Automatic Active Contour Modelling and Its Potential Application for Non-Destructive Testing

Srinath Vepathur Sitaraman
Louisiana State University and Agricultural and Mechanical College

AUTOMATIC ACTIVE CONTOUR MODELLING AND ITS POTENTIAL APPLICATION FOR NON-DESTRUCTIVE TESTING

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

in

The Department of Electrical and Computer Engineering

By
Srinath Vepathur Sitaraman
B.E., University of Madras, May 2004.

December 2006.

Acknowledgements

First and foremost, I would like to sincerely express my gratitude to Dr. Hsiao-Chun Wu for his constant encouragement, guidance and for being very supportive right throughout my studies at LSU. I would also like to thank Dr. Subash Kak for accepting to be part of my theses defense committee. I would like to take this opportunity to thank him for his enlightening discussions in his classroom sessions. I would also like to express my sincere thanks to Dr. Suresh Rai for sparing his time to serve in my defense committee.

I would like to thank all my friends here at LSU for their support and for having made my stay to be a pleasant experience. I would like to thank my employers at LSU, Dr. Sai Pinnepalli and Ms. Melissa Brocato for the financial assistance provided to me while studying at LSU. I would also like to acknowledge Southern University, Louisiana for providing equipment for obtaining sample images for my theses.

Last but not least, I would like to thank my parents and sisters to have made my life so blissful with their love and blessings. Without their unflinching support and guidance, I couldn’t have come all this way.

Table of Contents

List of Tables

List of Figures

Abstract

Active contouring techniques are very useful in medical imaging, digital mapping, non-destructive ultrasonic evaluation, etc. Therefore, we try to explore and investigate the advanced automatic active contouring methods, which can benefit the aforementioned applications. In this thesis work, we study the automatic active contour models adopted for the object characterization, whose extensions could include the potential defect analysis in the ultrasonic non-destructive testing. The active contouring scheme (also called a snake) or an energy minimizing spline, is an algorithm which is very sensitive to the manually marked initial points, and thereby requires an expertly operation. Therefore, we make new research endeavors to handle this major problem and design a new expert-free snake technique, which can lead to the completely automatic contouring technology for the future applications. Even though this initialization problem has been addressed in the literature for quite a while, to the best of our knowledge, there exists no satisfactory solution so far. In this thesis, we propose a novel initialization algorithm for the automatic snake technique, which can possess a faster convergence than other existing automatic contouring methods and also avoid the human operational error incurred in the conventional snake schemes.

Chapter 1: Introduction

1.1 Problem Statement

Ultrasonic non-destructive testing (NDT) is usually adopted by the manufacturing industry to spot the defects such as cracks or inhomogeneity in materials for quality assessment. Resulting ultrasonic images can provide the crucial information for monitoring the manufacturing process. Such crucial information includes the crack shapes and sizes, the defect population and so on. Hence, the computer-aided techniques for studying and investigating the defects are in high demand of the NDT technology nowadays. *Active contour models*, first proposed by Kass et. al., are adopted in a variety of applications including edge detection, shape modeling, image segmentation and motion tracking [1]. In addition, it can be very useful for studying defects and cracks in materials.

However, two major problems still remain in the active contour models. First, an appropriate initial contour, which has to be as close to the true boundary as possible, has to be mandated by an expert. Otherwise, the active contour will likely converge to the unsatisfactory result [2]. Second, active contour models do not tend to progress into the cavities unless an appropriate initialization is appointed. Hence, in this thesis work, we make dedications to the research of the automatic active contour models which can provide the expert-free automatic initialization and the robust contouring technique for the NDT in the future.

1.2 Motivation

Several new parameters and energy functionals have been proposed since the active contour models were introduced in [1]. However, there still remain some problems in this line of research. One example, the initialization problem, has hindered the potential application in practice but it has not been tackled with any solution until very lately in [2, 3, 4]. Consequently, in this thesis, we provide a promising solution to result in the faster convergence of the active contour models than other existing schemes in [2, 3, 4].

1.3 Thesis Outline

The outline of this thesis can be organized as follows. Chapter 2 enunciates the demand for the object characterization in the non-destructive testing. The active contour models, also called snakes, are introduced and discussed in Chapter 3. The major initialization problem incurred in the active contour models is addressed in Chapter 4, wherein the existing techniques to combat this problem are also surveyed. We propose a novel effective initialization algorithm and design an automatic active contouring method, which are presented together with simulation results in Chapter 5. Ultimately, the concluding remarks will be drawn in Chapter 6.

Chapter 2: Object Characterization in Non-destructive Testing

2.1 Non – Destructive Testing

Many existing mechanical characterization techniques demand the destructive sectioning of the material sample to provide the desired physical or mechanical measures. On the other hand, non-destructive testing (NDT) techniques denote those which produce no material alteration, i.e., it is free of any intrusion or structural damage in the material sample [5]. In other words, an object can be examined very closely, without affecting its usefulness in anyway. The NDT can also be referred as *non-destructive evaluation* (NDE), *non-destructive characterization*, or *non-destructive inspection* [6]. Reliability measurement for quality assessment can also be performed using NDT.

The NDT applications can be found not only in the industrial and medical fields but also in our daily life. For industrial applications, NDT can be utilized in the fabrication of metals, non-metals and all materials containing particles. The NDT can also be utilized to detect cracks, imperfect welds, junctions, inclusions, and surface contamination without any material alternation. Thus, we are motivated to develop novel reliable NDT techniques due to its importance. Our objective of this thesis is to process the ultrasonic C-scan imaging data to extract the essential NDT information.

2.2 Various Existing NDT Methods

Prior to our discussion on the underlying ultrasonic imaging techniques in this thesis, we also introduce other existing NDT methods here. There are five different NDT techniques [7], namely, radiography, magnetic particle inspection, liquid penetrant inspection, Eddy current testing, and ultrasonic inspection. They are introduced as follows:

a) Radiography:

X-rays are used to produce the object images through the film or other detectors sensitive to the radiation. The test object is placed between the radiation source and the detector. The thickness and the density of the test material for x-rays to penetrate affect the radiation amount received at the detector. This variation in radiation produces an image on the detector that often shows the internal features regarding the test object. This method is used to inspect any material for its surface and subsurface defects. X-rays can also be used to locate and measure the internal features, to confirm the location of hidden parts in an assembly, and to measure the thickness of the material [7].

b) Liquid Penetrant Inspection:

Liquid penetrant inspection (LPI) is a simple but effective method of examining the surface areas for cracks, defects or structural discontinuities [8]. The LPI method is used very extensively, even more often than any other method, because it is relatively simple and inexpensive to carry out, and there are very few limitations on the specimen material or geometry. The LPI equipment is very simple and the inspection can be performed at many stages in the production of the material.

During LPI, the penetrant solution is applied to the surface of a pre-cleaned component. The liquid is pulled into the surface-breaking defects by the capillary action. Excess penetrant material is carefully cleaned from the surface. A developer is applied to pull the trapped penetrant back to the surface when it is spread out and forms an indication. The indication is much easier to see than the actual defect [7].

Then, the penetrant testing is used to locate cracks, porosity, and other defects that break the surface of a material and have enough volume to trap and hold the penetrant solution. LPI is used to inspect the large areas very efficiently and will work for most non-porous materials.

c) Magnetic Particle Inspection:

Magnetic particle inspection (MPI) is primarily applicable to the detection of surface-breaking discontinuities, particularly cracks. It can also detect discontinuities just below the surface but its sensitivity diminishes rapidly with depth. MPI can be easily manipulated without lengthy procedures and does not require the surface preparation work as the LPI described in the previous section. These advantages make MPI a popular NDT method. In principle, MPI utilizes the magnetic fields and the small magnetic particles, such as iron fillings to detect the flaws in the test samples.

A magnetic field is established in a component made from Ferro-magnetic material. The magnetic lines of force, travel through the material and exit and re-enter the material at the poles. Defects such as cracks or voids cannot support much flux, and then force some of the flux outflow the material. Magnetic particles distributed throughout the component will be attracted to the areas of flux leakage and produce a visible indication. It is noted that, in the MPI, the careful examination and evaluation must be carried out, and the additional demagnetization and cleaning work of the specimen are needed after the inspection [7].

d) Eddy Current Testing:

The eddy current testing technique is based on inducing electron flow (eddy currents) in electrically conductive material. Any defect such as cracks, inclusions, discontinuities etc., disrupts the flow of eddy currents. A varying electric current flowing in a coil gives rise to a varying magnetic field. A nearby conductor resists such an effect of the varying magnetic field. This is manifested by an eddy current flowing in a closed loop in the surface layer of the conductor so as to oppose the change and drive a back electromotive force (emf) in the coil.

Eddy currents produce their own magnetic fields that can be measured and used to find the flaws and characterize the conductivity, the permeability, and the dimensional features. This method is used to detect the surface and near-surface flaws in conductive materials such as metals. Eddy current inspection is also used to sort materials based on electrical conductivity and magnetic permeability, and measure the thickness of the thin sheets of metal and non-conductive coatings such as paint.

e) Ultrasonic Testing:

Ultrasonic Testing (UT) is a NDT method which can be used for the volumetric inspection of the underlying object. UT is carried out using the acoustic waves with high frequency above the audible range, that is, above 20 KHz, which are known as ultrasound. Thus, this non-destructive testing method is called *ultrasonic NDT*. The frequencies used in the ultrasonic testing range from less than 0.1 MHz to greater than 15 MHz and the typical values of wavelengths in the ultrasonic testing are from 1 to 10 mm.

