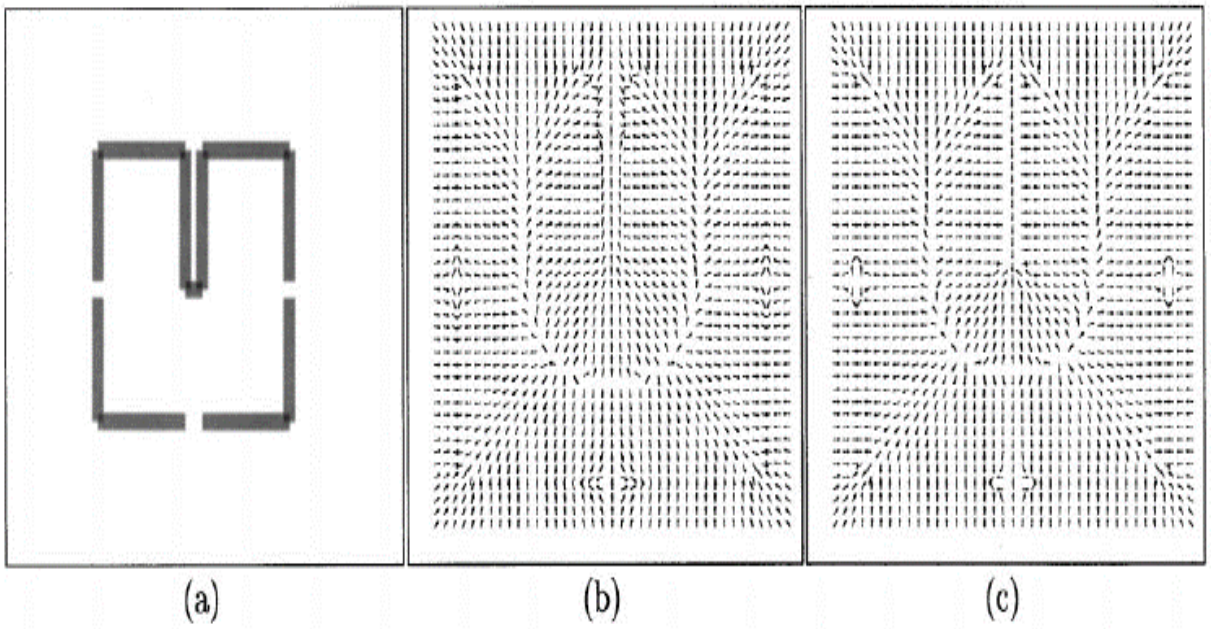


Fig.III.5. (a) A square with a long, thin indentation and broken boundary; (b) original GVF field (zoomed); (e) proposed GGVF field (zoomed); (d) initial snake position for both the GVF snake and the GGVF snake [43].

III.7.5. Gradient Vector Diffusion

III.7.5.1. Previous Work and Analysis

In the work of *Xu [17] etc.*, the following diffusion equations were used:

$$\begin{cases} \dfrac{du}{dt} = \mu\nabla^2 u - (u - f_x)(f_x^2 + f_y^2) \\ \dfrac{dv}{dt} = \mu\nabla^2 v - (v - f_y)(f_x^2 + f_y^2) \end{cases}$$

(Eq III.27.a)

(Eq III.27.b)

where (u, v) is initialized by $\nabla f(x, y)$ and $f(x, y)$ is an edge map of the original image: $f(x, y) = |\nabla I(x, y)|$

This diffusion model was originally proposed to remedy the problem of the traditional snake model in case of boundary concavities (see **Fig III.6** (a) where *ACB* are shown). However, it does not work well when the boundary concavities are long and thin. Then *Xu* [17]*etc.* proposed a generalized version of this model, aiming to handle the long and thin concavities. Unfortunately even the generalized version (GGVF) still could not handle these cases efficiently. The reason is that, as shown in fig. **Fig III.6** (a), then vectors propagated from *C* to *D* are very "weak" (with low magnitudes) while the vectors propagated from *A* and *B* to *D* are relatively much stronger, yielding the diffused vectors around *D* pointing either up or down. Another

problem with both models is how to prevent the snakes from moving out of the boundaries "gap" or low-contrast boundaries near high contrast boundaries as shown in **Fig III.6** (a) around point *G [36]*.

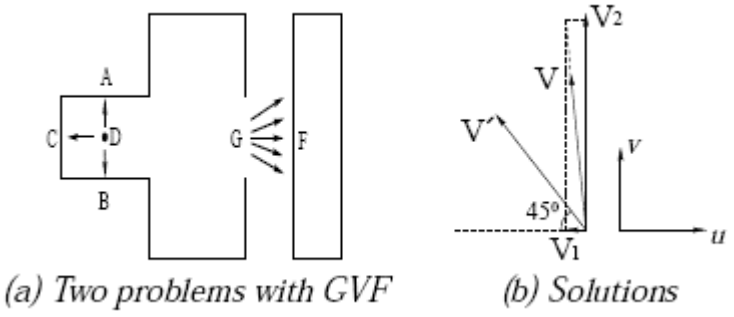


Fig III.6. Illustration of the problems and solutions. *V* and *V'* in (b) stand for the diffused vector by Xu's method and GVD method, respectively [34]

III.7.5.2 GVD Approach

Remember that GVF and GGVF apply the diffusion on Cartesian coordinate representation *(u* and *v)* of vectors. Actually, applying the diffusion on *r* (the magnitude of vector) and θ (the orientation of vector) can greatly improve the behaviors of the vector diffusion. **Fig III.6** (b) shows how these two different ways diffuse two vectors and have totally different results. For the diffusion scheme based on r and θ, a very weak vector can largely affect a strong vector, both on its magnitude and on its orientation. This is the essential idea for our new diffusion approach. Furthermore, by using an anisotropic scheme, we can choose to weaken the orientation diffusion (as we will see in the following) if two vectors are pointing in almost opposite directions. Then the problem of boundary "gaps" can be easily solved before we describe our diffusion equations, we shall give a definition of *sink,* which will be used in our algorithm:

Definition Given a vector field $y(i,j), i = 0,\dots,m-1;\ j = 0,\dots,n-1$, where m, n are the size of the image domain. The **sink** of a pixel A at (i,j), denoted by $sink(i,j)$, is defined as the total incoming amounts at A from all its neighbors minus the magnitude of the vector at A In the following we will use different schemes to diffuse the vector magnitude *r* and orientation

Our approach uses an anisotropic diffusion scheme for the diffusion process of *r*

$$\frac{dr}{dt} = \mu\nabla(s\left(\overrightarrow{v^{(0)}}\right)c_1(\|\nabla r\|\nabla r) - h(r^{(0)})(r - r^{(0)}) \qquad \textbf{(Eq III.28)}$$

where:

$$c_1(\|\nabla r\|) = \exp\left(-\frac{\|\nabla r\|^2}{K^2}\right), h\left(r^{(0)}\right) = 1 - \exp(-\frac{r^{(0)}}{K})$$

and

$$s\left(\overrightarrow{v(i,j)}\right) = \exp(-sink_+(i,j))$$

where $\boldsymbol{sink_+(i,j)}$ is equal to $\boldsymbol{sink(i,j)}$ if $\boldsymbol{sink(i,j)} \geq 0$. Otherwise it is equal to zero. In equation (3), $\boldsymbol{r^{(0)}} = \overrightarrow{\boldsymbol{v^{(0)}}}$ where $\overrightarrow{\boldsymbol{v^{(0)}}}$, stands for the initial vector field that can be determined by ∇f. According to the definition, the *sink* is much larger around the boundary points than elsewhere. So $s\left(\overrightarrow{v^{(0)}}\right)$ can reduce the diffusion effect around boundary points while encouraging the diffusion elsewhere.

The diffusion equation for vector orientation θ plays an important role on making the vector field more "sensitive" to the boundary concavities but less "sensitive" around the boundary gaps. We shall assume that the orientations of the vectors are periodically defined on $[-\pi, \pi]$ the equation for vector orientation θ looks similar to (3

$$\frac{d\theta}{dt} = \mu\nabla(s\left(\overrightarrow{v^{(0)}}\right) c_2(dif(\theta, \theta^*))dif(\theta, \theta^*)sign(\theta, \theta^*) - h(r^{(0)})dif(\theta, \theta^*)sign(\theta, \theta^*)$$

(Eq III.29)

where θ^* is the orientation of one of the neighbors. $s\left(\overrightarrow{v^{(0)}}\right)$ and $h(.)$ are defined same as in (3). But $c_2(.)$ is different from $c_1(.)$ in **(Eq III.28)** :

$$c_2(x) = \begin{cases} \exp\left(-\frac{2x}{\pi}\right) & if\ 0 \leq x \leq \frac{\pi}{2} \\ \frac{2}{\pi e}\frac{(x-\pi)^2}{x} & if\ \frac{\pi}{2} \leq x \leq \pi \end{cases}$$

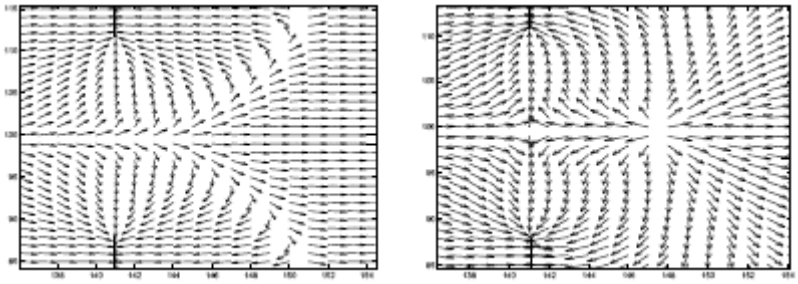
This weighting function basically tells us that, if the difference of two vectors' orientations $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$is about $\frac{\boldsymbol{\pi}}{\boldsymbol{2}}$ then they can affect each other most significantly. But if the difference is a little bit greater than $\frac{\boldsymbol{\pi}}{\boldsymbol{2}}$ their influence to each other will rapidly decrease to zero.

In (4) we replace $\nabla\theta$ (as it should be in most anisotropic diffusion methods) with $dif(\theta, \theta^*)sign(\theta, \theta^*)$.The reason for this change is that in our case the orientation θ is periodically defined on $[-\pi, \pi]$ .So the difference between two angles $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$can not be simply

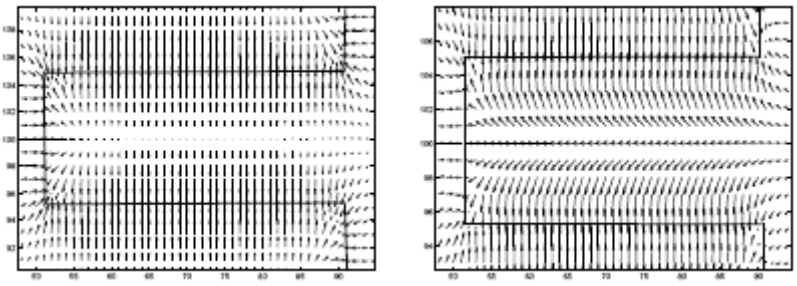
written as$|\boldsymbol{\theta_1} - \boldsymbol{\theta_2}|$, but should be the angle between the two corresponding vectors. And this angle could be positive or negative depending on the orientations of these two vectors. Therefore, the well-known *Maximum Principle* in traditional heat diffusion scheme is no longer true in the case of orientation diffusion [44].

III.7.5.3. Comparison with Xu's Method:

Fig III.7 *(a)* and *(b)* show the enlarged vector field near the boundary concavity (see Fig III.6 (a) around *ACB).* It is clearly shown that, by GVD approach, the vectors in the concavity obviously changed their magnitudes and orientations and thus the snake can easily move into the bottom of the concavity. Fig III.7 (c) and *(d)* show the enlarged vector field near the boundary gap (see Fig III.6 (a) around *G).* As we can see *from* Fig III.7 *(c),* traditional GVF has difficulty preventing the vectors near the boundary gap from being significantly influenced by the nearby boundaries, so that the snake may move out of the boundary gap. However, this approach can avoid this problem [44]



(a) Xu's method (b) GVD method



(c) Xu's method (d) GVD method

Fig III.7. The comparison between the traditional GVF method and GVD approach for their behaviors on the boundary concavity and boundary gap [44]

III.8. Conclusion

In this chapter we present Energy of image (Focus External Force GVF and present different forms), and study of different models deformable.

There are many of forms of GVF we present in this chapter four of them (Traditional GVF, GVF of Xu, GGVF and GVD), but there are also others I like to reference them: Hsu et al, [45] introduced a vector field inspired from the Poisson equation, the Poisson Gradient Vector Field (PGVF). This field has a similar diffusion as the one obtained with the GGVF, avoiding the problem of the stability condition and the smoothing parameter mentioned in [46] to iteratively solve the equation of the force field, since we don't need to introduce time variable to solve the Poisson equation. Another improvement was proposed by Li. et al. [47], the Edge Preserving Gradient Vector Flow (EPGVF), for preserving weak edges, they showed that both GVF and GGVF have difficulties to stop the snake at. Weak edges, when they are located near a strong one. Recently, Li and Acton in [48] presented a new vector field the Vector Field Convolution (VFC), as its name suggests, it is computed by a simple convolution of the contour map, derived from the gradient of the image, with a kernel vector .The main advantage of using the VFC is that the computational cost is reduced and good results can be obtained in the presence of noise.

Finally we conclude from this chapter the area of GVF are wide field and always evolve to solution the problems to access the perfect detection.

CHAPTER IV

B-SNAKE

IV.1 Introduction

In this chapter, we present a novel adaptive B-Snake model for object contour extraction. A cubic B-Snake model is developed for extracting 2D deformable objects from medical images, with an adaptive control point insertion algorithm that is suggested to increase the flexibility of B-Snake to describe complex shape. This method overcomes the problems that exist in other B-spline based model that have to decide beforehand or exhaustively search over a range of value for the number of control points. Hence, these methods are less flexible to describe unknown complex shapes. A minimum energy method which we called Minimum Mean Square Error (MMSE) is proposed for B-Snake to push it to the target boundary. The internal forces are not required in deforming B-Snake since the representation of B-Spline has guaranteed smoothness by hard implicit constraints. The proposed B-Snake model has been tested on object contour extraction such as human brain ventric1e in Magnetic Resonance (MR) images. The experimental results demonstrate the capability of adaptive shape description and object contour extraction.

IV.2 Advantages of B-snake model to the other Snakes models

The closed-form adaptive B-Snake model [49] with a knot insertion strategy and a Minimum Mean Square Error (MMSE) approach. It possesses two main novelties:

(1) The proposed B-Snake model has the capability to increase the number of knots (or control points) to form a complex shape to adapt the structure of the objects while still has a small number of parameters. This is in contrast to most B-Snake models, which have to fix the value or search over a range of values or depend on a ratio of the length of data for the number of control points, thus lacking the flexibility.

(2)A method of Minimum Mean Square Error (MMSE) is developed to iteratively estimate the position of those control points in the B-Snake model. As MMSE deforms the segments of the B-Spline instead of individual points, it is very robust against local minima.

And her advantages comparison to other models snakes are:

B-Snake uses far lesser parameters compared to point-based Snake model. As the cubic B-Spline is proposed to be used, deformable curves are represented by four or more parameters (control points).

Internal forces are not required in deforming B-Snake since the representation of B-Spline has guaranteed smoothness by hard implicit constraints.

The B-Snake model is also easy for object boundary tracking application since the parameters of B-Snake for the current frame are usually similar to those in the previous frame, i.e. the movement of the control points are smaller [50].

IV.3 B-Snake

In active contour models, a contour is initiated on the image and is left to deform in a way that, firstly, moves it toward features of interest in the image and, secondly, maintains a certain degree of smoothness and continuity in the contour. In order to favor this type of contour deformation, an energy term is associated with the contour and is designed to be inversely proportional to smoothness of the contour and to be fit to desired image features.

The deformation of the contour in the image plane will change its energy, thus one can imagine an energy (potential) surface on top of which the contour moves seeking valleys of low energy. The traditional Snake is a point-based deformable model. From the original philosophy of Snake, B-Snake is using a B-Spline in place of the original elastic rod simulation. The B-Splines can be constructed of any order (with the complexity of the possible conformation increasing with spline order), but third order (cubic) is most often encountered (See chapter 2). .

The cubic B-Splines exhibit a favourable trade-off between their shape complexity for describing natural curves and computational burden required to solve them. Therefore, we are concerned with the implementation of cubic B-Spline in this chapter. The uniform cubic B-Spline is a simple and particularly useful class of B-Spline, the term uniform means that the knots are equally spaced [50].

IV.2.1 Uniform Close Cubic B-Splines:

For a given set of $n+1$ control points, $\{Q_i = [x_i, y_i]^T, i = 0,1, \dots, n\}$ the close uniform cubic B-Spline consists of $n+1$ connected curve segments $\{g_i(s) = (x_i(s), y_i(s)), i = 1, 2, \dots, n+1\}$. Each curve segment is a linear combination of four cubic polynomials by the parameter s, where s is normalized between 0 and 1 ($0 \leq s \leq 1$). That is, [50].

$$g_i(s) = M_R(s)\begin{bmatrix} Q_{(i-1)mod(n+1)} \\ Q_{(i)mod(n+1)} \\ Q_{(i+1)mod(n+1)} \\ Q_{(i+2)mod(n+1)} \end{bmatrix}, \text{ for i= 1, 2,…, n+1} \qquad \text{(Eq IV.1)}$$

where

$$M_R(s) = [s^3\ s^2\ s\ 1]\begin{bmatrix} \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & -1 & \frac{1}{2} & 0 \\ -\frac{1}{6} & \frac{1}{2} & \frac{1}{6} & -\frac{1}{2} \end{bmatrix} \qquad \text{(EqIV.2)}$$

IV.2.2 The B-Snake Model

The cubic B-Snake is defined as follows:

$$r(s) = \sum_i g_i\,(s), where 0 \le s \le 1 \qquad (EqIV.3)$$

The control points are positioned so that the curve locates within the desired object contour. To identify the boundaries of an object in an image, the total energy function must be minimized along the B-Spline curve. Consequently, when equilibrium is achieved, the forces vanished in the image, r(s) is stabilized close to the contour of the object [51].

The external energy term on $r(s)$ is defined as $E(r(s))$.

There are no internal forces since the B-Spline representation maintains smoothness via hard constraints implicit in the representation. Therefore, the total energy function of the B-Snake $E_{B-Snake}$ can be defined by integrating $E(r(s))$ along the B-Snake. That is,

$$E_{B-Snake} = \int_0^1 E(r(s))\,ds \qquad \text{(EqIV. 4)}$$

To identify the edge features in the image, the above expression must be minimized along the B-Spline curve. So that, at equilibrium (when image forces vanish), $\mathrm{r(s)}$ stabilizes close to a contour of object.[50]

The points connecting the neighbouring segments are called the knot points $Pi(i = 1,2,\dots,n-1)$, where the B-Spline bases are tied together. Given the set of knot points $P = (P_1, P_2, \dots, P_{n-1})$ of a uniform cubic B-Spline curve, we can uniquely determine control

points $Q = (Q_0, Q_1, \dots, Q_n)$), by substituting s = 0 into (Eq IV.1). The relationship between the control points and the knot points is given as:

$$Q = A^{-1}P \qquad \text{(EqIV.5)}$$

where

$$A = \begin{bmatrix} \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & \dots & \dots & 0 \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & \dots & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ \frac{1}{6} & \dots & \dots & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ \frac{2}{3} & \frac{1}{6} & \dots & 0 & 0 & \frac{1}{6} & \frac{2}{3} \end{bmatrix} \qquad \text{(Eq IV.6)}$$

IV.3 Estimation of B-Snake Parameters from Image Data

Based on the initial location of the control points that are determined either by manual or as a result of previous frame, the B-Snake would further deform to the desired contour accurately in the current frame. In [52], Wang et al. have presented an open B-Snake lane model with a fixed number of control points, and Gradient Vector Flow (GVF) (See Chapter3) has been used as its external force to detect the lane boundary by MMSE method. In this Section, a new method for B-Snake model is presented. Compared to other B-Snake models, ithas a few novelties: (1) a control point insertion strategy is embedded to adapt the complex shape without prior fixing the number of control points; (2) the normal portion of GVF is chosen to efficiently deform the B-Snake; (3) MMSE method is implemented to not only the open B-Snake, but also the close B-Snake [51].

IV.3.1. External Force for B-Snake

Gradient Vector Flow (GVF) [53] (More details in Chapter III) is selected here as the definition of external forces for B-Snake, since GVF has a larger capture range. It is computed as a diffusion of the gradient vector of a gray-level or binary edge map derived from the image.

$$E_{ext} = \sum_i \sum_{s=0}^{1} (E_i(s)) \qquad \text{(Eq IV. 7)}$$

IV.3.2. Minimum Mean Square Error Approach

B-Snake should be updated to minimize: (1) the sum of the external forces from the B-Spline for achieving accurate position of B-Snake, and (2) the external forces should be transmitted to each control point when updating B-Snake. The details are described as follows

To deform the B-Snake to the object boundaries, the energy function has to be minimized as:

$$E_{ext} = \sum_{i} \sum_{s=0}^{1} (E_i(s)) = 0 \qquad \text{(Eq IV. 8)}$$

In practical application, the energy function would never reach zero, but a minimal value. Once the energy function of B-Snake is minimized, then no movement is driven for control points and hence no change in the shape of the B-Snake. To iteratively minimize the external force, the following equation shows the relationship between the movement of control points and the energy function:

$$E_{\text{ext}} = \gamma M_R(s) \Delta Q(t) \qquad \text{(Eq IV. 9)}$$

where γ is a step-size and $\Delta Q(t)$ is the adjustment of the control points Q in each iteration step.

$$Q(t) = Q(t-1) + \Delta Q(t) \qquad \text{(Eq IV. 10)}$$

External force can be sampled along the B-Spline of B-Snake at a certain distance (we are using 2-3 pixels). Then Eq. (IV. 9) can be solved digitally. Here, the MMSE solution for the digital version of the Eq. (IV. 9) is given as a matrix form [49]

$$\Delta Q(t) = \gamma^{-1} [M^T M]^{-1} M^T E_{\text{ext}} \qquad \text{(Eq IV.11)}$$

$$M = \begin{bmatrix} M_1 & 0 & \cdots & \cdots & 0 \\ 0 & M_2 & 0 & \cdots & 0 \\ . & . & . & . & . \\ 0 & \cdots & 0 & M_{n-3} & 0 \\ 0 & \cdots & \cdots & 0 & M_{n-2} \\ M'_{n-1} & 0 & \ldots & 0 & M_{n-1} \\ M'_n & 0 & \ldots & 0 & M_n \\ M'_{n+1} & 0 & \ldots & 0 & M_{n+1} \end{bmatrix}$$

$$M_i = \begin{bmatrix} s_{i,1}^3 & s_{i,1}^2 & s_{i,1} & 1 \\ s_{i,2}^3 & s_{i,2}^2 & s_{i,2} & 1 \\ . & . & . & . \\ s_{i,m}^3 & s_{i,m}^2 & s_{i,m} & 1 \end{bmatrix} \times \begin{bmatrix} -\frac{1}{6} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{6} \\ \frac{1}{2} & -1 & \frac{1}{2} & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \end{bmatrix}, i = 1, 2, \ldots, n-2.$$

$$M_{n-1} = \begin{bmatrix} s_{n-1,1}^3 & s_{n-1,1}^2 & s_{n-1,1} & 1 \\ s_{n-1,2}^3 & s_{n-1,2}^2 & s_{n-1,2} & 1 \\ . & . & . & . \\ s_{n-1,m}^3 & s_{n-1,m}^2 & s_{n-1,m} & 1 \end{bmatrix} \times \begin{bmatrix} 0 & -\frac{1}{6} & \frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{1}{2} & -1 & \frac{1}{2} \\ 0 & -\frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{bmatrix},$$

$$M'_{n-1} = \begin{bmatrix} s_{n-1,1}^3 & s_{n-1,1}^2 & s_{n-1,1} & 1 \\ s_{n-1,2}^3 & s_{n-1,2}^2 & s_{n-1,2} & 1 \\ . & . & . & . \\ s_{n-1,m}^3 & s_{n-1,m}^2 & s_{n-1,m} & 1 \end{bmatrix} \times \begin{bmatrix} \frac{1}{6} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M_n = \begin{bmatrix} s_{n,1}^3 & s_{n,1}^2 & s_{n,1} & 1 \\ s_{n,2}^3 & s_{n,2}^2 & s_{n,2} & 1 \\ . & . & . & . \\ s_{n,m}^3 & s_{n,m}^2 & s_{n,m} & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & -\frac{1}{6} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & -1 \\ 0 & 0 & -\frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{6} & \frac{2}{3} \end{bmatrix},$$

$$M'_n = \begin{bmatrix} s_{n,1}^3 & s_{n,1}^2 & s_{n,1} & 1 \\ s_{n,2}^3 & s_{n,2}^2 & s_{n,2} & 1 \\ . & . & . & . \\ s_{n,m}^3 & s_{n,m}^2 & s_{n,m} & 1 \end{bmatrix} \begin{bmatrix} -\frac{1}{2} & \frac{1}{6} & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{6} & 0 & 0 & 0 \end{bmatrix}$$

$$M_{n+1} = \begin{bmatrix} s_{n+1,1}^3 & s_{n+1,1}^2 & s_{n+1,1} & 1 \\ s_{n+1,2}^3 & s_{n+1,2}^2 & s_{n+1,2} & 1 \\ . & . & . & . \\ s_{n+1,m}^3 & s_{n+1,m}^2 & s_{n+1,m} & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & -\frac{1}{6} \\ 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{6} \end{bmatrix},$$

$$M'_{n+1} = \begin{bmatrix} s_{n+1,1}^3 & s_{n+1,1}^2 & s_{n+1,1} & 1 \\ s_{n+1,2}^3 & s_{n+1,2}^2 & s_{n+1,2} & 1 \\ . & . & . & . \\ s_{n+1,m}^3 & s_{n+1,m}^2 & s_{n+1,m} & 1 \end{bmatrix} \times \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & \frac{1}{6} & 0 \\ -1 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ \frac{2}{3} & \frac{1}{6} & 0 & 0 \end{bmatrix}$$

and $s_{i,m}$is the parameter s to form the sampling point m in ith segment [49].

VI.3.3 Control Point Insertion Strategy

To adaptively approach a comp1ex contour without prior fixing of the number of control points, a structure adaptive knot point insertion strategy is developed. The location of new knot point is determined by checking the external force on the B-Snake when B-Snake is converged in the deforming procedure. The new knot point is inserted into the location where it has maximal external force on the B-Spline curve. In the GVF field, the magnitude of the force reflects the distance to the nearest significant edge. The stronger external force means the further distance to the nearest object boundary. In this case, the knot point should be inserted to increase the flexibility of the B-Spline curve to adapt to the object shape. Compared to other algorithm [3] which has to try a range of values to determine the best number of control points for B-Snake model, our method is simple and working fast.

Suppose that we have a B-Spline curve defined on a sequence of control points $\mathrm{Q} = \mathrm{Q}_0, \mathrm{Q}_1, \ldots, \mathrm{Q}_\mathrm{n})$, a knot P_t is expected to insert between P_i and $\mathrm{P}_{\mathrm{i}+1}$ at s. The new set of control points is $\widehat{\mathrm{Q}} = \widehat{\mathrm{Q}}_0, \widehat{\mathrm{Q}}_1, \ldots, \widehat{\mathrm{Q}}_\mathrm{n}, \widehat{\mathrm{Q}}_{\mathrm{n}+1}$. is determined by

$$\hat{Q}_j = (1 - \alpha_j) Q_{j-1} + \alpha_j Q_j \qquad \text{(Eq IV.15)}$$

where:

$$\alpha_j = \begin{cases} 1 & (j \le i - 1) \\ \frac{s_t + 2 + (j - i)}{3} & (i \le j \le i + 2) \\ 0 & (j \ge i + 3) \end{cases} \qquad \text{(Eq IV.16)}$$

Equations (Eq IV.15) and (Eq IV.15) mean that the new $\hat{Q}_j$ divides the line segments $Q_{j-1}\, Q_j$ in the ratio $\alpha_j : (1 - \alpha_j)$ [50].

VI.3.4. the Complete Object Contour Extraction Algorithm

In order to arrive at the solutions in Eq. (13), an iterative procedure is adopted. This iterative minimization is guaranteed to converge at 1east to a local minimum of $_{\text{ER-snake}}$ in Eq. (6). The steps contained in this iterative minimization process are as follows:

(1)Ca1culate the edge image of original image by using Canny edge detector.

(2) Ca1culate the GVF of the edge image as the external force of B-Snake.

(3) Initialize the control point parameters.

(4) Compute the MMSE in (Eq IV.11) for obtaining $\Delta Q(t)$

(5)If $\|\Delta Q(t)\| > threshold^1$, set $Q(t)$ to $Q(t-1)$, then go to step 4; Otherwise, go to step 6.

(6)If max $\|E_{B-Snake}\| > threshold^2$, a new knot point is inserted in that location which has maximum external force, then go to step 4; Otherwise, go to step 7.

(7)Stop. The last estimations of $Q(t)$ are regarded as the final results.

This iterative minimization is guaranteed to converge to at least a local minimum of E_{ext} in (Eq IV. 9) . There is no exact mie to set the thresholds. However, if thresholds are set too strictly, although B-Snake would match object very precisely, the processing time would be longer. Therefore, based on our experience,$threshold^1$ and $threshold^2$, are set to 0.1 pixel and 0.3, respectively. It is the best tradeoff between computational time and matching accuracy [50].

IV.4.Conclusion

Based on the initial location of the control points, the B-Snake would further deform to object edge accurately in the current frame. The minimum energy method MMSE that finds the correspondence between B-Snake and the real edge image is implemented in this B-Snake model to determine the parameters iteratively to deal with this problem. Normal external forces are sampled in the B-Spline by a limit distance to each other and are applied to the curve itself. But, for iterative adjustment of displacement, it is necessary to compute the force transmitted to each control point. This is done by positioning the B-Spline so that it minimizes the sum of the external force of the B-Snake model. Since the B-Snake representation maintains smoothness via hard constraints implicit in the representation, the internal forces are not required. The GVF is chosen as the external force for B-Snake to object boundary detection since it has a larger capture range. The structure adaptive capability of our B-Snake model is reached by the strategy of adaptively inserting control points during the B-Snake deformation procedure.

This method overcomes the problems that exist in other B-spline based models that have to decide beforehand or exhaustively search over a range of values or depend on a ratio of the length of data for the number of control points, and hence lack flexibility to describe unknown complex shapes when initially too few control points are chosen. The proposed method has been

implemented to various objects' shape extraction. The experimental results show that the proposed method is robust and accurate in object contour extraction. Currently work is done to extending the 2D model to 3D by developing a new control point insertion algorithm and deformation strategy for B-Surface.

However, in the B-Snake model presented in this paper, no prior knowledge of object shape has been used. For the case where a prior information is available, such as the geometric and appearance information of the object shape, current research makes it possible to be implemented into Snake model. In our future work, these information will be employed in B-Snake model to improve the object extraction.

CHAPTER V

RESULTS AND CONCEPTION

V.1 Introduction

In this chapter we have applied the method of B-snake, where we have presented our results and compared them with theoretical results.

All steps of this work was shown and depending to the Main organigram (Fig V1)

The logicial used here is matlab 2011a, and the proprieties of our PC are (PC-Lenovo, Processor Pentium R(Duel-Cor) CPU T4400 2.20Ghz 2.20Ghz,RAM 2.0Ghz,System 32bits, Windows 7 Edition Integrate SPA).

V.2 Representation of the Matlab

MATLAB: Faster startup on Windows with Startup Accelerator, reading and writing portions of arrays from MAT-files, and new spreadsheet import tool

Parallel Computing Toolbox: Increase in local workers from 8 to 12

Image Processing Toolbox: Parallel block processing of large images with Parallel Computing Toolbox

Global Optimization Toolbox: Mixed integer nonlinear programming in genetic algorithm solver

Statistics Toolbox: Lasso and elastic net for linear regression variable selection from high dimensional data sets

Financial Derivatives Toolbox: Pricing and sensitivity calculations for sinking fund provisions, range bonds, and step up/down coupon bonds

Data Acquisition Toolbox: Measurement support for IEPE accelerometers

Instrument Control Toolbox: Bluetooth serial communication support

Bioinformatics Toolbox: NGS Browser for viewing multiple tracks of sequence alignment data stored in SAM/BAM formats

Robust Control Toolbox: Automatic tuning of arbitrary controller architectures

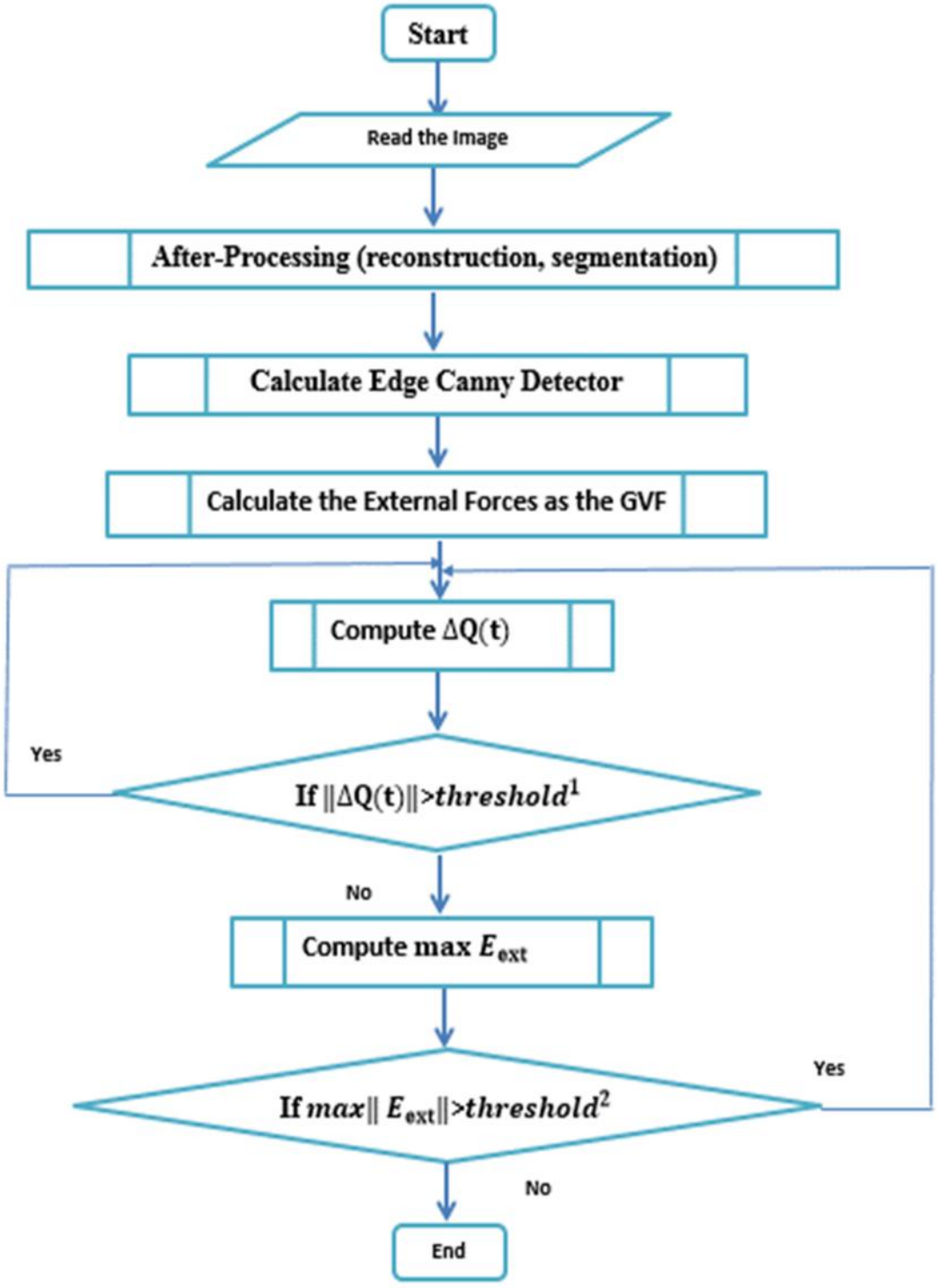


Fig V.1: The Main Flow Chart of our Work

V.3 Conception and Results

V.3.1 Read the Image

Input the image data (IRM) from matlab is the first work of the detection. Fig: V.2 present an MRI image for execution the detection active on it.