The ultrasonic testing has been applied in the scientific and engineering fields for more than 70 years up to now. A very important development in this field arose from the studies of Firestone in 1940 and Simmons in 1945, namely, the pulsed ultrasonic testing using the echo principle [9]. The pulse-echo method [10] in which the same transducer transmits and receives the ultrasonic pulses has become the ubiquitous contemporary ultrasonic testing system.

Ultrasonic analysis for material characterization is based on a simple physics principle: the motion behavior of any acoustic wave will be influenced by the medium through which it travels. Hence, the structural discontinuities and the defects give rise to the scattering and the reflection of the waves, and the detection of the reflected or transmitted waves is related to the localization of the defects [11]. Thus, the changes in one or more of four easily measurable parameters associated with the passage of a high frequency sound wave through a material, namely *transit time*, *attenuation*, *scattering* and *frequency content*, can often be correlated with the changes in physical properties such as hardness, elastic modulus, density, homogeneity, or grain structure.

In this method high frequency sound waves are sent into a material by use of a transducer. The sound waves travel through the material and are received by the same transducer or a second transducer. The amount of energy transmitted or received is analyzed to determine the presence of flaws. Changes in material thickness and variations in material properties can also be measured. This method is used to locate the surface and sub-surface defects in many materials including metals, plastics and wood. Ultrasonic inspection is also used to measure the thickness of materials and characterize the properties of materials based on the sound velocity and attenuation measurements.

As a matter of fact, no single NDT method will work for all mechanical characterization applications. Each of the aforementioned methods has its own advantages and disadvantages when compared to other methods. The table below summarizes the essential properties regarding these NDT methods.

Table 2.1: Comparison of Different Non-Destructive Testing Techniques [6]

		Test	**Method**		
	Ultrasonics	**X-ray**	**Eddy Current**	**MPI**	**LPI**
Capital cost	Medium to high	High	Low to medium	Medium	Low
Consumable cost	Very low	High	Low	Medium	Medium
Time of results	Immediate	Delayed	Immediate	Short delay	Short delay

Table 2.1 (contd.)

Effect of geometry	Important	Important	Important	Not too important	Not too important
Access problems	Important	Important	Important	Important	Important
Type of defect	Internal	Most	External	External	Surface break
Relative Sensitivity	High	Medium	High	Low	Low
Formal record	Expensive	Standard	Expensive	Unusual	Unusual
Operator skill	High	High	Medium	Low	Low
Operator training	Important	Important	Important	Important	
Training needs	High	High	Medium	Low	Low
Portability of equipment	High	Low	High to medium	High to medium	High
Dependent on material composition	Very	Quiet	Very	Magnetic only	Little
Ability to automate	Good	Fair	Good	Fair	Fair
Capabilities	Thickness gauging; some composition testing	Thickness gauging	Thickness gauging; grade sorting	Defects only	Defects only

Due to its special advantages of high sensitivity, good detection, strong penetration ability, and harmlessness to human body, ultrasonic inspection has become the most important NDT method [12]. Therefore, we focus on the signal processing algorithms for the ultrasonic images in this thesis work.

2.3 Object Characterization

NDT has been widely adopted for monitoring the structural health and determining the mechanical properties associated with materials. For example, *ultrasonic imaging* (UI) is a versatile NDT technique [13], which is capable of testing the materials ranging from metals, ceramics to polymers. This UI technique, as described above, is widely used for detecting the defects confined within the material, such as cracks, voids and inclusions [14 - 16]. Boundary information of the underlying object in an ultrasonic image plays an important role in many applications and it is the basis for the quantitative and qualitative analyses such as area and volume measurements, object characterization, motion analysis, three-dimensional assessment, and so on [17].

Although ultrasonic images can provide useful information about the structure of an object, the factors like speckles, textures and artifacts resulting from the ultrasonic imaging process can lead to the incorrect interpretation of the data. Given such circumstances, the deformable mathematical models appear to be the better solution for the object shape analysis. Following this motivation, we explore the efficient and robust active contour models for objective characterization in this thesis.

Chapter 3: Active Contour Models

3.1 Introduction to Active Contour Models

"Classical" image segmentation algorithms, such as region growing, edge detection and relaxation labeling, usually require huge computation burden for generating satisfactory results [18]. Therefore, we have to explore an alternative technique, or the active contour model, which is also called *snake*. Using the snake algorithm, one is able to dynamically track the object deformation and the outcomes are remarkable compared to other aforementioned methods. Thus, active contours (snakes) are very promising for image segmentation. The snake model arose from the regularization paradigm which was applied to the computer vision technology in which the vision was seen as a general under-constrained problem of inverse optics [19].

There are two typical active contour models: parametric active contour [1] and geometric active contour models [20]. In this thesis, we will only consider the parametric active contour model, which synthesizes the parametric curves within an image domain and allows them to move towards the desired image features, usually edges. Snakes are widely used in many applications, including edge detection, shape modeling, segmentation and motion tracking. However, it is important to note that, the time consumed by the first few iterations will dominate the overall computational time. In addition, the contouring outcomes are very sensitive to the initial object shapes and hence the initialization task demands on the expertly maneuver. With appropriate initialization, the snakes can evolve to the actual boundaries of the subject contours and track the corresponding variations reliably.

3.2 The Snake Model

A snake is an active contour; first proposed by Kass et al. [1]. It can be used as a tool to solve the image segmentation problem. As a matter of fact, a snake is nothing but an *energy-minimizing spline* which is controlled by three types of forces, namely *internal energy-minimizing forces*, *external constraint forces* and *image forces*. The aggregate effect among those forces will drive the snake towards the salient image features (subject contours), such as lines and edges. In a nutshell, active contours are a series of connected points which move under the influence of internal forces (internal energy-minimizing forces) determined by the snake itself, and of external forces (external constraint forces and image forces) derived from the image data. The snake will finally stop once its internal and external forces reach the equilibrium.

In mathematics, the snake is usually expressed as a parametric curve $\vec{v}(s) = [x(s), y(s)]$, $s \in [0,1]$, which moves in the spatial domain of an image to minimize the energy functional, i. e.,

$$E_{snake} \equiv \int_0^1 \left\{ E_{\text{int}}[\vec{v}(s)] + E_{\text{ext}}[\vec{v}(s)] + E_{\text{image}}[\vec{v}(s)] \right\} ds, \tag{3.1}$$

where $E_{\text{int}}[\vec{v}(s)]$ represents the internal energy of the contour due to the bending or the discontinuities of the snake itself, $E_{\text{image}}[\vec{v}(s)]$ represents the energy associated with the image forces, and $E_{\text{ext}}[\vec{v}(s)]$ represents the external constraint energy. The internal energy, which maintains the snake's structure and deformation, is given by

$$E_{\text{int}}[\vec{v}(s)] \equiv \frac{1}{2}\left\{\alpha(s)\| \vec{v}'(s) \|^2 + \beta(s)\|\vec{v}''(s)\|^2\right\}, \tag{3.2}$$

where $\alpha(s)$ and $\beta(s)$ are the weighting parameters that control the snake's tension and rigidity, respectively, and $\vec{v}'(s) \equiv \frac{d\,\vec{v}(s)}{d\,s}$, $\vec{v}''(s) \equiv \frac{d^2\,\vec{v}(s)}{d\,s^2}$ denote the first and second derivatives of $\vec{v}(s)$ with respect to s.

The values of $\alpha(s)$ and $\beta(s)$ at a point determine the extent to which a contour is allowed to stretch or bend. If $\alpha(s) = 0$, the discontinuities can occur at that point, and if $\beta(s) = 0$, the curve discontinuities (i.e., a 90° sharp corner) can arise. The first and second derivatives of $\vec{v}(s)$ can be approximated using the finite differences by Kass et. al. [1], such that

$$\left\|\frac{d\vec{v}_i}{ds}\right\|^2 \cong \| \vec{v}_i - \vec{v}_{i-1}\|^2 = (x - x_{i-1})^2 + (y - y_{i-1})^2, \tag{3.3}$$

and, $$\left\|\frac{d^2\vec{v}_i}{ds^2}\right\|^2 \cong \| \vec{v}_{i-1} - 2\vec{v}_i + \vec{v}_{i+1}\|^2 = (x_{i-1} - 2x_i + x_{i+1}) + (y_{i-1} - 2y_i + y_{i+1}), \tag{3.4}$$

where $\vec{v}_i \equiv (x_i, y_i)$, for the coordinates of the contour at the iteration i.