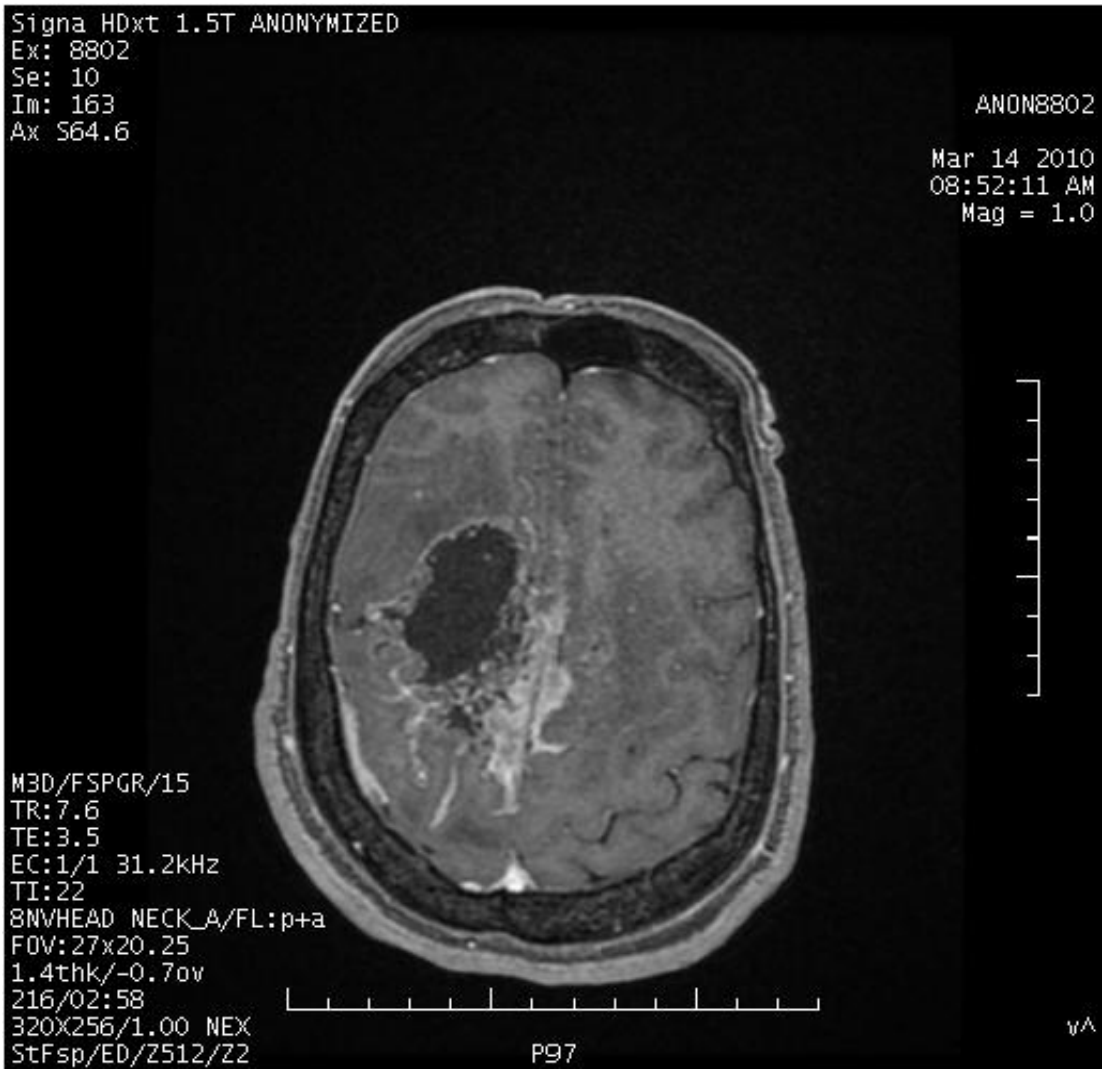


Fig: V.2 Read data of MRI image from matlab

In this step we must transform the data of image to binary for fast calculate the next steps of our results.

V.3.2 After Processing

This step depend to two main steps:

1-Segmentation the image (3D) to (2D): we look the same image in Fig V.2 but without third axes (z).see fig(V.3). But in the data of the image decrease to the third.

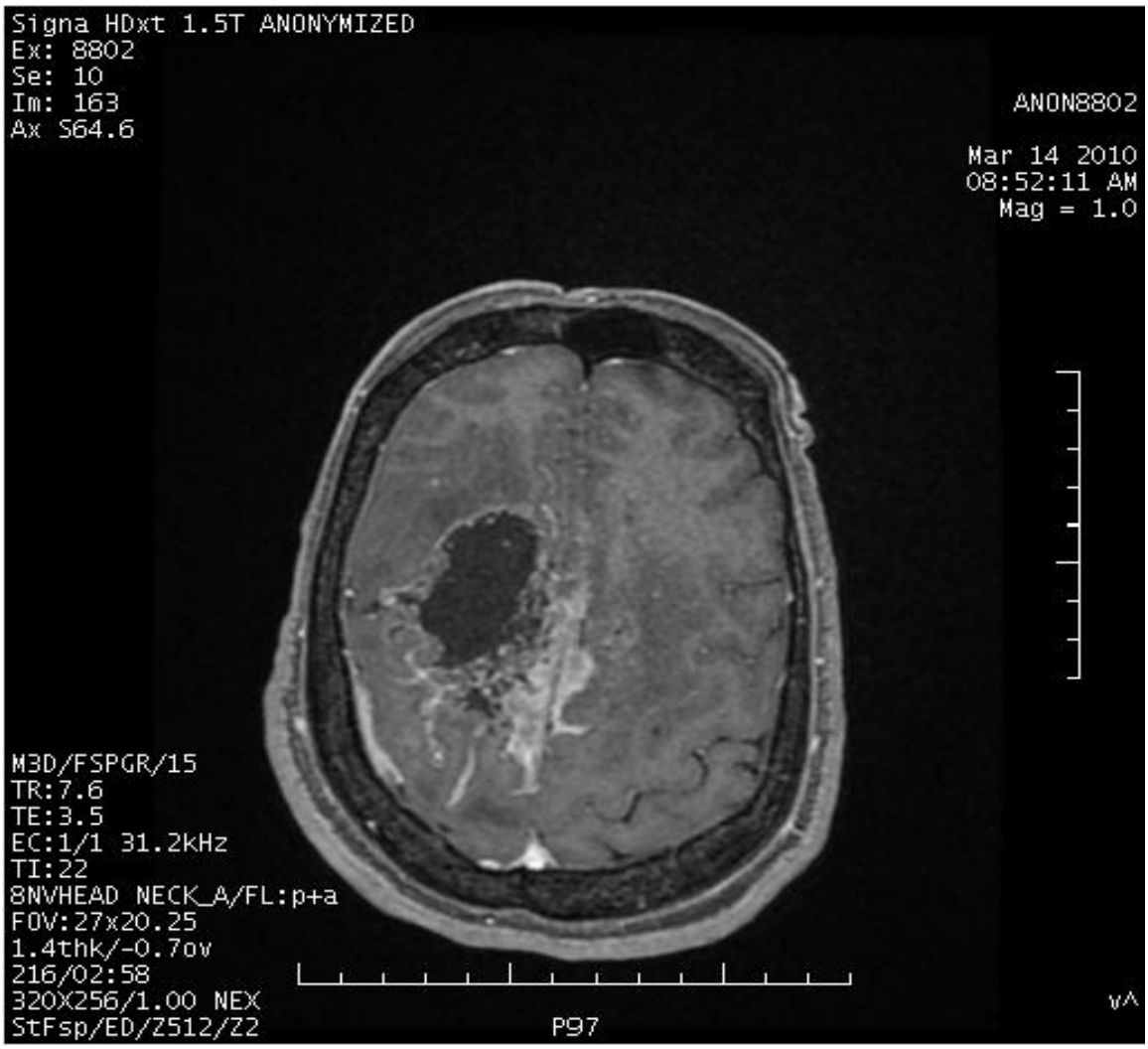


Fig V.3: translate image from 3D to 2D

2-Filtre the Image: This step used for delete the deformation the image by using the reconstruction of mathematic morphology to quick computation the results. and it applied with many technic (project of Mouaki) because this information affect at the data of image that we used before to calculate the (GVF), the Fig V.4 present the Fig V.3 before filtrate the image.

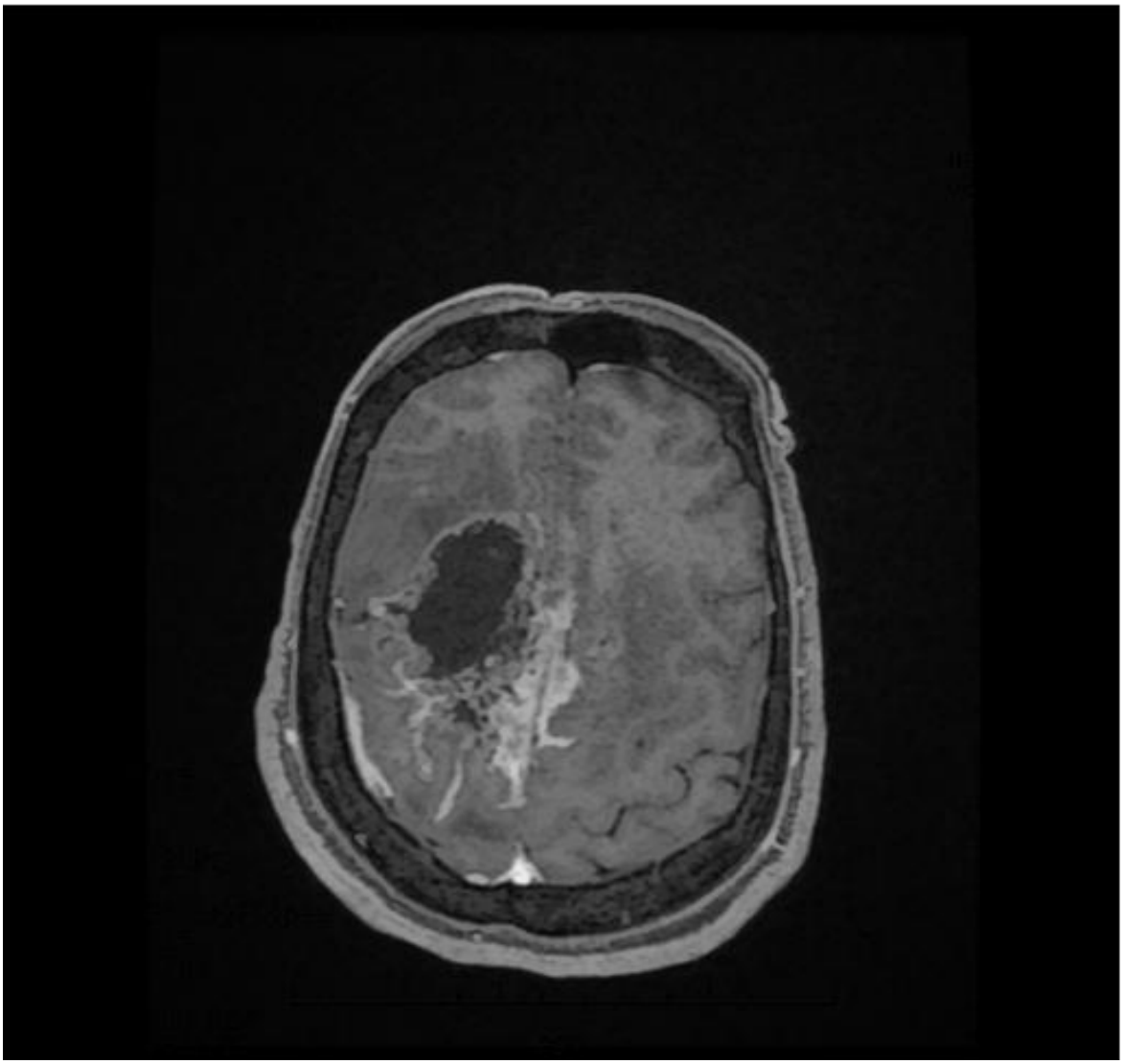


Fig V.4: IRM before filtrate.

V.3.3 Calculate the “Edge Canny”

Before any execution of force we need to translate image to binary for that we have choose canny edge.). Its algorithm was shown in Table I.1.Fig (V.5) present canny edge

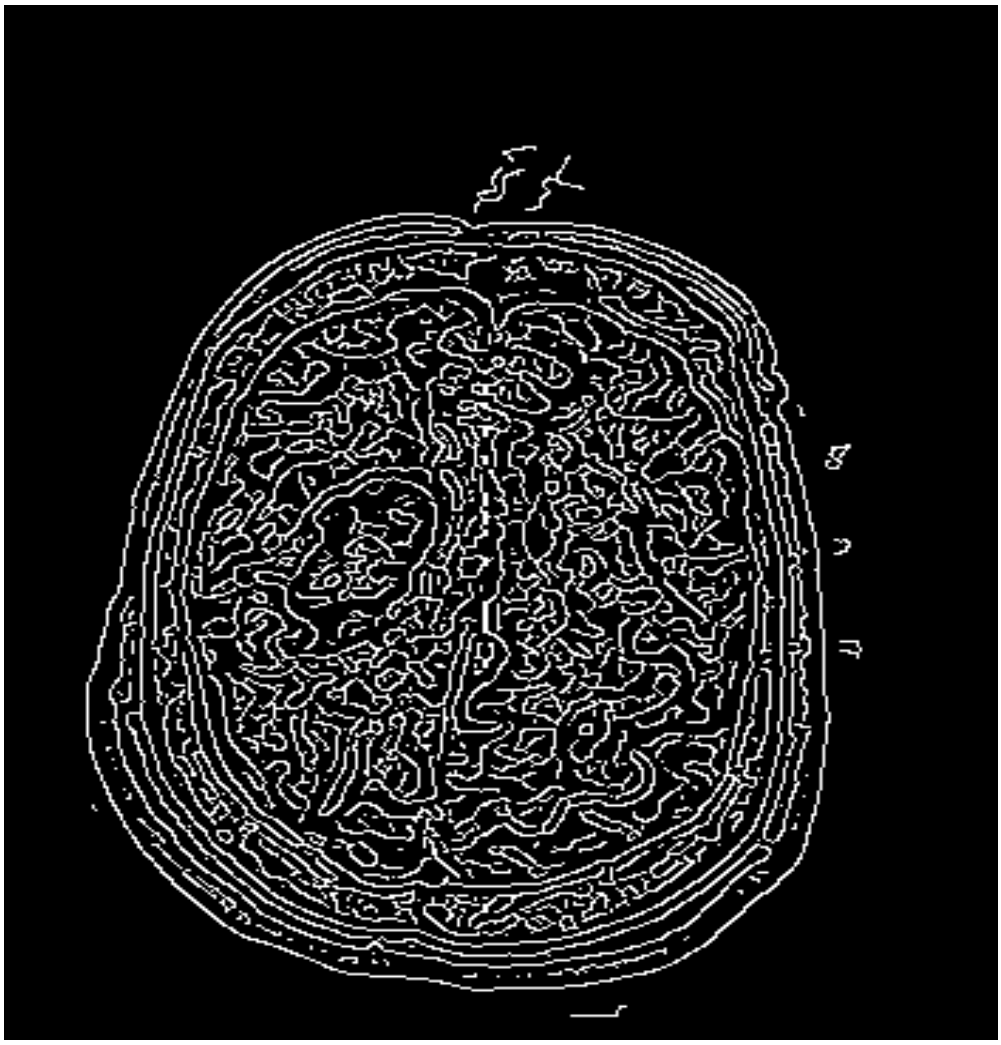


Fig V.5 new image (binary one) with Canny Edge

V.3.4 Calculate the GVF of the edge image

Fig V.6 present the Flow Chart of Calculate GVF with more explained in the next titles.

V.3.4.1 Calculate the Gradient of the Edge

This step used to detect all features of image (all Contours) its algorithm showing in Table I.1.

The data result of canny edge detector on form digital we also must transform it to binary for success the next calculate (the gradient of the edge).