Given a gray-level image $I(x, y)$ (a function of continuous position variable (x, y)), the typical external energies designed to drive an active contour towards step edges are [1]:

$$E^1_{\text{ext}}(x, y) \equiv -\left| \nabla I(x, y) \right|^2 \tag{3.5}$$

$$E^2_{\text{ext}}(x, y) \equiv -\left| \nabla\left(G_\sigma(x, y) \otimes I(x, y)\right) \right|^2 \tag{3.6}$$

where $G_\sigma(x, y)$ is a two-dimensional Gaussian kernel with standard deviation σ and ∇ is the gradient operator. The edge map resulting from Eq. (3.6) is obviously more informative than that resulting from Eq. (3.5), especially when noise is present. If the image is a sharp line (black on white), then appropriate external energies should include

$$E^3_{\text{ext}}(x, y) \equiv I(x, y), \tag{3.7}$$

and

$$E^4_{\text{ext}}(x, y) \equiv G_\sigma(x, y) \otimes I(x, y). \tag{3.8}$$

According to Eqs. (3.6), (3.8), the larger the σ, the more the resulting boundaries become blurry and distorted. However, such large σ's are often needed in order to increase the "capture range" of the active contour. The magnitude of the external forces diminishes quite rapidly away from the object boundaries. It turns out to be a dilemma that the capture range can be improved by a large σ, but the boundary localization of a snake appears less accurate and distinct. When σ is chosen too large, the Gaussian kernels obliterate the concavity itself ultimately (smear the object). This problem can be tackled using a multi-resolution approach; nevertheless, it leads to the extremely complicated algorithms.

The edge map $E^{\text{i}}_{\text{ext}}(x, y)$, i = 1, 2, 3, or 4, as defined by Eqs. (3.5) - (3.8), possess the property that they have the larger magnitudes close to the object boundaries. An edge map can be either gray-level or binary, though a binary edge map is often used. For notional convenience, we may refer to all kinds of edge maps as $f(x, y)$ such that

$$f(x, y) = E^{\text{i}}_{\text{ext}}(x, y), \tag{3.9}$$

where i = 1, 2, 3, or 4. Apart from Gaussian filtering as given in Eqs. (3.6) and (3.8), we may also apply other complicated noise-reduction techniques such as median filtering, morphological filtering, and anisotropic diffusion to improve the underlying edge maps.

To illustrate the effect of the process specified by Eq. (3.6), an example is presented here. An ultrasonic C-scan image scanned at 1MHz frequency is depicted in Figure 3.1 (a).

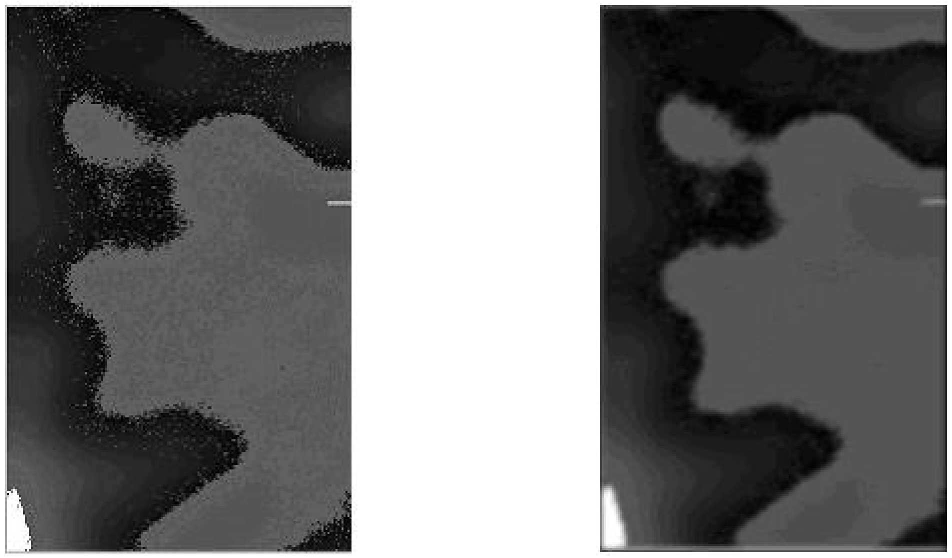


Figure 3.1. Gaussian blurring effect: (a) original image, (b) original image blurred by Gaussian filter with $\sigma = 1.5$.

Note that, when the image in Figure 3.1 (a) is filtered by a Gaussian kernel with $\sigma = 1.5$, the resulting image becomes smooth and some noise present in the original image is reduced as shown in Figure 3.1 (b). Obviously from Figure 3.1, Gaussian filtering significantly reduces the speckle noise, which is very often found in the medical and ultrasonic images.

An edge map defined in Eq. 3.9 has the three important properties in the context of active contour models as follows:

i) The gradient of an edge map, $\nabla f(x, y)$, results in the vectors pointing towards the edges which are normal to the tangents of the boundaries.

ii) These gradient vectors in general have large magnitudes in the immediate vicinity of the edges.

iii) In the homogeneous regions, the gradient of the edge map is nearly zero, i.e., $\nabla f(x, y) \approx 0$.

Now, we will discuss how the aforementioned properties can affect the behavior of a snake, when a particular edge map is used as an external force. Because of the first property, a snake initialized close to the edge will converge to a stable configuration near the edge. This highly desirable property paves the foundation of our proposed algorithm in this thesis. Because of the second property, the capture range will be generally very small. Because of the third property, the homogeneous regions will induce no external forces at all (snake cannot evolve therein). Thus it is quite clear that the last two properties will cause the difficulty for the snake algorithm.

A snake that minimizes E_{snake} given by eqn. 3.1 must satisfy the Euler equation. That is

$$\alpha(s)\vec{v}'(s) - \beta(s)\vec{v}''(s) - \nabla E_{\text{ext}}[\vec{v}(s)] = \vec{0}. \tag{3.10}$$

To find a solution to this Euler equation, the dynamics of the snake is considered by treating $\vec{v}(s)$ as a function of time *t* as well as *s*, i.e., $\vec{v}(s)$ is referred as $\vec{v}(s,t)$. Then the partial derivative of $\vec{v}(s)$ with respect to *t* is then set equal to the left hand side of Eq. (3.10), such that

$$\frac{\partial \vec{v}(s,t)}{\partial t} = \alpha(s)\vec{v}'(s) - \beta(s)\vec{v}''(s) - \nabla E_{\text{ext}}[\vec{v}(s)]. \tag{3.11}$$

When the solution $\vec{v}(s,t)$ converges, the term $\frac{\partial \vec{v}(s,t)}{\partial t}$ vanishes and a solution $\vec{v}_i$ to the Euler equation given by (3.10) can be found. A solution to (3.11) can be found by discretizing the variables and solving the equation iteratively. Eq. (3.1) can be approximated as

$$E_{snake} \approx \sum_{i=1}^{n} \left\{ E_{\text{int}}(i) + E_{\text{ext}}(i) + E_{\text{image}}(i) \right\}, \tag{3.12}$$

where $E_{\text{int}}(i)$ is the instantaneous internal energy for the i^{th} iteration and obtained when $\vec{v}(s)$, $\alpha(s)$, $\beta(s)$ are substituted with $\vec{v}_i = (x_i, y_i)$, α_i, β_i, respectively in Eq. (3.2), such that

$$E_{\text{int}}(i) = \alpha_i \| \vec{v}_i - \vec{v}_{i-1} \|^2 + \beta_i \| \vec{v}_{i-1} - 2\vec{v}_i + \vec{v}_{i+1} \|^2. \tag{3.13}$$

When we treat *x*(*s*), y(*s*) are two independent variables as *x*, *y* for simplification, the Euler equation given by (3.12) can be re-written in a matrix form as

$$\widetilde{A}\,\vec{x} + \vec{E}_{ext,x} = \vec{0}, \tag{3.14.a}$$

$$\widetilde{A}\,\vec{y} + \vec{E}_{ext,y} = \vec{0}, \tag{3.14.b}$$

where $\widetilde{A} \in \mathrm{R}^{N\times N}$ is a *penta-diagonal parameter matrix* given by

$$\widetilde{A} \equiv \begin{bmatrix} c & d & e & 0 & 0 & 0 \\ b & c & d & e & 0 & 0 \\ a & b & c & d & e & 0 \\ 0 & a & b & c & d & e \\ 0 & 0 & a & b & c & d \\ 0 & 0 & 0 & a & b & c \end{bmatrix}, \tag{3.15}$$

$\vec{x} \in \mathrm{R}^{N\times 1}$ and $\vec{y} \in \mathrm{R}^{N\times 1}$ are the vectors specifying the snake coordinates (the total number of snake pixels or *snaxels* is *N*) along *x*- and *y*-axes, respectively, $\vec{E}_{ext,x} \equiv \left[\frac{\partial E_{ext}}{\partial x}\right] \in \mathrm{R}^{N\times 1}$, $\vec{E}_{ext,y} \equiv \left[\frac{\partial E_{ext}}{\partial y}\right] \in \mathrm{R}^{N\times 1}$ are the vectors specifying the partial derivatives with respect to *x* and *y* on the snake coordinates, and $\vec{0} \in \mathrm{R}^{N\times 1}$ is the zero-vector. The entries in $\widetilde{A}$ can be formulated as

$$a = \beta_{i-1}$$
$$b = -\alpha_i - 2\beta_{i-1} - 2\beta_i$$
$$c = \alpha_i + \alpha_{i+1} + \beta_{i-1} + 4\beta_i + \beta_{i+1}$$
$$d = -\alpha_{i+1} - 2\beta_i - 2\beta_{i+1}$$
$$e = \beta_{i+1}$$

Now, solving the Euler equation as a function of time, we obtain

$$\widetilde{A}\,\vec{x}_t + \vec{f}_x(x_{t-1}, y_{t-1}) = -\gamma\,(\vec{x}_t - \vec{x}_{t-1}), \tag{3.16.a}$$

and

$$\widetilde{A}\,\vec{y}_t + \vec{f}_y(x_{t-1}, y_{t-1}) = -\gamma\,(\vec{y}_t - \vec{y}_{t-1}), \tag{3.16.b}$$

where γ is the step-size, $\vec{x}_t \in \mathrm{R}^{N\times 1}$ and $\vec{y}_t \in \mathrm{R}^{N\times 1}$ are the vectors specifying the snake coordinates along the *x*- and *y*-axes at a given time *t*, $\vec{f}_x(x_{t-1}, y_{t-1}) \equiv \frac{\partial f(x_{t-1}, y_{t-1})}{\partial x}$, $\vec{f}_y(x_{t-1}, y_{t-1}) \equiv \frac{\partial f(x_{t-1}, y_{t-1})}{\partial y} \in \mathrm{R}^{N\times 1}$ are the vectors specifying the partial derivatives of $f(x, y)$ given by Eq. (3.9) with respect to *x* and *y* on the snake coordinates at the time *t*-1, respectively. At equilibrium, the time derivative vanishes and therefore $\vec{x}_t$ and $\vec{y}_t$ can be solved using the matrix inversion. Thus,

$$\vec{x}_t = (\widetilde{A} + \gamma \widetilde{I})^{-1} (\gamma \vec{x}_{t-1} - \vec{f}_x(x_{t-1}, y_{t-1})), \tag{3.17.a}$$

$$\vec{y}_t = (\widetilde{A} + \gamma \widetilde{I})^{-1} (\gamma \vec{y}_{t-1} - \vec{f}_y(x_{t-1}, y_{t-1})). \tag{3.17.b}$$

To add extra flexibility to the snake model, it is possible to start from the force balance equation directly, and then to replace $\vec{F}_{ext}^1$ by another force $\vec{F}_{ext}^2$, which needs not be irrotational as follows

$$\vec{F}_{\text{int}} + \vec{F}_{ext}^2 = \vec{0}. \tag{3.18}$$

Balloon models in [21] comprise an important example for this approach. In these models, $\vec{F}_{ext}^2$ is the sum of the traditional potential forces and pressure (or normal) forces, which act in a direction normal to the snake points. This force replacement increases the capture range of an active contour but also requires that the balloon to be initialized to either shrink or grow. In addition, the strength of the pressure forces may be difficult to set, because these forces must be large enough to overcome the weak edges and noise but small enough so that they conform to the true edge forces. Without pressure forces, two major problems arise, namely initialization and convergence to concave regions. Initialization is a problem because the capture range of the traditional potential force (the derivative of the external energy) is generally small. Convergence to concave regions can be a problem in conventional snakes, because the contour often remains split across the boundary concavities.

Several methods have been proposed to address these two problems, such as pressure forces [21], multi-resolution methods [22], and distance potentials [23]. However, most of the existing methods can only provide the partial solutions while creating new problems. For example, the multi-resolution methods have addressed the issue of capture range due to the initialization, but how to build a snake across different resolutions would be very difficult. The pressure forces can push an active contour into the boundary concavities, but they cannot be too strong; otherwise the weak edges will be overwhelmed by the snake or the snake could actually surpass the actual boundary to overdevelop. Pressure forces must also be pre-determined to push out or push in and therefore the careful initialization is needed.

3.3 Gradient Vector Flow Method

A new class of external forces, called *gradient vector flow* (GVF) field forces, for active contour models was proposed by Xu et al. in [24]. This GVF method significantly improved the performance of the traditional snake algorithm. GVF fields are dense vector fields derived from images by minimizing certain energy functional in a variational framework [24]. Subsequently, the active contour that uses the GVF field as its external force is called a GVF snake. Particular advantages of the GVF snake over a traditional snake are its insensitivity to initialization and its ability to move into boundary concavities. The GVF algorithm utilizes the properties of an edge map that were discussed earlier in Section 3.2. The GVF approach is to maintain the highly desirable property of the gradients near the edges. In the meantime, this algorithm extends the gradient map to the regions far away from the edges and the homogeneous regions, via a computational diffusion process.

The gradient vector flow field is defined as a vector field $\vec{\xi}(x,y) = [\eta(x,y), \psi(x,y)]$ that will minimize the energy functional

$$\varepsilon = \iint \left\{ \mu\,(\eta_x^2 + \eta_y^2 + \psi_x^2 + \psi_y^2) + \| \nabla f(x,y) \|^2 \, \| \vec{\xi}(x,y) - \nabla f(x,y) \|^2 \right\} dxdy, \tag{3.19}$$

where $\nabla f(x,y)$ is the gradient vector of the edge-map defined in Eq. (3.9) [24]. For notional convenience, we denote $f_x \equiv \dfrac{\partial f(x,y)}{\partial x}$, $f_y \equiv \dfrac{\partial f(x,y)}{\partial y}$, $\eta_x \equiv \dfrac{\partial \eta(x,y)}{\partial x}$, $\eta_y \equiv \dfrac{\partial \eta(x,y)}{\partial y}$, $\psi_x \equiv \dfrac{\partial \psi(x,y)}{\partial x}$, $\psi_y \equiv \dfrac{\partial \psi(x,y)}{\partial y}$ This formulation in Eq. (3.19) follows a standard principle, wherein the result is noted to be smooth when $\nabla f(x,y) \approx 0, \quad \forall x, \forall y$. In particular, it is noted that when $\| \nabla f(x,y) \|^2$ is small, the energy term is dominated by $(\eta_x^2 + \eta_y^2 + \psi_x^2 + \psi_y^2)$ and yields a slowly varying field. On the other hand, when $\| \nabla f(x,y) \|^2$ is large, the second term $\| \nabla f(x,y) \|^2 \, \| \vec{\xi}(x,y) - \nabla f(x,y) \|^2$ dominates the integrand and can reach the minimum by setting $\vec{\xi}(x,y) = \nabla f(x,y)$, $\forall x, \forall y$. This produces the desired effect of keeping ε nearly equal to $\iint \left\{ \| \nabla f(x,y) \|^2 \, \| \vec{\xi}(x,y) - \nabla f(x,y) \|^2 \right\} dxdy$ when $\| \nabla f(x,y) \|^2$ is large but forcing the field to be slowly-varying in the homogeneous regions. μ is a regularization parameter governing the tradeoff 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. That is, for a very noisy image, μ has to be large. Using the calculus of variations [25], the Gradient Vector Flow can be found by solving the following Euler equations,

$$\mu \nabla^2 \eta - (\eta - f_x)(f_x^2 + f_y^2) = 0, \qquad (3.20.a)$$

$$\mu \nabla^2 \psi - (\psi - f_y)(f_x^2 + f_y^2) = 0, \qquad (3.20.b)$$

where f_x and f_y are the partial derivatives of the edge-map *f* with respect to *x* and *y* respectively, ∇^2 is the laplacian operator which defines the smoothing term – the first term within the integrand of the energy functional in equation (3.19). The digital implementation of a two-dimensional laplacian operator is obtained by the summation of the second order derivative in the x- and y- directions. For example,

$$\nabla^2 \eta = \left[\eta(x+1,y) + \eta(x-1,y) + \eta(x,y+1) + \eta(x,y-1) \right] - 4\eta(x,y) \qquad (3.21)$$

Equation (3.20) is fundamental to the concept of GVF. It can be seen that, in homogeneous regions, the second term of the both the equations. in 3.20 (a) and 3.30 (b) is zero, as ∇f (gradient of the *f*(*x*,*y*)) is zero. Therefore, within homogenous regions, η and ψ are each determined by the Laplace's equation.

Equations 3.20 (a) and 3.20 (b) can be solved by treating η and ψ as functions of time and solving

$$\eta_t(x,y,t) = \mu \nabla^2 \eta(x,y,t) - (\eta(x,y,t) - f_x(x,y))(f_x^2(x,y) + f_y^2(x,y)) \qquad (3.22.a)$$

$$\psi_t(x,y,t) = \mu \nabla^2 \psi(x,y,t) - (\psi(x,y,t) - f_y(x,y))(f_x^2(x,y) + f_y^2(x,y)) \qquad (3.22.b)$$

The following examples show the computation of the gradient vector flow of an image. For ease of explanation, let us first consider a simple example where there is just a single white pixel (a dot) on a black background as shown in Figure 3.2 (a). Figure 3.2 (b) shows the result of computing just the gradient of the edge map which is the case in the traditional snake models. Such a computation shows vectors pointing towards the edge (dot in this case) from only near by range. However, Figure 3.2 (c) shows the result of computation of the gradient vector flow and we are able to see that vectors are pointing towards the dot even from a far off distance. Thus having such a gradient flow would help a traditional snake with a replaced external energy in the form of a gradient vector, deform more accurately towards the edge.

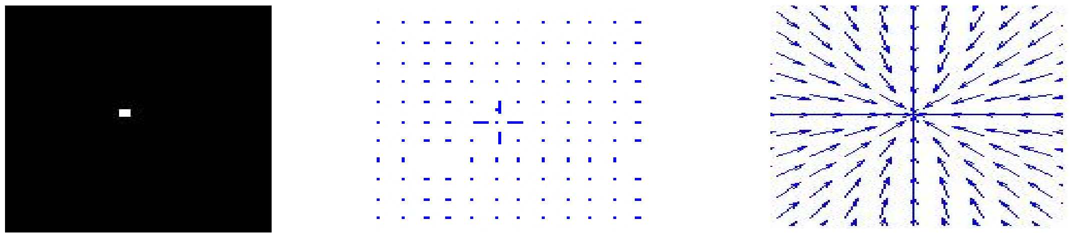


Figure 3.2. Gradient vector flow for a dot: (a) original image representing a dot, (b) Edge map gradient, (c) Normalized GVF Field.