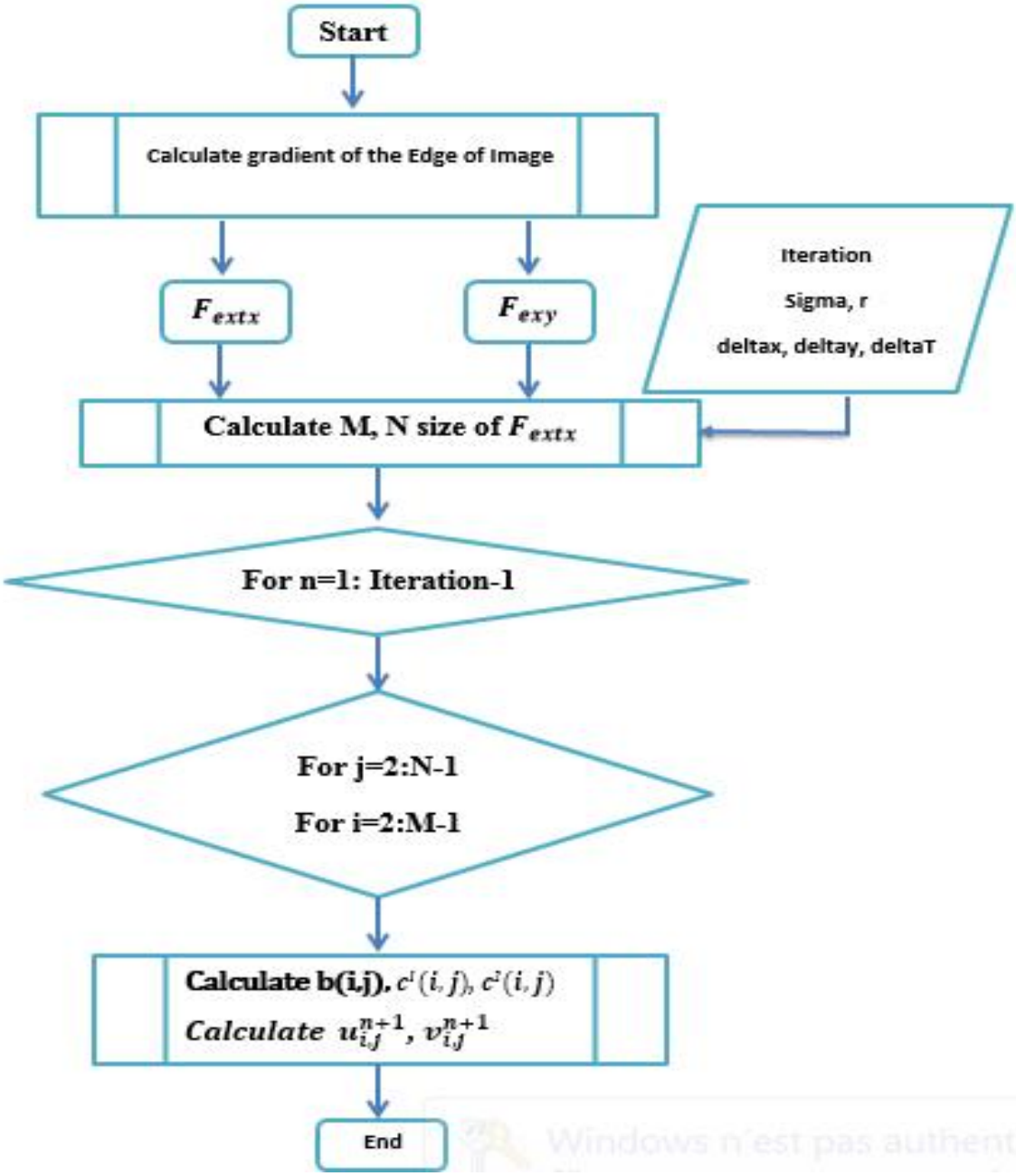
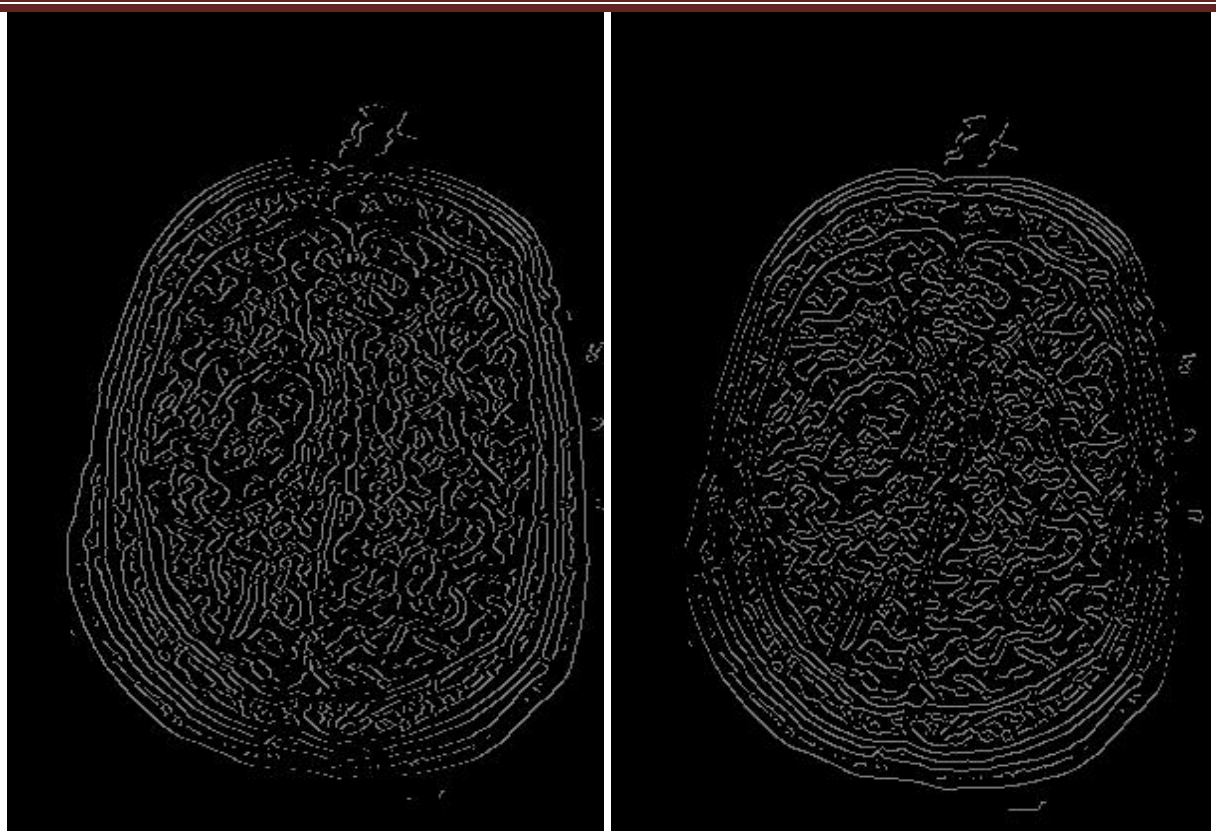


Fig V.6: The Flow Chart to Calculate the GVF

From GVF organigram we start by calculate the gradient from the edge of coordinate x fextx & the gradient from the edge of coordinate y (fexty). See Fig(V.7)



(a) (b)

Fig V.7: (a) The gradient of the Edge oriental, (b) The gradient of the Edge vertical

V.3.4.2 Calculate the GVF of the edge image

For the result in Fig V.8 we choose : deltax=deltay=deltat=1, r=gamma= 0.2, iterative=50

The different results of different iterations showing approx. After 50 iterations ,the GVF stabilized the results of orientation u(x,y) ,vertical vend v(x,y),and according wih u presented in (Fig V.8)(a,b and c respectively).

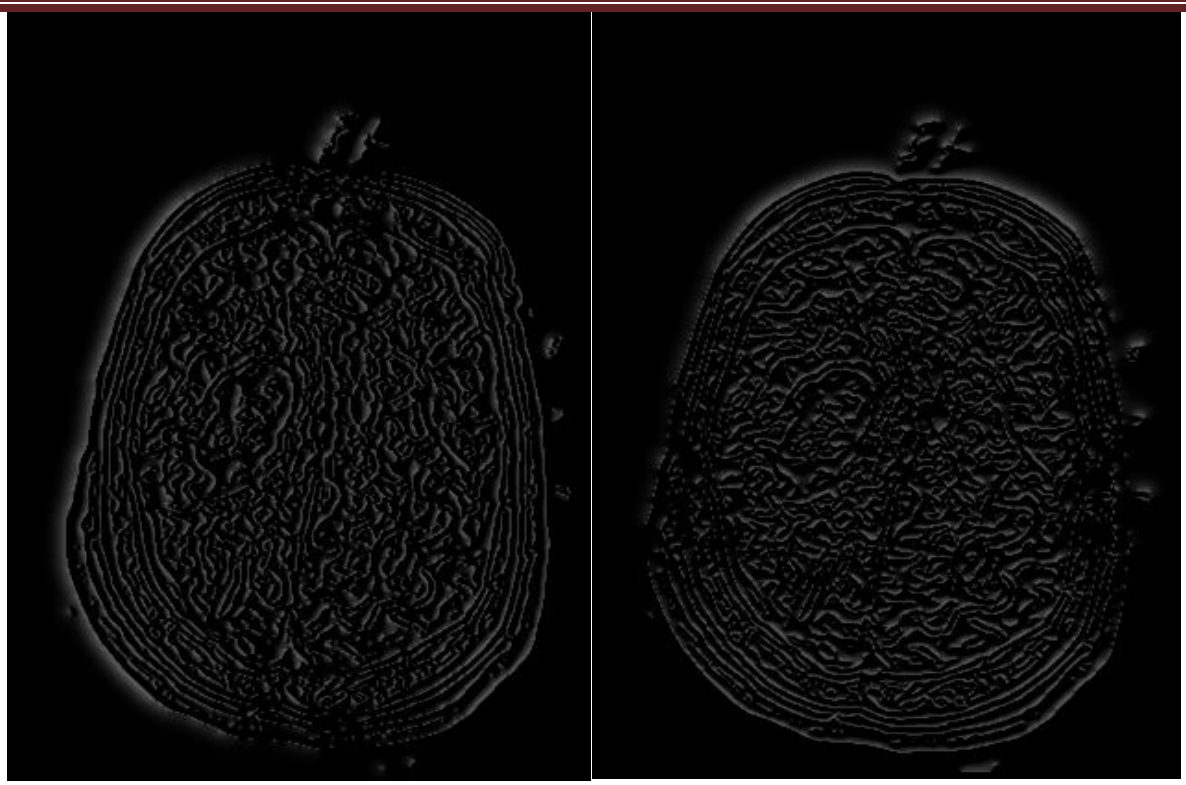
where

$$u_{i,j}^{n+1} = (1 - b_{i,j}\Delta t)u_{i,j}^{n} + r(u_{i+1,j}^{n} + u_{i,j+1}^{n} + u_{i-1,j}^{n} + u_{i,j-1}^{n} - 4u_{i,j}^{n}) + c_{i,j}^{1}\Delta t$$

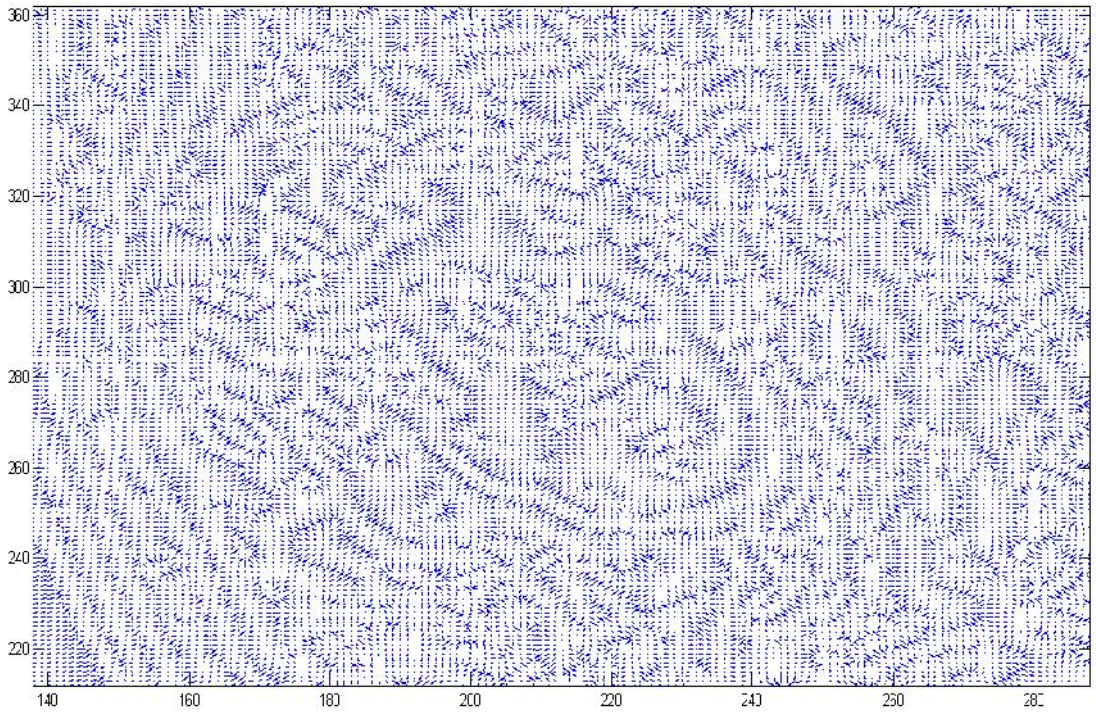
and

$$v_{i,j}^{n+1} = (1 - b_{i,j}\Delta t)v_{i,j}^{n} + r(v_{i+1,j}^{n} + v_{i,j+1}^{n} + v_{i-1,j}^{n} + v_{i,j-1}^{n} - 4v_{i,j}^{n}) + c_{i,j}^{2}\Delta t$$

they explained in the chapter III .



(a) (b)



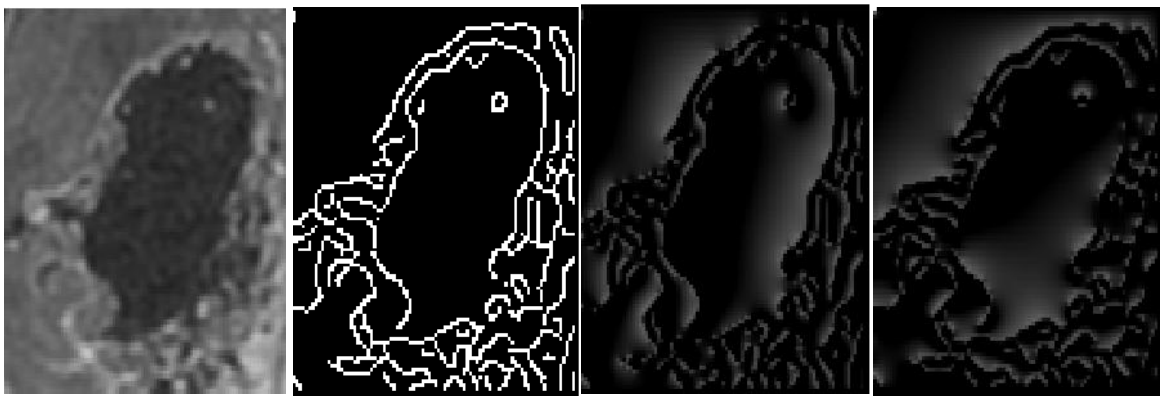
(c)

Fig V.8: (a) The result of GVF oriental (u(x,y)), (b) The result of GVF vertical (v(x,y)) and

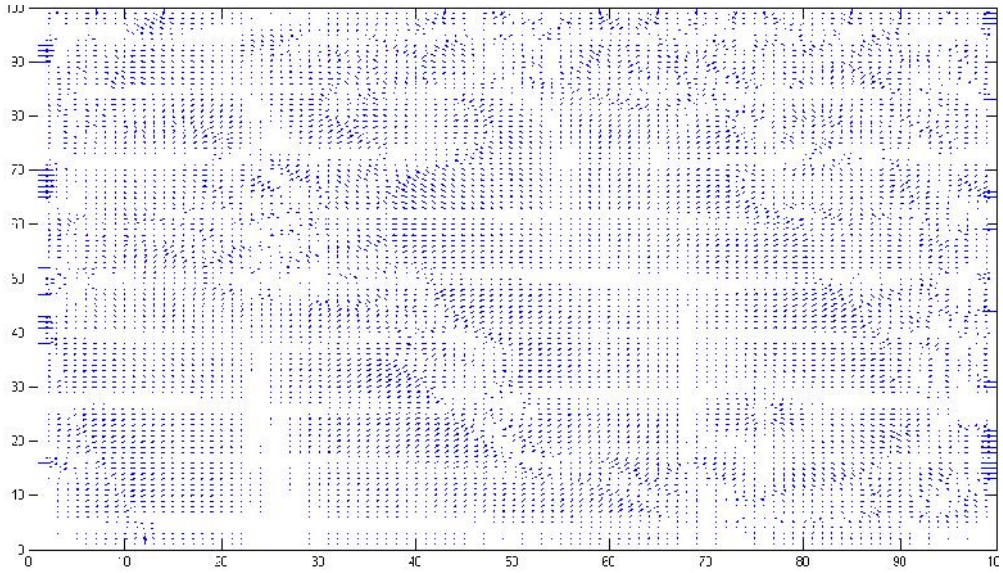
(b) The presentation of GVF (v according of u)

V.3.4.3 Results of the Deformable image

This is an example to applied the GVF into deformation image See fig(V.9.a), then detect it with canny edge(fig(V.9.b), then give the GVF orientation .fig(V.9.c), the GVF vertical and the according GVF with (u,v)fig(V.9.d)



(a) (b) (c) (d)



(e)

Fig V.9: (a) deformable MRI image, (b) Edge Canny to detect the deformation, (c) GVF oriental u(x,y) of the deformation (c)GVF vertical v(x,y) of the deformation and (e) the presentation of GVF (v according to u).

V.3.5 Selecting of the Location of the Control Points

There are several ways to choose the initial snakes (or seed points):

-by hand

- by “balloons”

- by the locus of the zero-crossing of the Laplacian of the smoothed images or source points.

The source points can be automatically generated from the gradient vector field.

Definition: A point is called source if none of its neighbors points to it. In other words, a point A is a source if and only if, for any neighbor B of A:

$$\vec{g} \cdot \vec{AB} \leq 0$$

where $\vec{g}$ is the diffused gradient vector at B and $\vec{AB}$ is the vector from B to A

In our case we use manual selection (by hand) expected in (Fig V.10) which present the Flow Chart of select the control points

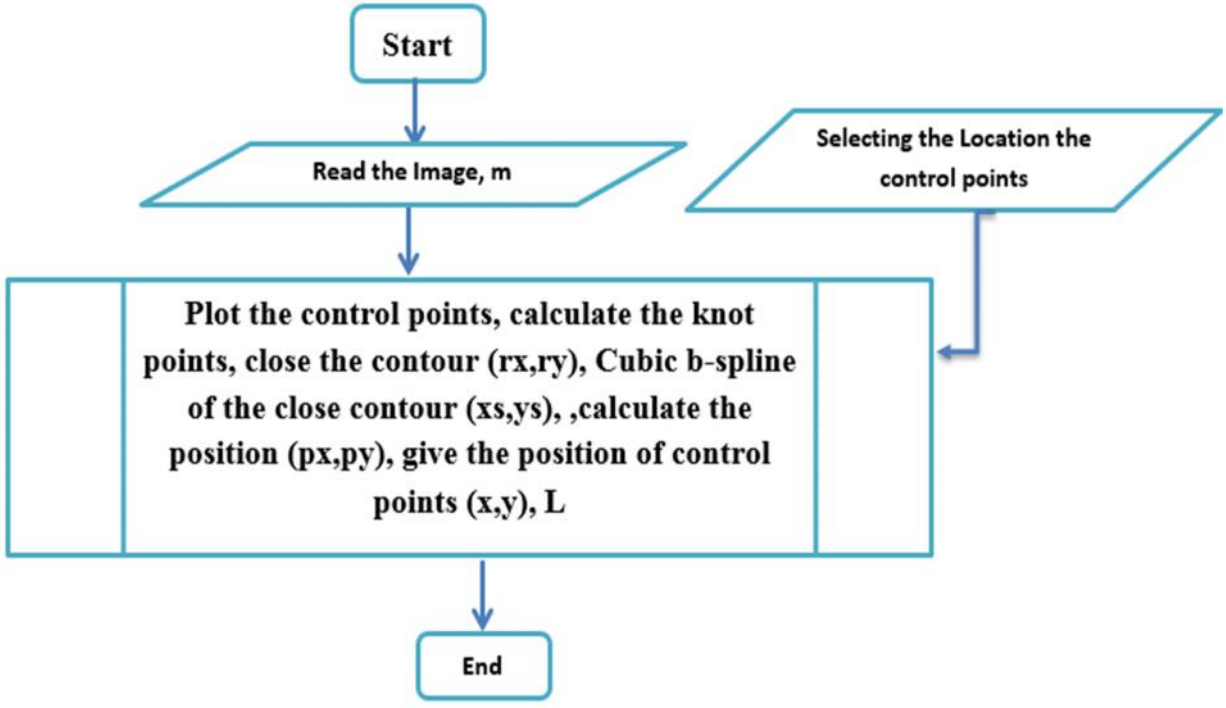


Fig V.10: Flow Chart of selecting the control points

V.3.5.1 Selection the control points (by hand)

Fig V.10 present the select of four control points by hand and plot them, and in the same time take their position (x,y) to study their energy.

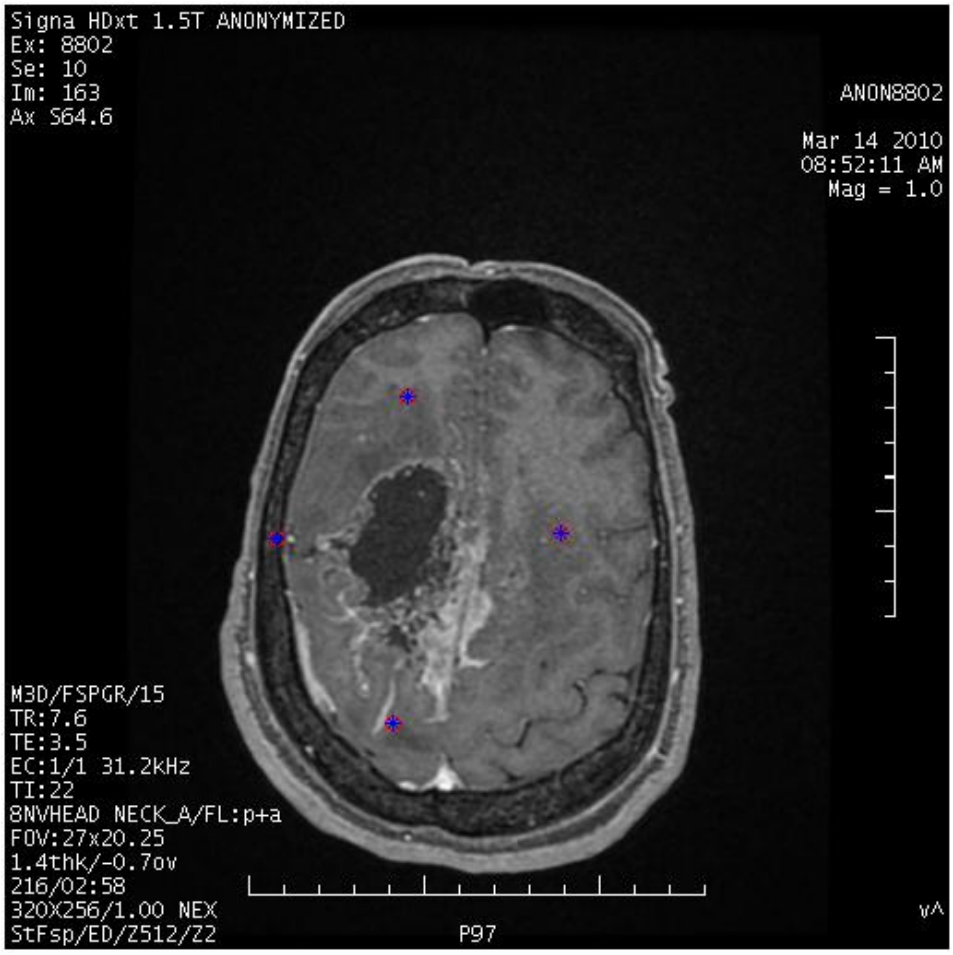


Fig V.11: Select four control points and plot them on the image (red color)

We can begin with four control points then more .if the control points initial increased, the result will be better (detection accurate).

V.3.5.2 Calculate the knot points

Fig V.12 present plotting Control points selecting and their Knot points which Calculate with using s=0 in (Eq IV.5).

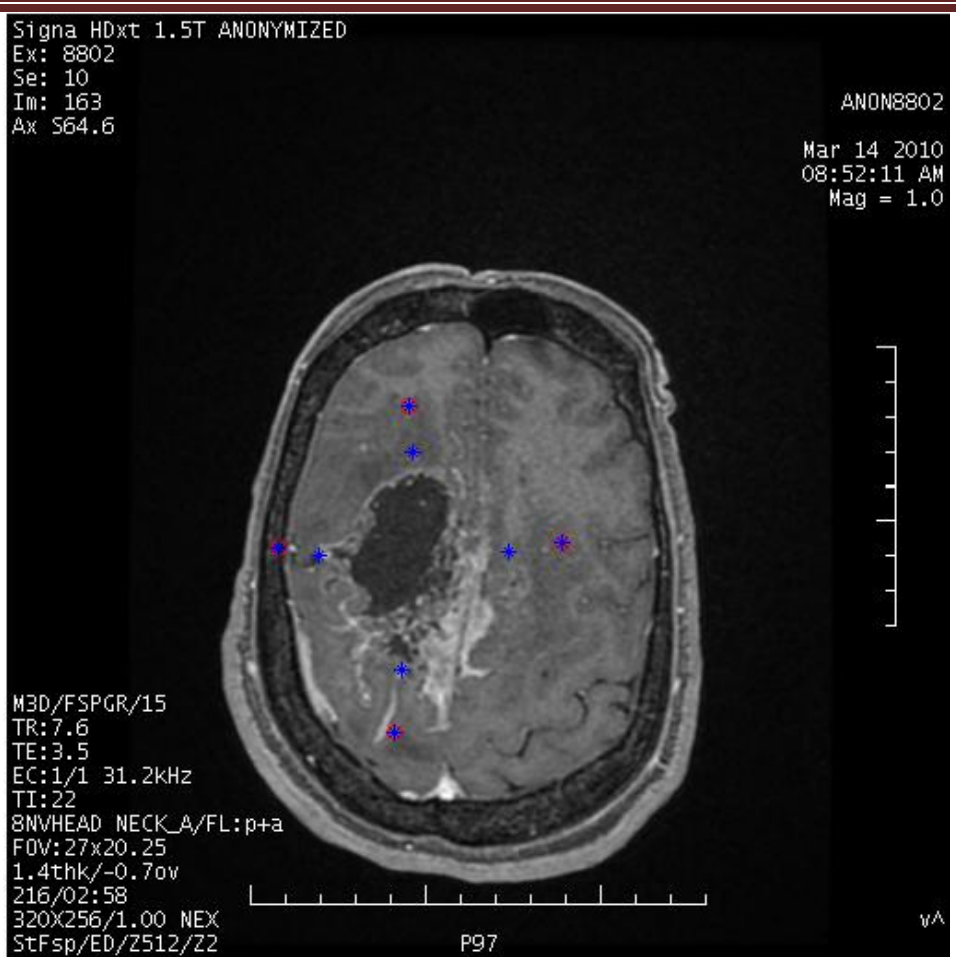


Fig V.12: Plot the Control points selecting and their Knot points

Knot points are related to control points by substituting $s = 0$ in Eq (IV.1) as shown in (EqIV.5) and (Eq IV.6) in which P and Q are L×2 matrices. Containing knot points and control points, respectively because $Q = \begin{pmatrix} Q_x \\ Q_y \end{pmatrix}$ and $P = \begin{pmatrix} P_x \\ P_y \end{pmatrix}$.

We replaced the matric A with s=0 in Eq (IV.1) to prove this matrix and to find the same results it showing in Fig V.13

The Fig present the relation between the control points Q, the segments g and the Knot points P. we have two important points of the Knot points:

1-stady of control points for the neighbouring energy of the control point to avoid the thin contours.

2-We show the knot points put flexible the insert points (control points) to gather the solution. The problem of insert points like we note it in the point Q_{n-2} in the Fig V.14 and them commensurate with the different shapes and the concavities. This case is clearer in the insert

more than 4 control points, and like the case of the insertion strategy that we will explained next.

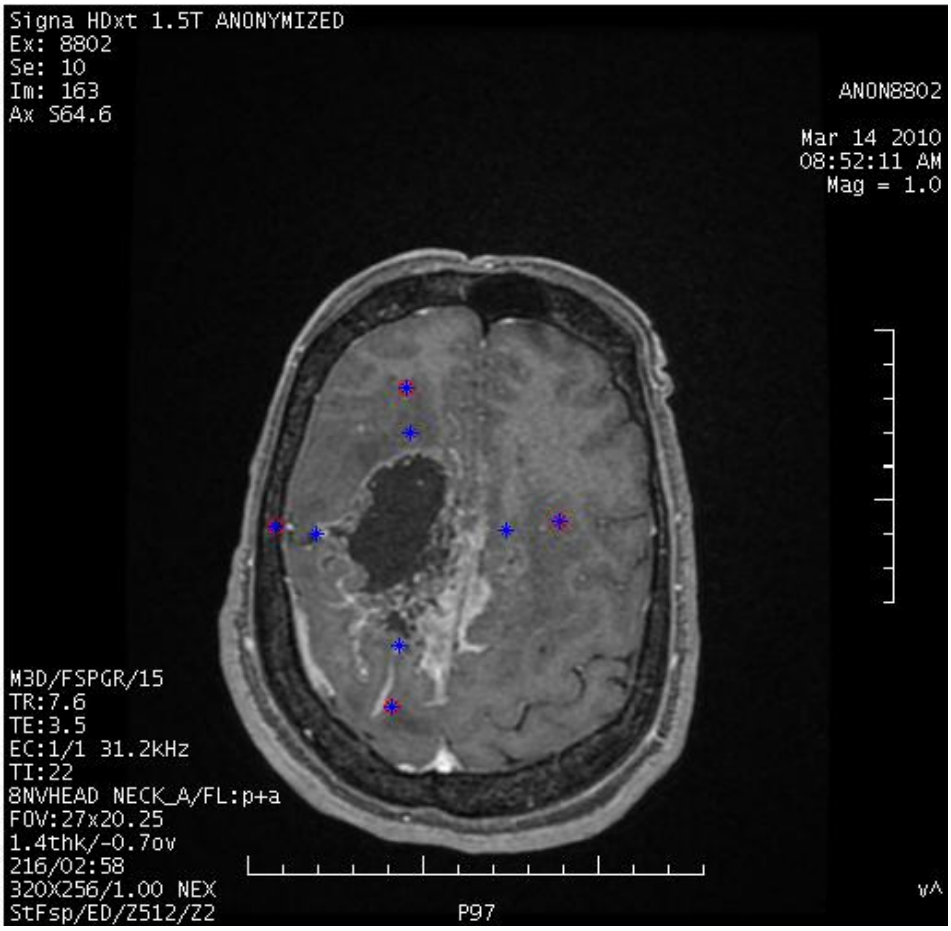


Fig V.13: Plotting the Knot points with using the matric A

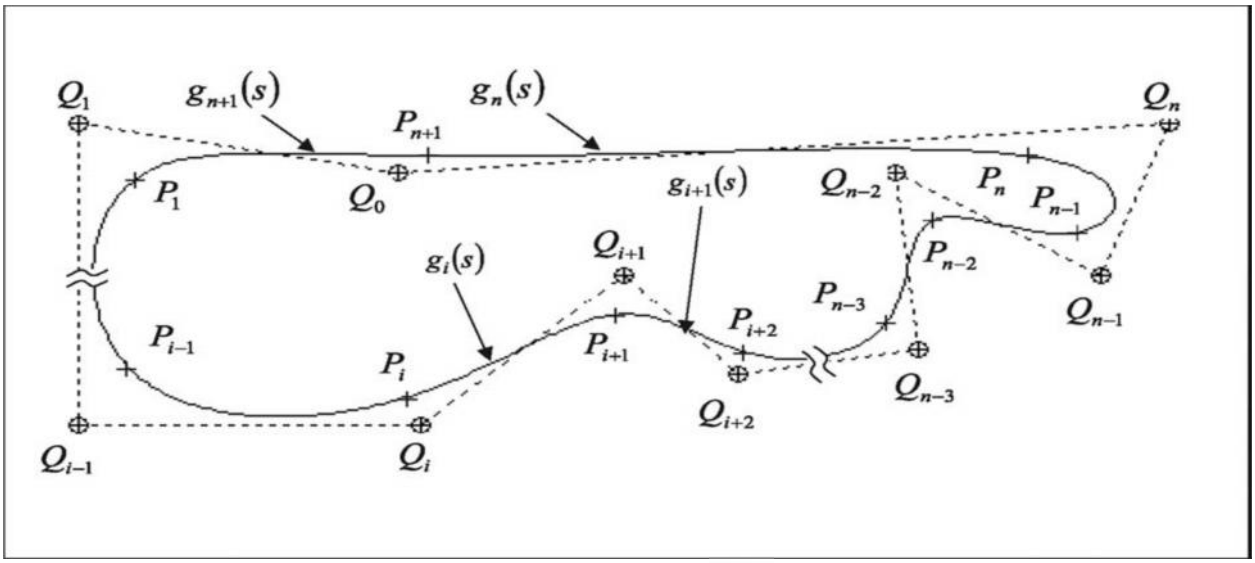


Fig V.14: Schema presentation the relationship between the control points, their Knot points and the segment curves g_i [54]

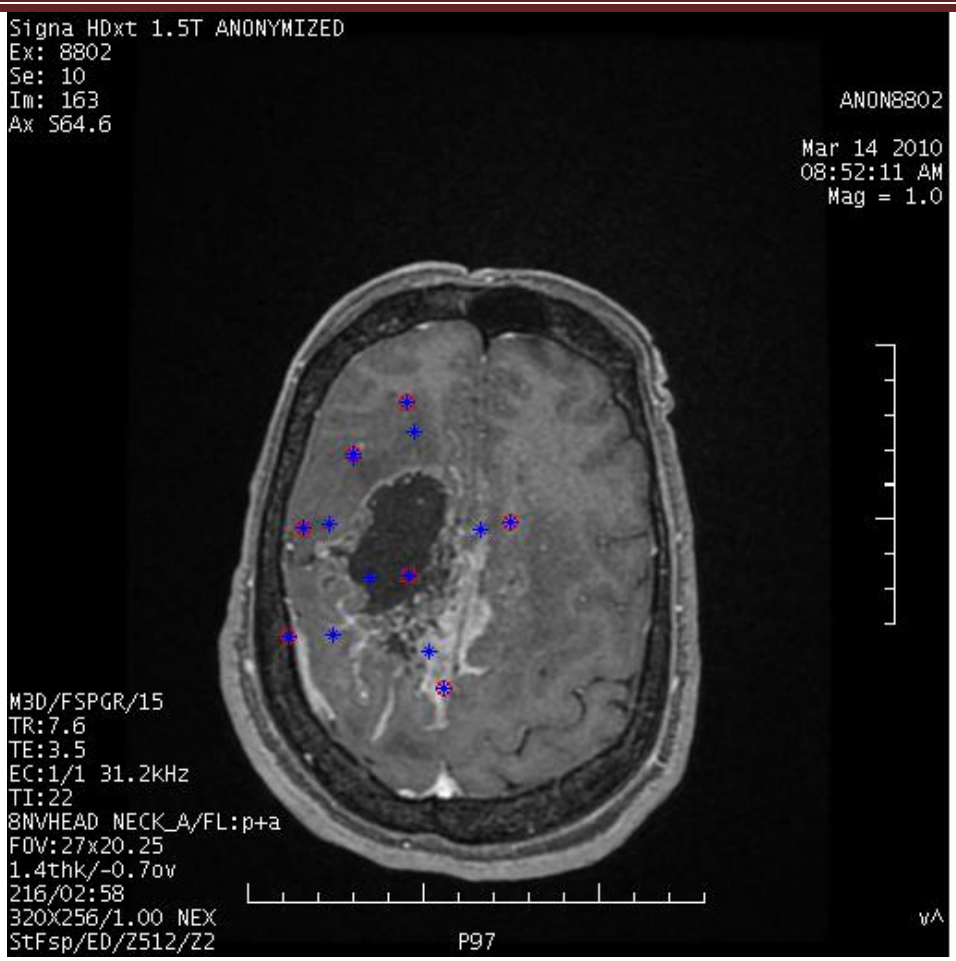


Fig V.15: the plotting of control points and their Knot points in

case to using more than 4 control points

-Make the contour close (rx,ry) to returned to the first point.

-The variable m: must be inserted to determine the number of discretization between each two knot points. An increase m is the best detection of contour (m present the number of points in each segment curve).

-Plot the initial contour with using B-Spline (xs,ys).

The three previous steps present their result in (Fig V.16).

-Calculate the px and py which are the position of discretization (xs,ys) (exactly it is calculate the number of pixel because the spline cannot give it).