The same GVF concept is explained as follows in Figure 3.3, where we deal with a more complex U-shape. Figure 3.3 (a) shows a U-shaped object, whose edge map gradient is shown in Figure 3.3 (b).

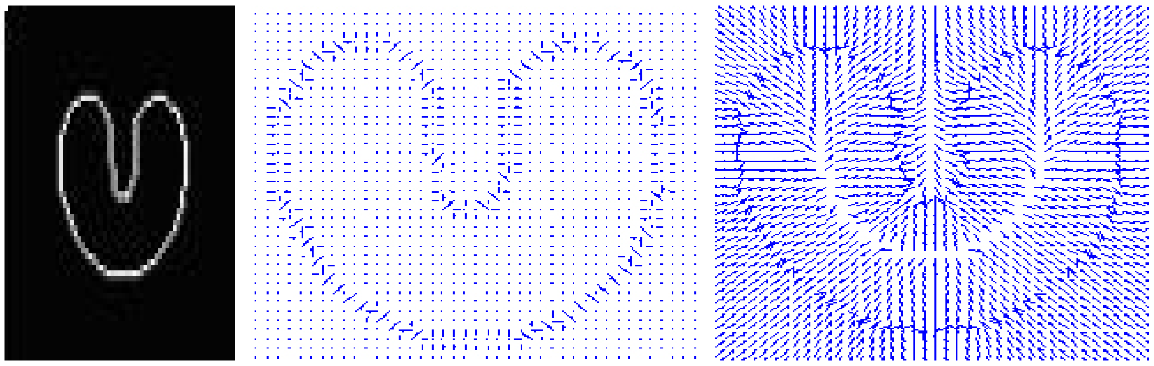


Figure 3.3. Gradient vector flow for U-shape: (a) original image representing a U-shape, (b) Edge map gradient, (c) Normalized GVF Field for the U-shape.

It is very clear from Figure 3.3 (b), that the resulting vectors are no good when a snake is initialized far off from these vectors. The problem of dealing with cavities also becomes more apparent in this case and the snake struggles to wiggle itself in to the U-shaped cavity. However, this problem is efficiently tackled with the computation of the GVF field, as shown in Figure 3.3 (c). Now, a snake initialized even from quite a far range can still conform to the boundary of the cavity by just following the long range directional gradient vectors. After $\xi(x,y)$ is computed, the negative gradient of the potential force $-\nabla E_{ext}$ in the traditional snake equation is replaced by $\xi(x, y)$ obtained from equation (3.22). Now, the gradient vector flow snake is driven by

$$v_t(s,t) = \alpha v'(s,t) - \beta v''(s,t) + \xi \tag{3.23}$$

Solution to this equation is derived in a fashion similar to that of the traditional snake – i.e. by discretization and iterative solution. Such a GVF snake will have a much larger capture range than the traditional snakes. We have discussed GVF in detail in this thesis because it has been thought to be an effective solution to the initialization problem.

3.4 GVF for Higher Dimensions

GVF can be easily generalized to higher dimensions [26] and is similar to the two-dimensional case discussed above. Let $f(x)$: $R^n \to R$ be an edge map defined in R^n. The gradient vector force field in R^n is defined as the vector field $\vec{\xi}$: $R^n \to R$ that minimizes the energy functional

$$\varepsilon = \int_{R^n} \left[\mu \left\| \nabla \vec{\xi} \right\|^2 + \left\| \nabla f \right\|^2 \left\| \vec{\xi} - \nabla f \right\|^2 \right] dx \tag{3.24}$$

where the gradient operator ∇ is applied to each component of $\vec{\xi}$ separately. Using the calculus of variations, it is found that the GVF field must satisfy the Euler equation

$$\mu \nabla^2 \xi - (\vec{\xi} - \nabla f) \left\| \nabla f \right\|^2 = \vec{0} \tag{3.25}$$

and ∇^2 is applied to each component of the vector field $\vec{\xi}$ separately. A solution to these Euler equations is found by introducing a time variable t and finding the steady state solution of the following linear parabolic partial differential equation:

$$\vec{\xi}_t = \mu \nabla^2 \vec{\xi} - (\vec{\xi} - \nabla f) \left\| \nabla f \right\|^2 \tag{3.26}$$

where $\vec{\xi}_t$ denotes the partial derivative of $\vec{\xi}$ with respect to *t*.

Chapter 4: An Overview of Automatic Active Contour Models

4.1 Limitations of Conventional Active Contour Models

The conventional *active contour models* have several shortcomings that limit their capability of resolving the image segmentation problems [27]. These limitations are summarized as follows:

i) The conventional active contour models are very sensitive to the initial contour positions and the internal parameters. The inappropriate choices make the snakes' behaviors unpredictable.

ii) Snakes are also very sensitive to noise and weak edges. A snake is susceptible to being drawn towards the undesired local minima. This can be solved often by Gaussian window smoothing. Nevertheless, how to choose an appropriate Gaussian window size still remains a problem.

iii) It is difficult to choose the proper time step for each iteration, which can lead to an appropriate snake convergence. It is highly likely that if the time-step is too large, an appropriate local minimum could be missed by the snake or an oscillation could occur between two *snaxels* (snake pixels) surrounding a local minimum.

iv) The manual initialization of active contours is not reliable and demands expertise. The initial contour should be placed as close to the true boundary as possible. However, manual initialization is subject to the human errors and hence it is not guaranteed to be successful. Improper connectivity of the initial points or large interceptions of the object pixels sometimes could hardly be perceived by human.

v) It is not easy to enforce a hard constraint such as keeping points a certain distance apart to avoid points on the contour converging.

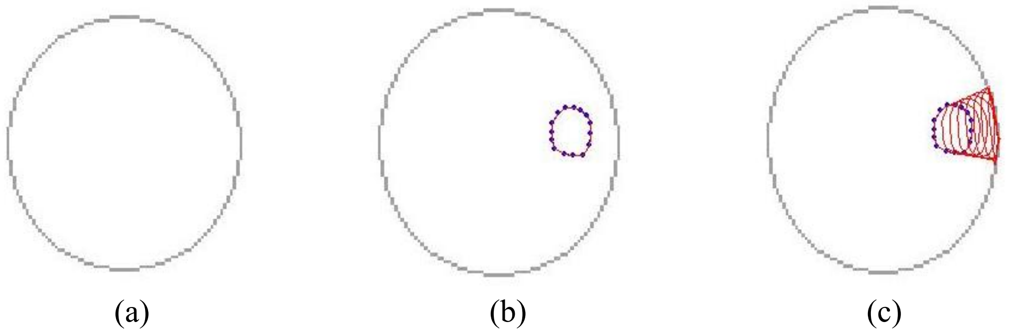
4.2 The Initial Contour Problem

Most of the aforementioned issues in Section 4.1 can be tackled if the initial contour is placed appropriately. As a consequence, researchers are looking for automatic initialization as a reasonable solution to these problems. This will remove the restriction of an expert's operation and in the meantime an effective initialization scheme will improve the performance of the snake. In this chapter, we will discuss a few existing schemes for automatic initialization of snakes. Gradient vector flow (GVF) in [24] or *Generalized-GVF* (GGVF) in [28] were proposed by Xu and Prince and they were effective. GVF (as described in Section 3.3) is an extension of the traditional snake's external forces, which is computed as a diffusion of the gradient vectors associated with a gray-level or binary edge map derived from the image. Since the GVF can expand the capture range remarkably, the requirement that the initial contour needs to be as close to the true boundary as possible is no longer stringent. However, a proper initial contour is still necessary; otherwise the active contour may converge to the undesired result despite the GVF employment.

4.3 Survey on Existing Initialization Methods

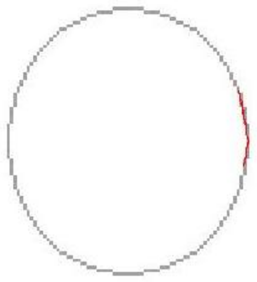
4.3.1 Center-of-Divergence Method

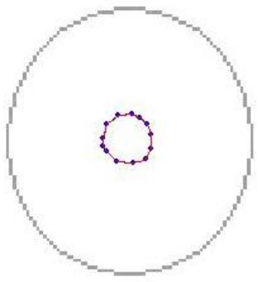
Even though GVF solves the initial contour problem to a certain extent, a proper scheme for automatic initialization is still necessary. This was first observed by Ge and Tian in [2]. Consider the following example in which the initial contour is placed in such a way that the gradient vector field force attracts the snake points towards the left of the object in Figure 4.1.