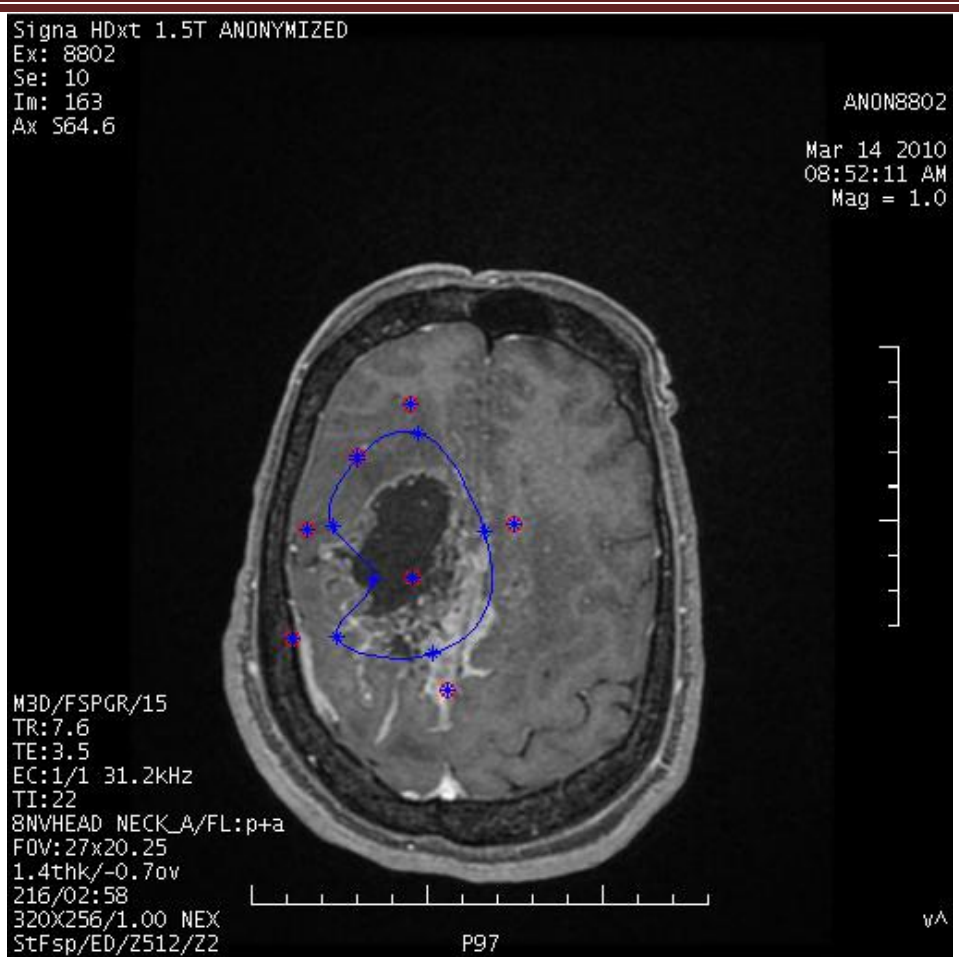


Fig V.16: The Spline of the initial contour (closed)

V.3.6 Calculate the $\Delta Q(t)$ from initial contour

Fig V.17 shown the flow chat of calculate the $\Delta(\mathbf{t})$ as the following

-Calculate Fxt and Fyt: GVF value of all the discretization of the segments curve (sum of (px,py)) Fxt, are the GVF oriental (u), and Fyt are the GVF vertical (v)).

-Calculate $\Delta \mathbf{Px}$, $\Delta \mathbf{Py}$ **and** $\Delta \mathbf{Q}$: $\Delta \mathbf{Px}$ is the sum of calculate Fxt/Sigmma and $\Delta \mathbf{Py}$ is the sum of calculate Fyt/Sigmma.

- ΔQx, ΔQy calculed by using (EqIV.5) and (Eq IV.6) .and the relation

$$\Delta \mathbf{Q} = \sqrt{\Delta \mathbf{Qx^2} + \Delta \mathbf{Qy^2}} \, . \qquad \text{(Eq V.1)}$$

After that we return to the main flow chart and compared $\|\Delta \mathbf{Q}\|$ with the **threwold**[1]

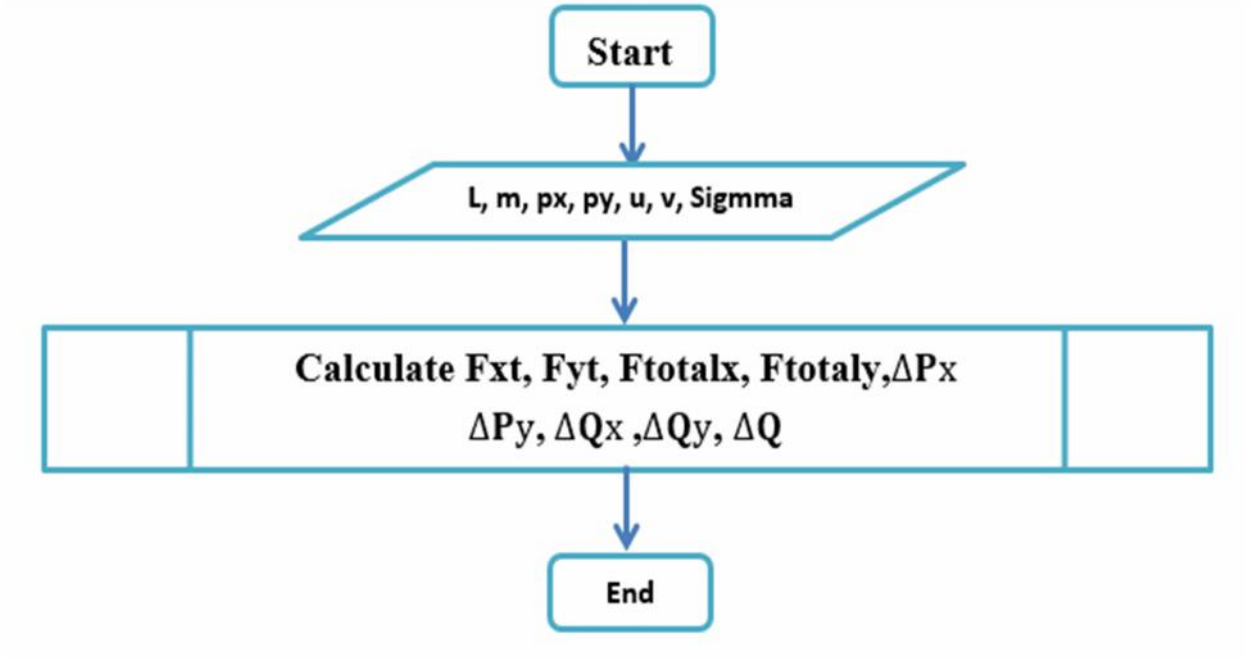


Fig V.17: The Flow Chart of Calculate ΔQ(t)

Than we applied the condition while if ||ΔQ ||be stronger than threwold[1] [1]

x=x+ΔQx and y=y+ΔQy and next calculate **ΔQ** of new contour**.**

We use the relation (Eq IV. 8) but in practical application, the energy function would never reach zero, but a minimal value. Once the energy function of B-Snake is minimized, then no movement drove for control points and hence no change in the shape of the B-Snake.

Where i = 1,2, ... , L and $\Delta Q(t)$ denotes the amount of movement of Q at time t. As stated before, each curve segment is defined by four control points and each control point affects four curve segments. Moving a control point in a given direction will affect the four curve segments to move in the same direction. For each control point, change in position is determined by its four nearest adjacent curve segments. In other words, external forces of each four adjacent segments are transmitted to their central control point. Obviously, external forces on any two adjacent curve segments have the most effect on the displacement of their common knot point at time t that is the displacement of Pi at time t is mainly influenced by Fxt and Fyt. The idea is to determine the displacement of each knot point Pi by transmitting external forces associated with its adjacent curve segments to Pi and to neglect the effect of other external forces. In order to avoid crisp contribution of samples, the proposed method passes on weighted external forces associated with adjacent curve segments to their common knot point to determine the displacement of that knot. This procedure is presented below:

$$\Delta Q(t) = A^{-1}\Delta P(t) \qquad \text{(Eq V.2)}$$

when end this condition (checks) go to step 7.

V.3.7 The Control Points Insertion Strategy

We applied the control points insertion strategy to adaptively approach a complex contour without prior fixing of the number of control points, a structure adaptive knot point insertion strategy is developed. The location of new knot point is determined by checking the external force on the B-Snake when B-Snake is converged in the deforming procedure. The new knot point is inserted into the location where it has maximal external force on the B-Spline curve. In the GVF field, the magnitude of the force reflects the distance to the nearest significant edge. The stronger external force means the further distance to the nearest object boundary. In this case, the knot point should be inserted to increase the flexibility of the B-Spline curve to adapt to the object shape.

The control points insertion strategy explicit in the (Fig V.18) which present the Flow Chart.

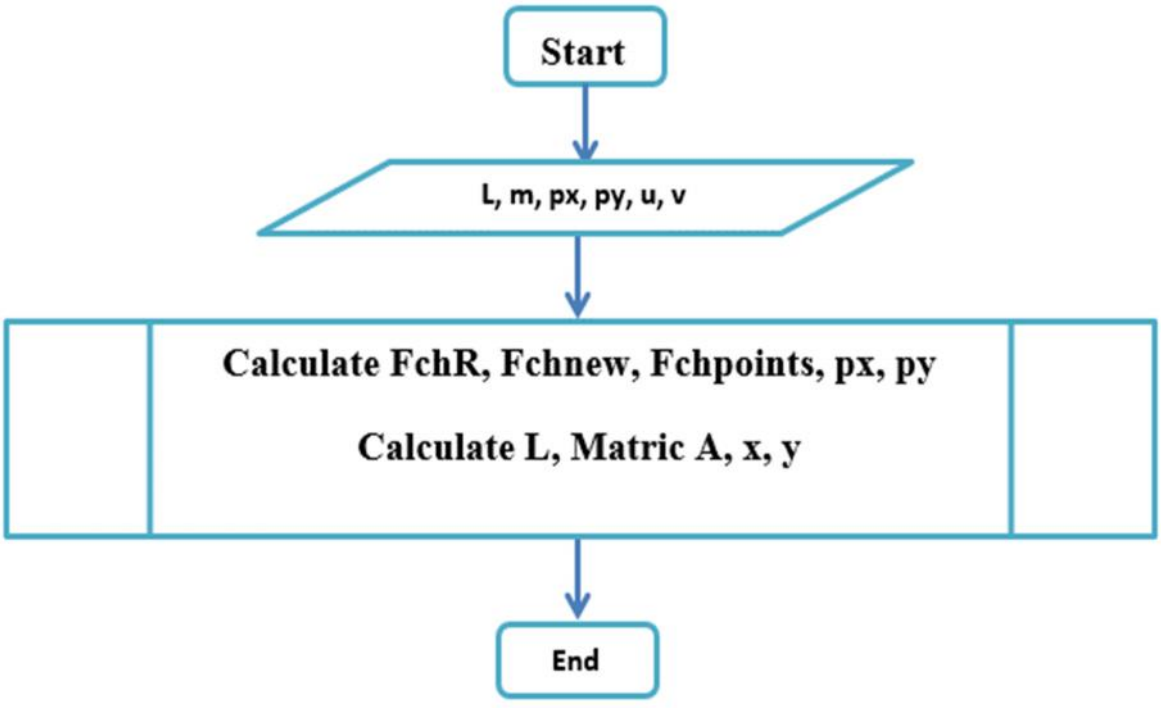


Fig V.18: The Flow Chart of Insertion Control point strategy.

-Calculate FchR: is the Tendon of each point in the curves (1 to L) (each curve segmentation to m points).

-Calculate Fchnew: is the max of External force (GVF) (max of FchR).

-Calculate Fchpoints: are the order of the new knot points inserted in the original knot points with respect their positions.

-Calculate the length of all knot points (L) and the matric A; with respect the new number of L.

-Calculate the control points with using the relation (Eq.IV.5) the control points (x, y)

Now we returned to the Main flow chart (Fig V.1)

First compared the Fchnew with the $\mathbf{threwold^2}$ then applied the condition while jus the Fchnew be small than $\mathbf{threwold^2}$

$$\textbf{if } \mathbf{Fchnew} \geq \mathbf{threwold^2}$$

we return to the step 6 calculate $\Delta\mathbf{Q(t)}$ of new control points while if $\|\Delta Q\|$ be stronger than $\mathbf{threwold^1}$ will calculate the $\mathbf{x=x}+\Delta Qx$ and $y=y+\Delta Qy$ in every time.

else that is meaning the end of programme and spline the last contour which controlled with the last position (x,y) (control points after we applied the movement of control point depending on the External energy (External forces: GVF) and applied the insertion control points strategy) but in the language of programing it is meaning the two convtions are investigator ($\mathbf{max} \| \mathbf{E_{ext}} \| < \boldsymbol{threshold^2}$ and $\|\Delta Q\| < \mathrm{threwold^1}$). [1]

V.3.8 Our finale Results

The final results will be presented in the next Figures.

Fig V.19: present the initial control point and the initial contour with 4 control points of the first shape.

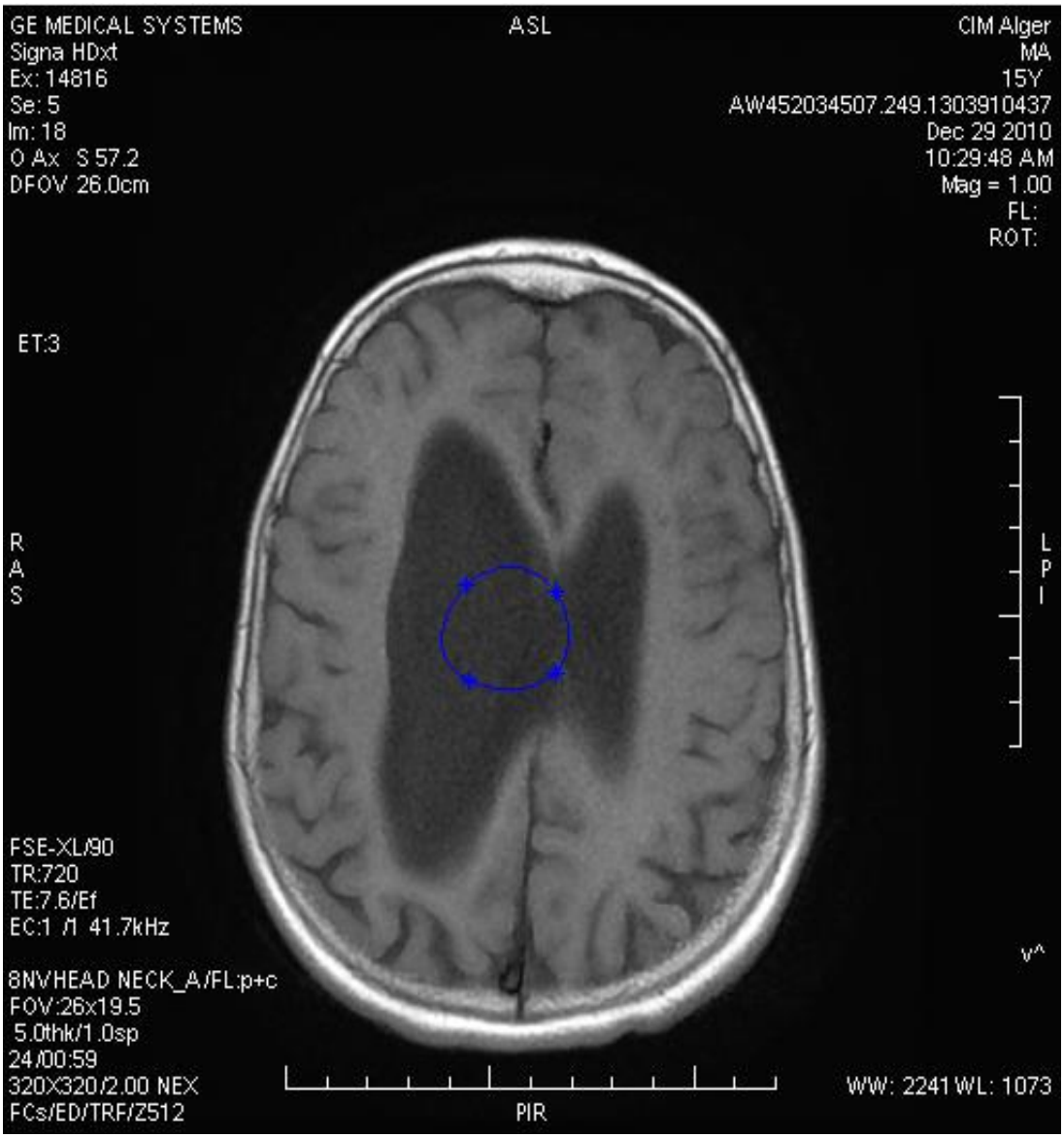


Fig V.19: Present the initial control points and the initial contour

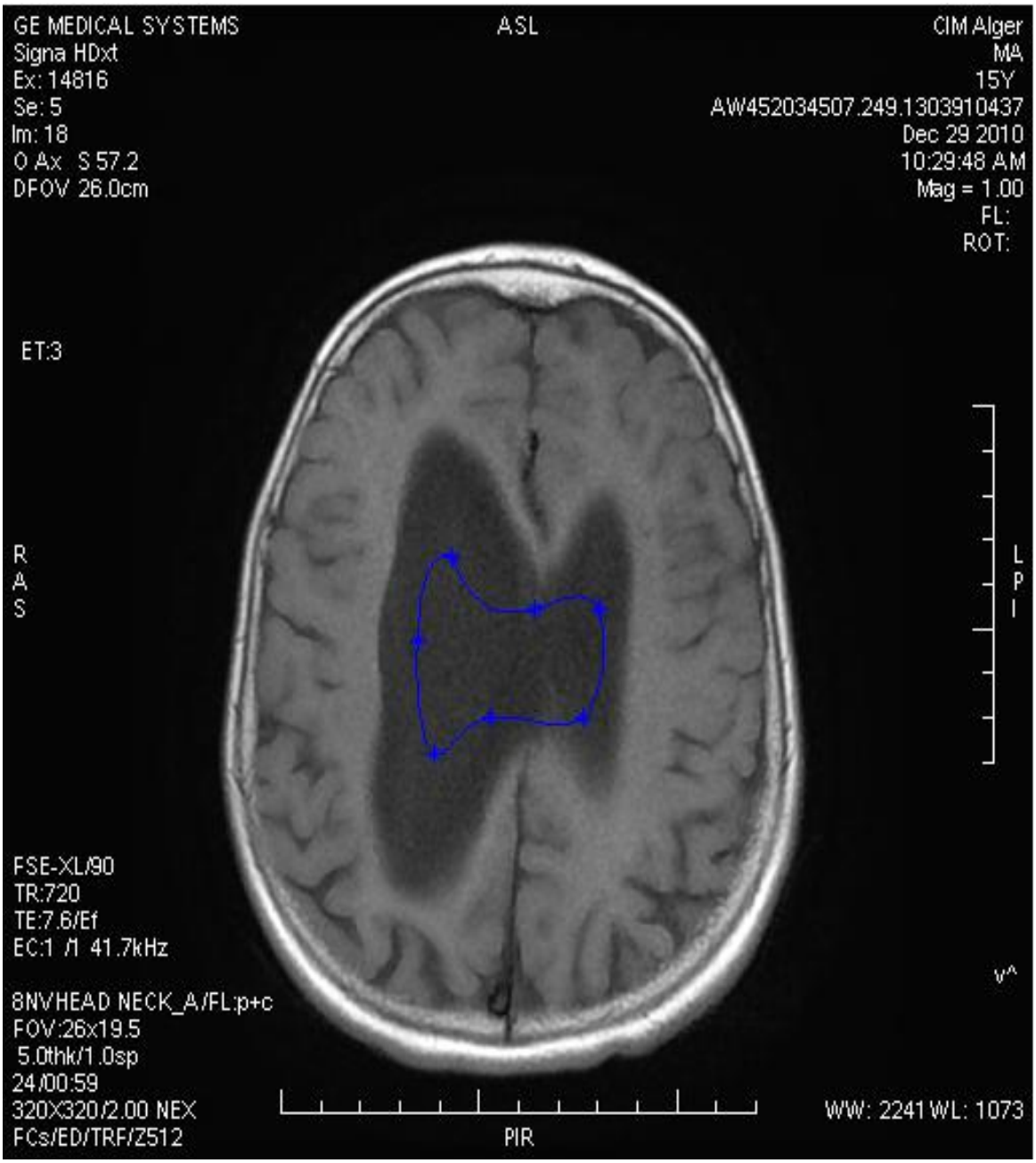


Fig V.20: Present the result after 10 Iterative

Fig V.20: Present the result after 10 Iterative of shape 1, where the control points move to the contour of deformable and the number of control points increases from 4 to 7 after10 iterative.

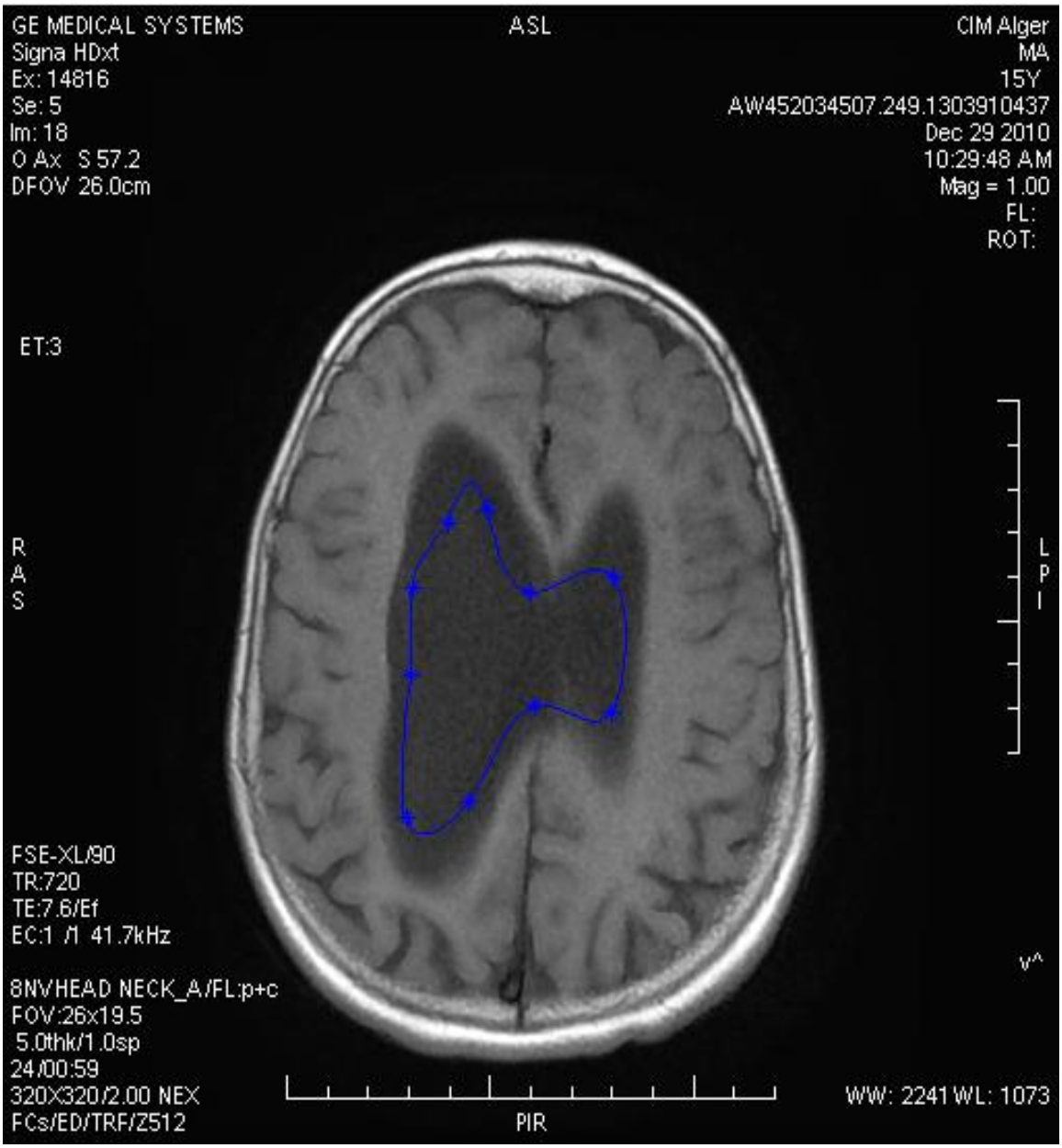


Fig V.21: Present the result after 20 Iterative

For the Fig V.21: Present the result after 20 Iterative of shape 1, where the control points move more to the deformable contour and the number of control points increases from 4 to 10 after 20 iterative.

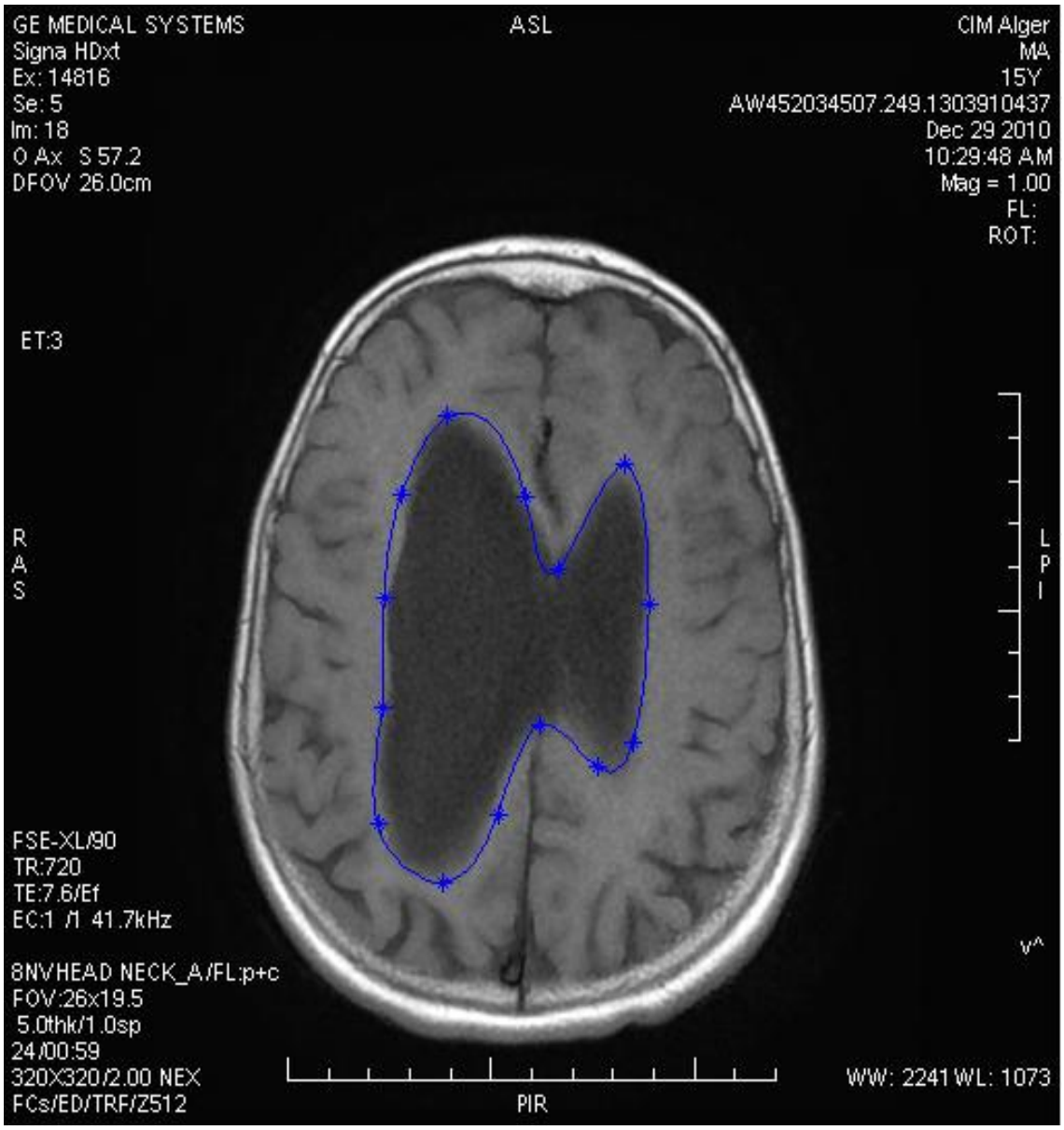


Fig V.22: Present the finale result after 27 Iterative

Fig V.22: Present the finale result after 27 Iterative, where the control points move more to the deformable contour and the number of control points increases from 4 to 14 after 27 iterative.

Fig V.19: present the initial control point and the initial contour with 4 control points of the second shape.

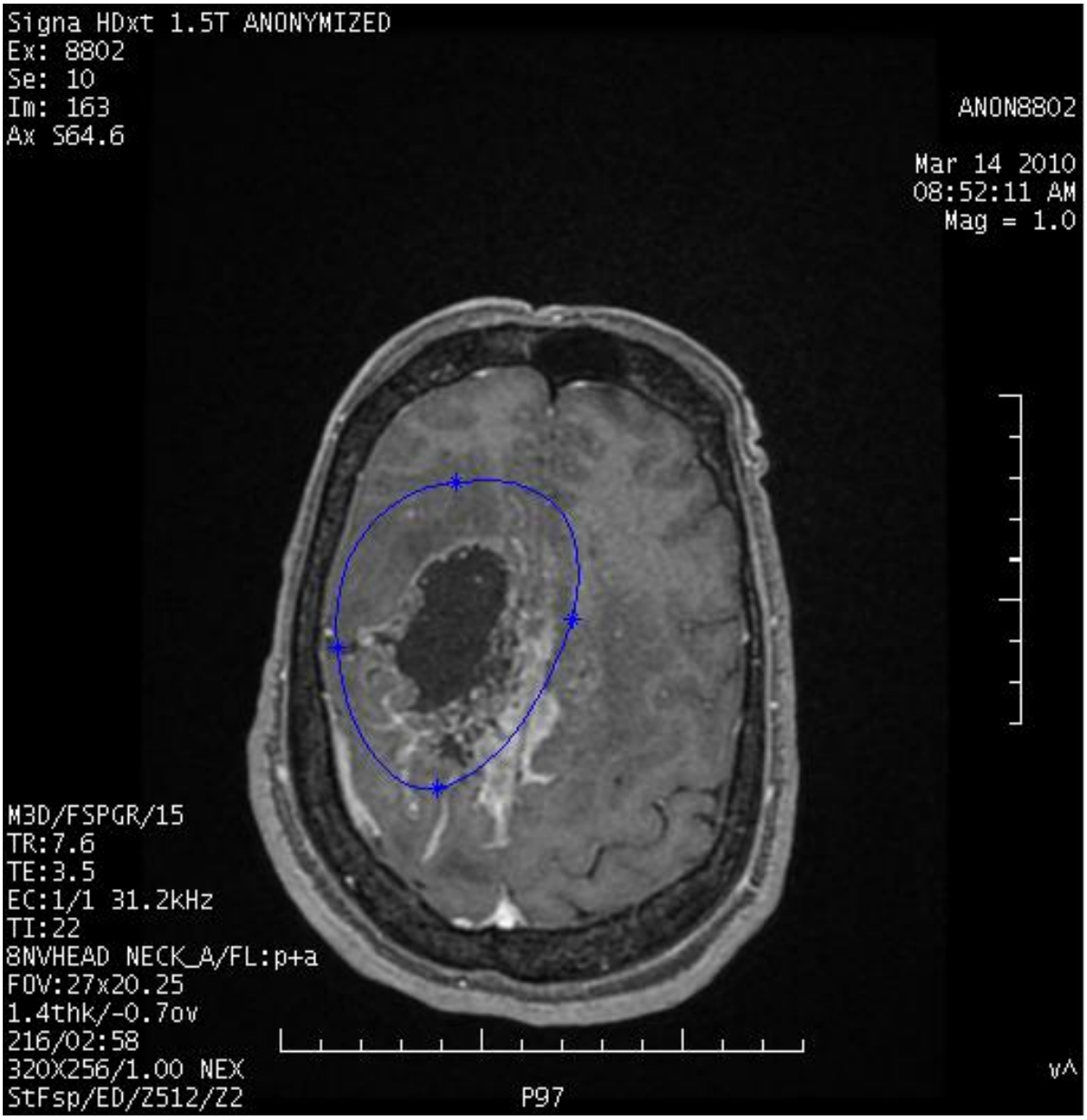


Fig V.23: Present the initial control points and the initial contour

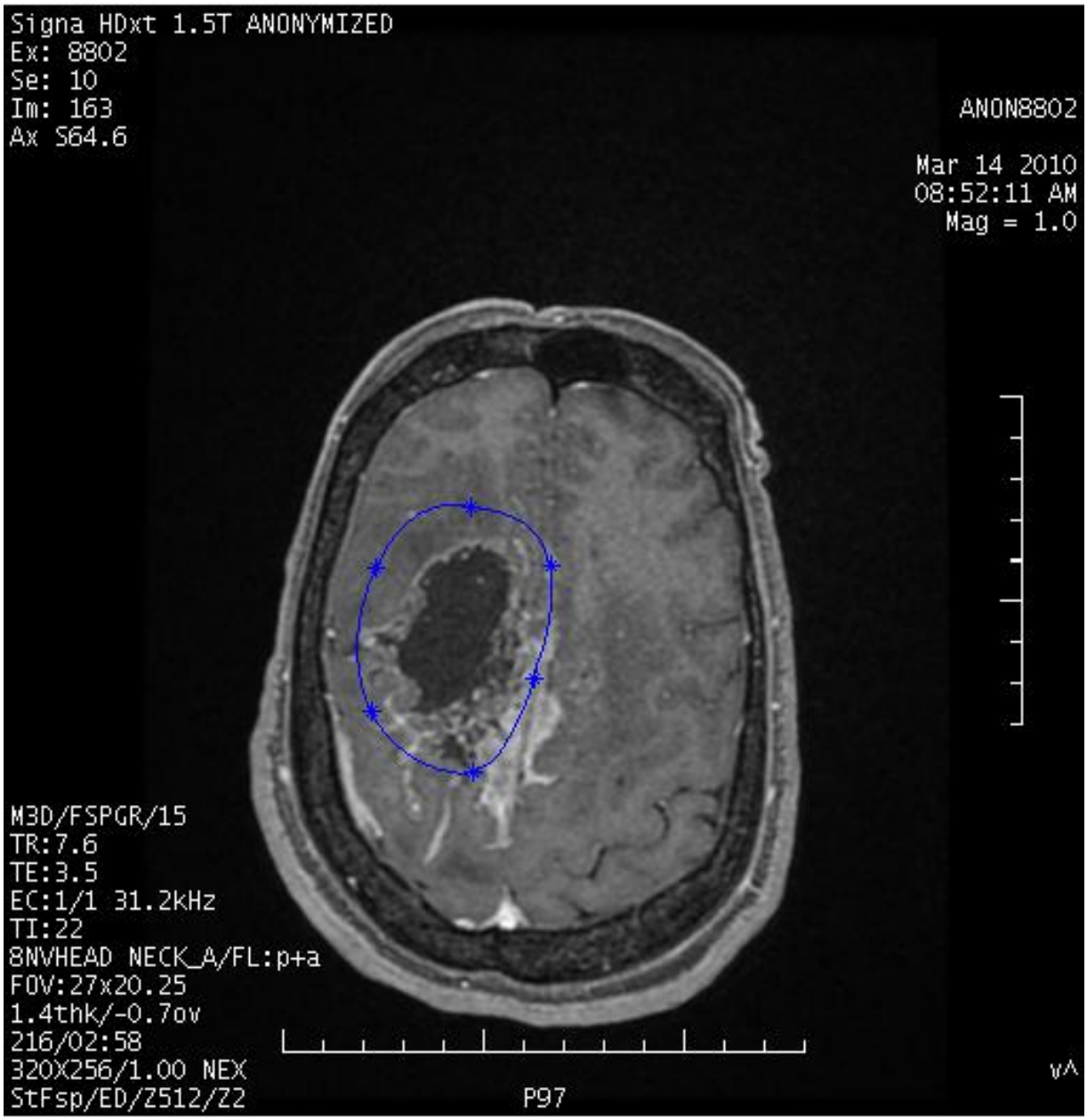


Fig V.24: Present the result after 10 Iterative

Fig V.24: Present the result after 10 Iterative of shape 2, where the control points move to the deformable contour and the number of control points increases from 4 to 6 after10 iterative.