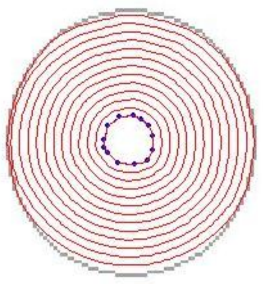


(a) (b) (c)

Figure 4.1. Effect of poor initialization in the GVF case: (a) A circle object is used as an example; (b) initial contour is placed closer to one end of the circle; (c) poor convergence of the active contour model is shown; (d) inappropriate final result of the snake is drawn to one end of circle; (e) another initial contour is placed around the center of divergence; (f) appropriate final result is achieved from the proper initial contour in (e). (fig. contd.)


(d)

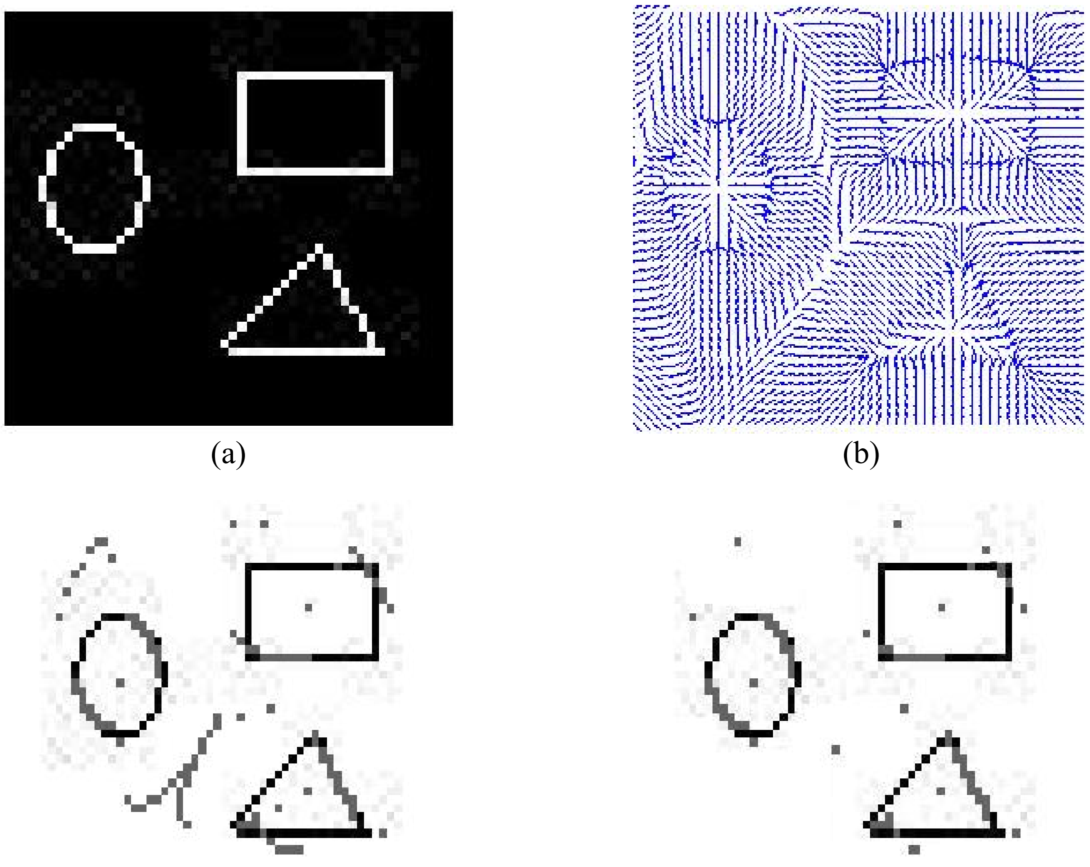

(e)


(f)

According to Figure 4.1, a circle object is depicted in Figure 4.1 (a). Figure 4.1 (b) illustrates the initialization of the active contour with a little circle which is eccentric to the desired boundary of the object, i.e. the large circle (figure 4.1 (a)). Figure 4.1 (c) shows the deformation of the GVF snake completely towards the left of the object. In Figure 4.1 (d), the contour has converged to an arc, along the object's left boundary, which obviously is an undesired result. This happens because the initial circle does not contain the "center-of-divergence" of the GVF field [2]. However, when an initial contour is place so that it encompasses the "center-of-divergence" of the GVF field (as in Figure 4.1 (e)), we obtain the desired result as shown in Figure 4.1 (f). Hence, according to Figure 4.1, it is very obvious that the initial contours must include the center of divergence for achieving the desired results, especially when GVF is used as the external energy for snake deformation. The concept of center-of-divergence, proposed by [2], is discussed in detail in this section.

Ge and Tian suggested in [2] that the initialization of contours depends on the points denoted as the *centers of divergence*. These points can be computed directly from the GVF field. Each center of divergence is used to generate one initial contour. After normal deformation, some desired as well as undesired contours are obtained. By an external energy criterion, these contours located at the true boundary locations are picked out.

We use an example in Figure 4.2 to illustrate the concept here. The computation of gradient vector flow can follow the procedure given in Section 3.3. The gradient vector flow fields of the image in Figure 4.2 (a) are computed and shown in Figure 4.2 (b).


(a) (b)

(c) (d)

Figure 4.2. Center of divergence concept: (a) an image with multiple objects, (b) the computed gradient vector flow for the image, (c) centers of divergence computed from the gradient vector flow for the image, (d) centers of divergence after eliminating some points in (c) by means of a minimum distance criterion.

Now, consider one object in Figure 4.2 (a), say "the circle". The corresponding GVF graph in Figure 4.2 (b) shows the fields pointing outwards from a center point. A section of the gradient vector flow of the circle is shown in Figure 4.3 for a close look at its *center of divergence*.

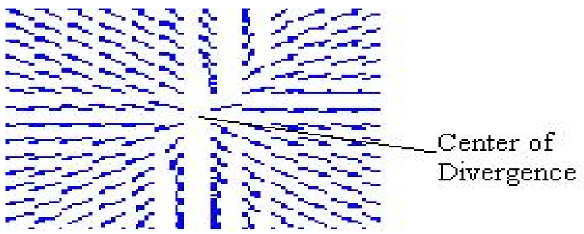


Figure 4.3. A close look at the center-of-divergence of the GVF of the circle shown in Figure 4.2 (a).

The criterion for finding such centers-of-divergence is discussed as follows. When the GVF vectors of four or more adjacent pixels diverge from a center point as shown in Figure 4.3, this center point is defined as a *center of divergence*. Consider four pixels adjoining each other, namely $p(i,j)$, $p(i+1,j)$, $p(i,j+1)$ and $p(i+1,j+1)$, where i and j are column and row indices. Also, $\bar{\xi}(i,j)=\left[\eta(i,j),\psi(i,j)\right]$ is the corresponding GVF vector of the pixel $p(i,j)$. Then the potential scattering point set P_s is defined as follows:

$$P_s\,\eta=\{(i,j)\mid \eta(i,j)\le\eta(i+1,j)\; and\; abs(\,sign(\eta(i,j))+sign(\eta(i+1,j))\,)\le 1\}, \qquad (4.1)$$

$$P_s\,\psi=\{(i,j)\mid \psi(i,j)\le\psi(i+1,j)\; and\; abs(\,sign(\psi(i,j))+sign(\psi(i+1,j))\,)\le 1\}. \qquad (4.2)$$

where the function sign is used to classify the direction of $\bar{\xi}(i,j)$:

$$sign(\eta)=\begin{cases}1;\eta>0\\ 0;\eta=0\\ -1;\eta<0\end{cases}. \qquad (4.3)$$

Then, $$P_s=\{P_s\eta,P_s\psi\}. \qquad (4.4)$$

The following table provides a detailed overview of the criteria described above:

Table 4.1. Detail review of the criterion used to discard and reserve points [2]

	$\eta(i,j)>\eta(i+1,j)$	$\eta(i,j)<\eta(i+1,j)$	$\eta(i,j)=\eta(i+1,j)$
$sign(\eta(i,j))+sign(\eta(i+1,j))=2$	Discarded $(\rightarrow\rightarrow)$	Discarded $(\rightarrow\rightarrow)$	Discarded $(\rightarrow\rightarrow)$
$sign(\eta(i,j))+sign(\eta(i+1,j))=1$	Discarded $(\rightarrow 0)$	Reserved $(0\rightarrow)$	Reserved
$sign(\eta(i,j))+sign(\eta(i+1,j))=0$	Discarded $(\rightarrow\leftarrow)$	Reserved $(\leftarrow\rightarrow)$	Reserved $(0\;\;0)$
$sign(\eta(i,j))+sign(\eta(i+1,j))=-1$	Discarded $(0\leftarrow)$	Reserved $(\leftarrow 0)$	Reserved
$sign(\eta(i,j))+sign(\eta(i+1,j))=-2$	Discarded $(\leftarrow\leftarrow)$	Discarded $(\leftarrow\leftarrow)$	Discarded $(\leftarrow\leftarrow)$

However, in the computation of P_s, there may be some points which are too close to each other. For example, the back-to-back points will appear in the P_s computed from the GVF field as shown in Figure 4.2 (c). However, they tend to represent the same center of divergence. Hence we need to eliminate such redundant points. That is, any of back-to-back points can be chosen to represent a reliable center-of-divergence. This can actually be performed by applying a criterion, where, if the distance between any two points in P_s is larger than a specified threshold D_{min} [2], then one of them can be dropped as shown in Figure 4.2 (d). After the centers of divergence are chosen, a contour is initialized as a small circle centering each one of them. The radius of the circle can be set equal to D_{min} or even a little bit larger [2]. Since the centers of divergence are more than one, there will be multiple active contours running simultaneously with respect to each center of divergence as shown in Figure 4.4.