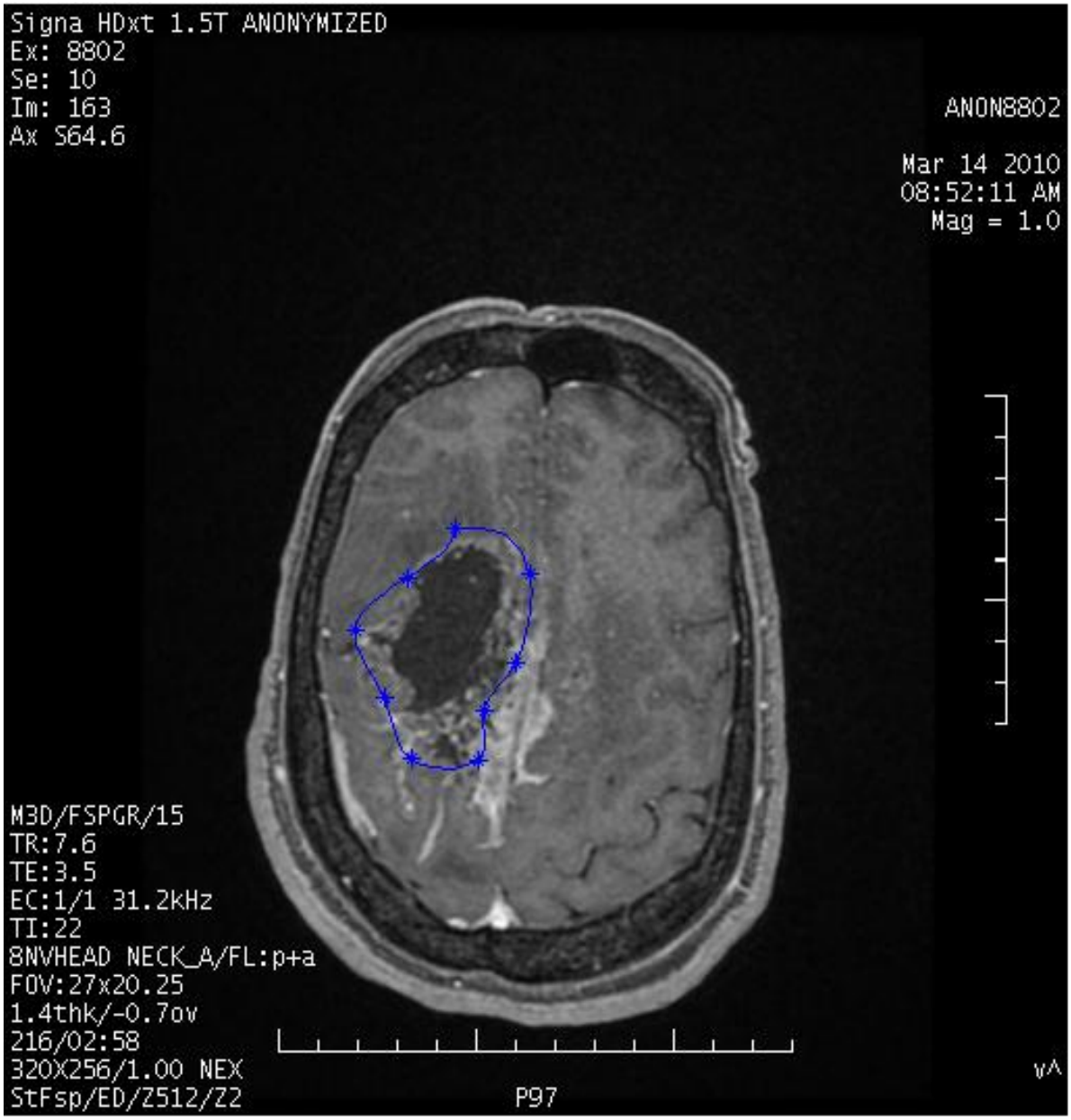


Fig V.25: Present the result after 20 Iterative

Fig V.25: Present the result after 20 Iterative of shape 2, where we can see the control points move more to the contour of deformable and the number of control points increases from 4 to 9 after 20 iterative.

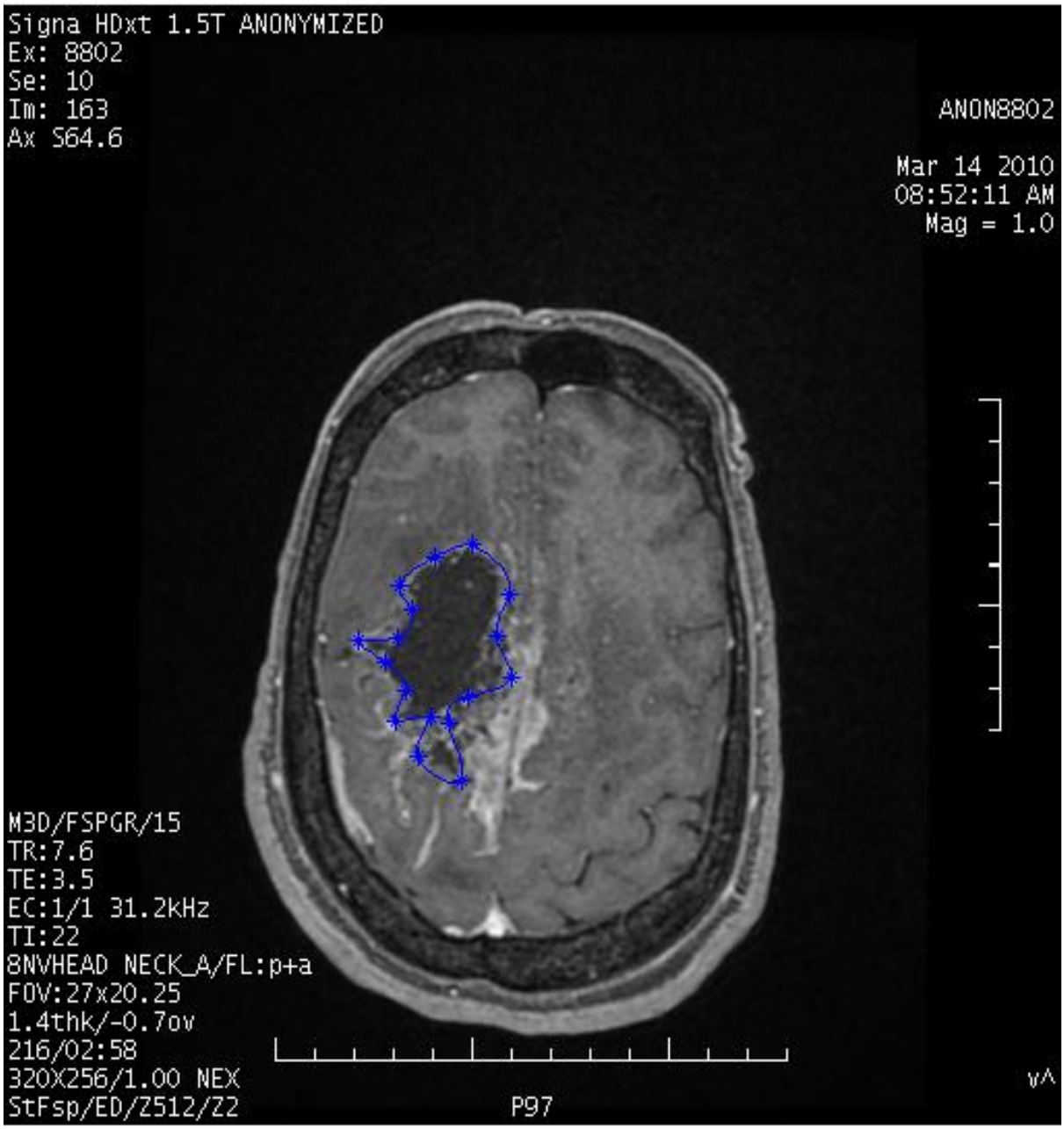


Fig V.26: Present the finale result after 33 Iterative

Fig V.26: Present the finale result after 33 Iterative, where the control points move more to the contour of deformable and the number of control points increases from 4 to 17 after 33 iterative.

V.4 Conclusion

The set of $\mathbf{threwold^1}$ and $\mathbf{threwold^2}$ are very important to the finally results, because they are like a guide of contour best set contour as final result.

In our results we chosen them0.2 for the first set and 0.5 for the second set of pixels respectively.

We compared our work in the chapter 5 with the chapter 4 which present a method of B-Snake (YUE WANG [54]) with same modifications. The difference is the part VI.3.4 , step 4 of the algorithm of Yue Wang[54], exactly in the point of calculate $\Delta Q(t)$ where the article used MMSE which is a multiplication operation between matrices ,but our work , used the sum operation between matrices. And here the power of Yue Wang article in front of our idea because, machine response is faster with multiplication.

General Conclusion

Active contours or (Snakes) is a major challenge in medical image processing, where the first work begin with the theory about (1950) and each time develops and improves because it is a big area for research.

This new method B-Snake is new relatively from (2006) and each time develops. Now and all information about this works is secret; they only give some of the reality of the theory and lets for you the part the idea and applications.

The purpose of the develops of this work is to access to the fast with high accuracy in image (2D) ;it must develop this work to (3D) and for video(24 image in the second) where, with video of patient we need achieve to the nature of Malignancies (active) quickly to detect the lesion then to eradication it without any error or spent long time in test.

And the strong purpose to of B-snake in IRM is transform this work to a machine like the work In ref [55] which present the using of method of B-Snake in the lane detection and transform them to machine, but in the trouble in our case is the sensitive of the field medical.

[1]A. A. A mini, T. E. Weymouth, and R. C. Ja in, "U sing dynamic gramming for solving variational problems in vision," IEEE Transtern Anal. Mach. Intell., vol. 12, n°.9, pp. 855-867, Sep. 1990.

[2] D. J. Kang, A fast and stable snake algorithrn for medical images", Pattern. Recognition, vol. 20, n°5, pp. 507-512, 1999.

[3] S. Menet, P. Saint- larc, and G. Medioni, Bvsnakes: Implementation and application re stereo," in Proc, Image Understanding Workshop, Sept. 1990, pp. 720-726. 1990.

[4] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int. J. Computer Vision*, vol. 1, no. 4, pp. 321-331, 1987.

[5] E. Pers on and K.S. Fu, "Shape discrimination using fourier descriptors," IEEE Transactions on Systems Man Cybernet, Vol. 7, pp. 170-179, 1977.

[6] W.S.I. Ali and F.S. Cohen, "Registering coronalhistological2-D sections of a rat brain with coronal sections of a 3-D brain atlas using geometric curve invariants and B-Spline representation," IEEE Transactions on Medical Imaging, Vol. 17, No. 6, 1998

[7] Y. Wang, E.K. Teoh and D. Shen, "Structure-Adaptive B-Snake for Segmenting Comp1ex objects," International Conference on Image Processing (ICIP 2001), Thessaloniki, Greece, Oct 7-10, 2001, pp. 769-772

[8] A.L.N. Fred, J.S. Marques, and P.M. Jorge, "Hidden Markov models vs syntactic modeling in object recognition, “Proceedings of International on Image Processing (ICIP'97),199, pp. 893-896.

[9] J. Flusser and T. Suk, “Pattern recognition by an affine moment invariant,” Pattern Recognition, Vol. 26, pp. 167–174, 1993.

[10] Cheolha Pedro Lee, “Robust Image Segmentation using Active Contours: Level Set Approaches”, Degree of Doctor of Philosophy, North Carolina State University, pp (9-22).

[11] R. Gonzales and R. Woods, “Digital Image Processing”. Addison-Wesley Publishing, 1st ed. 1993.

[12] B. Fisher, S. Perkins, A.Walker, and E.Wolfart, “Hypermedia image processing reference: Digital filters.” http://www.cee.hw.ac.uk/hipr/html/filtops.html,
http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm
04-2013.

[13] K. Engel (2006). *Real-time volume graphics,* pp. 112–114.
http://en.wikipedia.org/wiki/Sobel_operator 04-2013

[14] J. Canny, “A computational approach to edge detection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, no. 8, p. 769, 1986.

[15] http://members.tripod.com/asim_saeed/paper.htm.04-2013

[16] W. Snyder and H. Qi, Machine Vision. Cambridge University Press, 2004.

[17] Hiriyannaiah, G. Bilbro, and W. Snyder, “Restoration of locally homogeneous images using mean field annealing,” Journal of the Optical Society of America (A), vol. 6, pp. 1901–1912, December 1989.

[18] G. Bilbro and W. Snyder, “Mean field annealing, an application to image noise removal,” Journal of Neural Network Computing, Fall 1990.

[19] J. Roerdink and A. Meijster, “The watershed transform: Definitions, algorithms, and parallel strategies,” Fundamenta Informaticae, vol. 41, pp. 187–228, 2000.

[20] H. Heijimans, Morphological Image Operators. Boston: Academic Press, 1994.

[21] H. Nguyen, M. Worring, and R. Boomgaard, “Watersnakes: Energy-driven watershed segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, pp. 330–342, March 2003.

[22] D. Gabor, “Theory of communication,” Journal of IEE, vol. 93, pp. 429–457, 1946.

[23] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes, active contour model,” International Journal of Computer Vision, pp. 321–331, 1988.

[24] S. Osher and J. Sethian, “Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations,” Journal of Computationl Physics, pp. 12–49, 1988.

[25] Vedad Hadziavdic, “A Comparative Study of Active Contour Models for Boundary Detection in Brain Images Vedad”, Diploma Project, Faculty of Mathematical and Natural Sciences University of Tromso, pp (16-21).

[26]Chenyang Xu, “Deformable Models with Application to Human Cerebral Cortex Reconstruction from Magnetic Resonance Images”, Degree of Doctor Philosophy, University of Johns Hopkins, Baltimore Maryland, April 2000.

[27] I. Cohen, L. D. Cohen, and N. Ayache, "Using deformable surfaces to segment 3-D images and infer differential structures," *CVGIP: Image Understanding,* vol. 56, pp. 242-263, Sept. 1992.

[28] L. D. Cohen, "On active contour models and balloons," *CVGIP: Image Urulerstanding,* vol. 53, pp. 211-218, Mar. 1991.

[29] D. Collins, C. Holmes, T. Peters, and A. Evans, "Automatic 3-D model-based neuroanatomical segmentation," *Human Brain Mapping,* vol. 3, pp. 190-208, 1995.

[30] A. Blake and A. Zisserman, *Visual Reconstruction.* Boston: MIT Press, 1987.

[31] L. Davis, *Genetic Algorithms and Simulated Annealing.* London: Pitman, 1987.

[32] A. C. W. Kotcheff and C. J. Taylor, "Automatic construction of eigenshape models by genetic algorithm," in *the XVth Int. Conf. Inf. Proc. Med. Imag. (IPMI)*, pp. 1-14, Springer-Verlag, 1997.

[33]L. D. Cohen and I. Cohen, "Finite-element methods for active contour models and balloons for 2-D and 3-D images," *IEEE Trans. on Pattern Anal. Machine Intell*, vol. 15, pp. 1131-1147, Nov. 1993.

[34] T. McInerney and D. Terzopoulos, "Déformable models in medical image analysis: a survey," *Medical Image Analysis*, vol. 1, no. 2, pp. 91-108, 1996.

[35] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. on Computers*, vol. 22, no. 1, pp. 67-92, 1973.

[36] B. Widrow, "The "rubber-mask" technique," *Pattern Recognition*, vol. 5, pp. 175-211, 1973.

[37] B. Leroy, I. Herlin, and L. D. Cohen, "Multi-resolution algorithms for active contour models," in *12th International Conference on Analysis and Optimizatian of Systems*, pp. 58-65, 1996.

[38] P. M. Morse and H. Feshbach, *Methods of Theoretical Physics.* New York: McGraw-Hill Book Company, 1953.

[39] J. L. Prince and C. Xu, "A new external force model for snakes," in 1996 *Image and Multidimensional Signal Processing Workshop*, pp. 30-31, 1996.

[40] A. K. Jain, *Fundamentals of Digital Image Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1989.

[41] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Anificial Intelligence*, vol. 17, pp. 185-203, 1981.

[42] W. F. Ames, *Numericol Methods for Partial Differential Equations.* Boston: Academie Press, 3rd ed, 1992.

[43] Chenyang Xu, Jerry L. Prince" Generalized gradient vector Flow external forces for active contours" Image Analysis and Communications Laboratory, Department of Electrical and Computer Engineering, The Johns Hopkins University,Baltimore, MD 21218, USA,

[44] Zeyun Yu and Chandrajit Bajaj, "Image Segmentation Using Gradient Vector Diffusion and Region Merging", Department of Computer Science, University of Texas at Austin, Austin, Texas 78712, USA

[45] C. Y. Hsu, S. H. Chen and K. L Wang "Active Contour Model with a Novel Image Force Field", 1 6th IPPR Conference On Computer Vision, Graphic and Image Processing, 17-19 Aug, pp. 477 -483, 2003.

[46] C. X u, J. L Prince, 'Generalized gradient vector flow external forces for active contours", Signal Process. vol. 71, pp. 131-139, 1998.

[47] C. Li, J. Liu and M. D. Foxa, Segmentation of external force "field for automatic initialization and splitting of snakes", Pattern Recognition. vol. 38, pp. 1947 - 19602005.

[48] B. Li and S. T. Acton, Active Contour External Force U sing Vector Field Convolution for Image Segmentation", IEEE Transactions on Image Processing, vol. 16, N08, pp. 2069- 106, 2007.

[50] YUE WANG AND EAM KHWANG TEOH, "Object Contour Extraction Using Adaptive B-Snake Model", School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, 13 March 2006

[51] Davar Giveki , Gholamreza Bahmanyar, Mohammad Ali Soltanshahi, Younes Khademian and Hamid Salimi "Object Boundary Recognition Using B-Snake", Canadian Journal on Image Processing and Computer Vision Vol. 2 No. 6, July 2011

[52] Y. Wang, E.K. Teoh, and D. Shen, "Lane detection and tracking using B-Snake", Image and Vision Computing; Vol. 22, No. 4, 2004, pp. 269-280.

[53] C. Xu and J.L. Prince, "Snakes, shapes, and gradient vector Flow," IEEE Transactions on Image Processing, pp. 359-369, March 1998.

[54] Y. Wang, "abject segmentation and matching using B-Spline model", PhD thesis Nanyang Technological University, Singapore, May 2004.

[55] Juan M. Collado, Cristina Hilario, Arturo de la Escalera, and Jose M. Armingol, Adaptative Road Lanes Detection and Classification, Intelligent Systems Lab, Universidad Carlos III de Madrid, Spain.