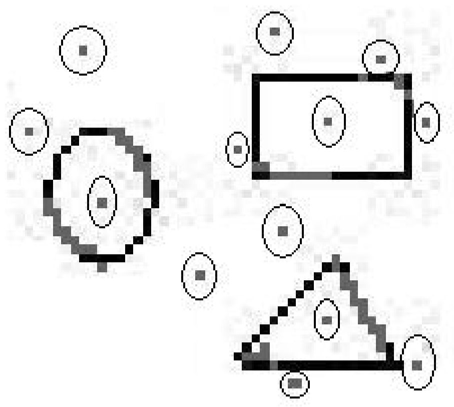


Figure 4.4. Initialization of contours around the centers of divergence as small circles of radius D_{min}.

Once the contours have been initialized, they move towards the desired boundary to satisfy the Euler equations. All of the vectors surrounding the centers of divergence run away from them. Therefore, the contours continue to deform until they arrive at the desired boundary. A contour is considered to have reached its destination when all of its snaxels have been inactive for a user-specified number of iterations. The contours that do not arrive at boundary after the user-specified number of iterations are considered to be irrelevant to the objects in the image. In Figure 4.4, there are totally 8 simultaneous active contours. While the contours developed within the object boundaries in the image are relevant, we will still have 5 irrelevant contours which will develop the boundaries shared by the outer edges of the objects. Such boundaries are called pseudo- or quasi-boundaries [2]. They should be discarded according to some criteria.

Ge et al. have also suggested a criterion for discarding the pseudo-boundaries based on the external energy of the contour [2]. This method is fairly intuitive and derived from the observation that unexpected contours often have a significant portion in the non-boundary areas. These portions have lower external energies than those lying on the boundaries. Thus, by mere comparison of external energies associated with each contour based on the gradient field of the edge map ∇f, from which the GVF field is computed, the pseudo-boundaries can be eliminated. $\|\nabla f\|^2$ for the pixels close to the true boundary is much higher than that far from the true boundary. The corresponding aggregate external energy of each active contour is significantly different from others and can be clearly distinguished. The algorithm is summarized as:

- Sort the contours C(i) by their external energies $E_{ext}(i)$ where $E_{ext}(i) > E_{ext}(i+1)$
- Calculate the difference Diff(i,i+1) = $(E_{ext}(i) - E_{ext}(i+1))/ E_{ext}(i+1)$ of the external energy between adjoining contours
- Find the largest Diff(j,j+1) among the differences and record its position j
- Use the position j as the dividing point; the contours $C(i)$, $i = 1,2,\cdots,j$, are discarded

However, there still exist some disadvantages in the aforementioned method as follows:

- Sometimes, too many irrelevant centers of divergence can be found, especially in the case of the images with deep cavities.
- Initialization of contours based on the center of divergence is very time consuming; especially for the images having objects with large spatial areas and deep cavities, the deformation time may be large.
- The application of having multiple active contour models for deforming simultaneously can sometimes be very demanding when the constraints on computational resource is a factor.
- The GVF computation of three-dimensional images is potentially a very time consuming process. Additional computational effort required to find centers of divergence may even make the process much slower.

- Sometimes the external energy based criterion can lead to the drawback for removing quasi-boundaries when the desired boundary has a number of gaps. This is because the external energy of the contour with gaps is lower than those without gaps. Therefore, the actual contours lying on the true boundaries with gaps may be classified as the quasi-boundaries to be eliminated.
- This algorithm may fail while working on the images with concentric objects. This may either result in too many centers of divergence or in some cases the outer objects may not have the well-defined centers of divergence for a proper initial contour.

4.3.2 Center-of-Strong/Weak Divergence

Another contour initialization approach based on the center of divergence concept was introduced by Tauber et al. in [3]. Here, the concept of center-of-divergence has been further characterized, wherein the *centers of divergence* have been further classified into the centers of strong divergence (Cd_{strong}) and centers of weak divergence (Cd_{weak}). They are defined as follows:

$$Cd_{strong} = \{(i,j) \mid (i,j) \in P_s\eta \; and \; (i,j) \in P_s\psi\}, \tag{4.5}$$

$$Cd_{weak} = \{(i,j) \mid (i,j) \in P_s\eta \; or \; (i,j) \in P_s\psi\}. \tag{4.6}$$

where $P_s\,\eta$ and $P_s\,\psi$ are the centers of divergence in the η- and ψ- directions respectively and they are defined in Eqs. (4.1) and (4.2). In [3], Tauber et al. suggest that "The initial curve of an active contour based on any type of GVF energy should be included in the region to be segmented. It should also include all the centers of strong divergence of the region and all the centers of weak divergence connecting them."

The initialization can be done by manually or automatically selecting a point $p = p(i,j)$ in the desired region containing the strong and weak centers of divergence. The inverse S-GVF is used to reach the closest Cd_{strong} of the region:

$$while\left(p \notin Cd_{strong}\right) p \leftarrow (p - v(p)). \tag{4.7}$$

Once a point $p_s \in Cd_{strong}$ is found, all Cd_{strong} connected to it via Cd_{weak} points are selected. A morphological operator is used to dilate the connected path and extract its boundary. This boundary is used as the initial curve for the snake. As a consequence, it is ensured that the initial curve will be attracted toward different directions [3].

4.3.3 Automatic Initialization Approach to Segment CT Scans of Lungs

Condurache et al. proposed a new method for initialization, but their method strictly pertains to finding the boundaries of lungs [4]. For lung segmentation, it was proposed in [4] to use a closed snake which is automatically initialized using the bones observable in the analyzed images. For a robust automatic segmentation, the snake evolves in two stages, each using different external forces. For the first stage, the external force is computed after the image is morphologically processed so that only bony structures remain. Thus the possibility of the snake hanging onto the atelectatic lung instead of the true lung boundaries is avoided. Then, for a precise segmentation, during the second stage, the external force is computed using the original image.

To robustly and automatically initialize the active contour model, the ribs which are visible in all images are used. Thus, the images are segmented first to find their extended convex hulls.

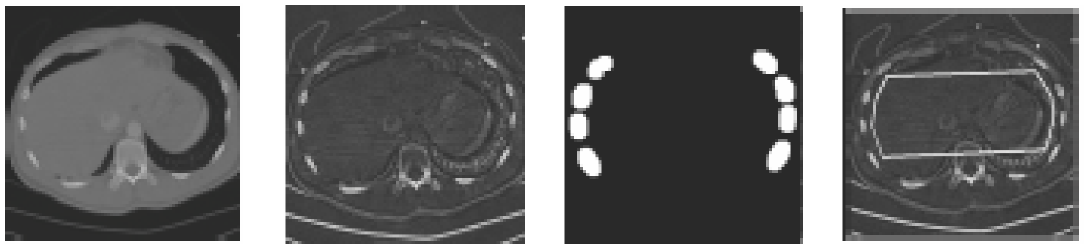

(a) (b) (c) (d)

Figure 4.5. Automatic initialization approach for segmenting lungs: (a) original image (lungs), (b) tophat result, (c) extended convex hull, (d) initialization points (source: [4])

The points on the convex hull border closest to the image center are used to define the initialization curve, as shown in Figure 4.5. To ensure a proper initialization of the snake, the segmentation result is processed to obtain the extended convex hull of each detected rib. For this purpose, the distance between each segmented pixel and the image center is computed first. The segmented pixels with the distances from the image center below a certain threshold are eliminated from the segmentation since the ribs are characterized as the pixels with larger distances from the image center. Then the image is dilated, so the objects are connected to obtain the extended rib convex hull. The points on the convex hull closest to the image center are used to initialize the active contour. Starting from the initial contour, the snake evolves from the tophat result to avoid stopping on the edges of the atelectatic lung tissue. This method, as mentioned earlier, works for segmenting lung boundaries only.

In chapter 5, we will introduce a novel scheme for automatically initializing the active contour.

Chapter 5: Novel Automatic Active Contouring Algorithm

As previously discussed, it is important that the initial contour should be as close to the true boundary as possible. If not, the active contour is likely to converge to the undesired result [2]. In this chapter, we present a new algorithm that overcomes some difficulties experienced by the schemes in Section 4.3.

5.1 Description of Novel Automatic Initial Contour Algorithm

5.1.1 Connected Object Extraction

One of the most important processes in computer vision is to quantify the huge amount of object information which we have to recognize by preserving the essential points only [29]. Therefore, as a first step, *eight segmented analysis* (details stated in the Appendix) is to be implemented to extract the regions of interest. By performing this analysis, we are able to avoid performing the contour analysis to the object of interest and thus save the unnecessary computation.

5.1.2 Boundary Anchoring and Confinement

Now, we have to identify the four extremities (north point, south point, east point and west point) along the boundary of the segmented object from the previous step. Once these four anchoring points are set, all subsequent procedures are carried out in the region confined by these points to reduce the computational time. Assume that the set ϑ contains the coordinates (x, y) of all the pixels belonging to the object. The collection of all *x*-values in ϑ can form a set ϑ_x, while the collection of all *y*-values in ϑ can form another set ϑ_y. Then, we can find

$$x_{\inf} = \inf\{\vartheta_x\}, \tag{5.1.a}$$

$$x_{\sup} = \sup\{\vartheta_x\}, \tag{5.1.b}$$

$$y_{\inf} = \inf\{\vartheta_y\}, \tag{5.1.c}$$

$$y_{\sup} = \sup\{\vartheta_y\}. \tag{5.1.d}$$

where $\inf\{\ \}$ and $\sup\{\ \}$ are the inferiorem and supremum of the subject set, respectively. Then we can determine the corresponding *x*-/*y*-values for those values given by Eq. (5.1), such that

$$y_{west} \equiv \inf\{\vartheta_{y,west}\}, \vartheta_{y,west} \equiv \{y \mid y \in \vartheta_y \cap (x_{\inf}, y) \in \vartheta\}, \tag{5.2.a}$$

$$y_{east} \equiv \sup\{\vartheta_{y,east}\}, \vartheta_{y,east} \equiv \{y \mid y \in \vartheta_y \cap (x_{\sup}, y) \in \vartheta\}, \tag{5.2.b}$$

$$x_{north} \equiv \sup\{\vartheta_{x,north}\}, \vartheta_{x,north} \equiv \{x \mid x \in \vartheta_x \cap (x, y_{\sup}) \in \vartheta\}, \tag{5.2.c}$$

$$x_{south} \equiv \inf\{\vartheta_{x,south}\}, \vartheta_{x,south} \equiv \{x \mid x \in \vartheta_x \cap (x, y_{\inf}) \in \vartheta\}. \tag{5.2.d}$$

According to Eqs. (5.1) and (5.2), we can construct the four extremities as

$$\vec{x}_N = (x_{north}, y_{\sup}), \tag{5.3.a}$$

$$\vec{x}_S = (x_{south}, y_{\inf}), \tag{5.3.b}$$

$$\vec{x}_E = (x_{\sup}, y_{east}), \tag{5.3.c}$$

$$\vec{x}_W = (x_{\inf}, y_{west}). \tag{5.3.d}$$

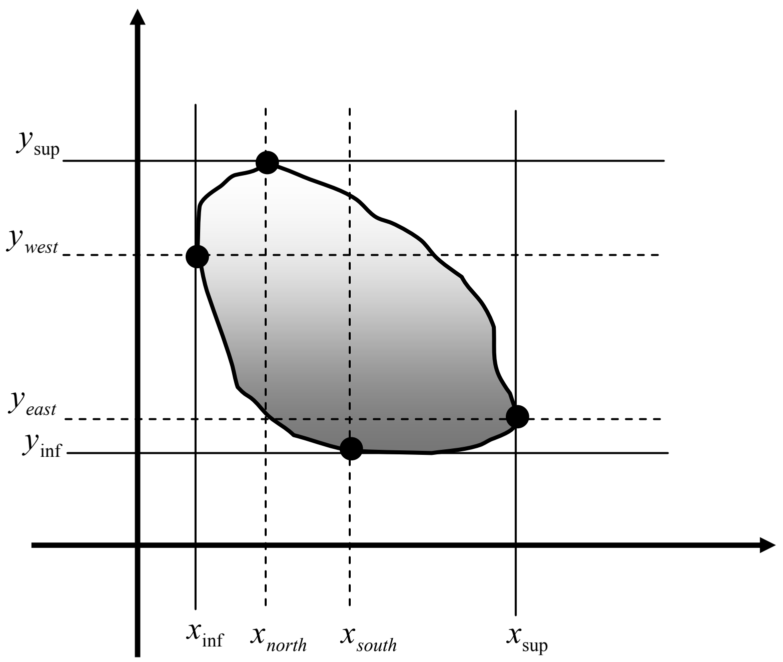


Figure 5.1. Boundary Anchoring

5.1.3 Initializing the Total Number of Search Steps

In order to determine the initial contour, we need to run a search procedure for obtaining an appropriate quadrilateral. Hence, we need to define the total numbers of search steps, σ_h (for scanning along the horizontal direction) and σ_v (for scanning along the vertical direction), for this search procedure. The total numbers of search steps are initially set as

$$\sigma_h = \sigma_v = 2 . \tag{5.4}$$

These total numbers of search steps can be used later for determining the actual search step sizes horizontally and vertically.

5.1.4 Setting Interception Threshold

In order to assure that the initial contour lies within the actual object, we have to detect if an edge intercepts with any object point. Hence, here we define λ_{total} as the total number of interceptions such that

$$\lambda_{total} \equiv \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 , \tag{5.5}$$

where λ_1, λ_2, λ_3, λ_4 are the respective numbers of interceptions with the object pixels between each pair among the four points constituting the quadrilateral. We iteratively try to find the appropriate initialization points constituting a quadrilateral until the total number of interceptions λ_{total} is lesser than the threshold value λ_{th} which are the maximum number of interceptions allowed along the quadrilateral periphery, i.e., $\lambda_{total} \le \lambda_{th}$. A typical example of interceptions is illustrated in Figure 5.2.

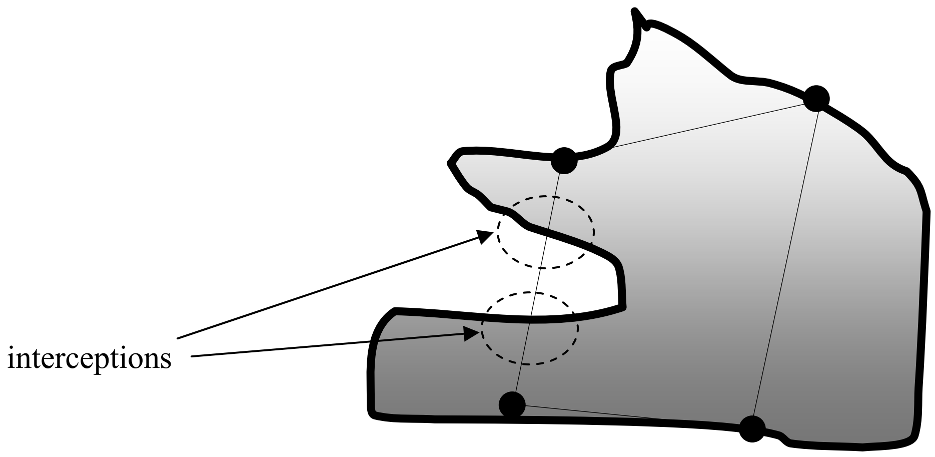


Figure 5.2. Initialization points constituting a quadrilateral.

5.1.5 Determining the Step-size for Scanning in Each Direction

The step size for scanning in the horizontal direction (from $\vec{x}_W$ to $\vec{x}_E$) in a initialization search procedure is given by δ_h. That is

$$\delta_h \equiv \frac{x_{\sup} - x_{\inf}}{\sigma_h}, \tag{5.6}$$

where σ_h is the total number of search steps defined in Section 5.1.3. Similarly, δ_v is defined as the step-size for scanning in the vertical direction (from $\vec{x}_N$ to $\vec{x}_S$) such that

$$\delta_v \equiv \frac{y_{\inf} - y_{\sup}}{\sigma_v}, \tag{5.7}$$

where σ_v is the total number of steps also defined in Section 5.1.3.

5.1.6 Creating two Sets for the Initialization Search

In search for the initialization points, we have to create two sets S_{hor} and S_{ver} which are defined as

$$S_{hor} \equiv \{(x,y) \mid x = x_{\inf} + m\delta_h, m = 0,1,\cdots,\sigma_h;\ y = y_{\inf}, y_{\inf}+1,\cdots,y_{\sup}\}, \tag{5.8}$$

and

$$S_{ver} \equiv \{(x,y) \mid x = x_{\inf}, x_{\inf}+1,\cdots,x_{\sup};\ y = y_{\inf} + n\delta_v, n = 0,1,\cdots,\sigma_v\}, \tag{5.9}$$

S_{hor} and S_{ver} are the sets for the horizontal and vertical initialization search procedures, respectively.

5.1.7 Search for the Set of Points Belonging to the Object

From the sets given by Eqs. (5.8) and (5.9), we may search for the object pixels and construct two series of sets $S^{y}_{hor,m}$, $m = 0,1,\cdots,\sigma_h$ and $S^{x}_{ver,n}$, $n = 0,1,\cdots,\sigma_v$ such that

$$S^{y}_{hor,m} \equiv \{y \mid x = x_{\inf} + m\delta_h; (x,y) \in \vartheta\},\ m = 0,1,\cdots,\sigma_h, \tag{5.10}$$

and

$$S^{x}_{ver,n} \equiv \{x \mid y = y_{\inf} + n\delta_v; (x,y) \in \vartheta\},\ n = 0,1,\cdots,\sigma_v, \tag{5.11}$$

where $S^{y}_{hor,m}$, $m = 0,1,\cdots,\sigma_h$ and $S^{x}_{ver,n}$, $n = 0,1,\cdots,\sigma_v$ all are totally ordered sets in ascending order. Then, we form the two series of sets $DS^{y}_{hor,m}$, $m = 0,1,\cdots,\sigma_h$ and $DS^{x}_{ver,n}$, $n = 0,1,\cdots,\sigma_v$ such that

$$DS^{y}_{hor,m} \equiv \{d_k \mid d_k = S^{y}_{hor,m}(k+1) - S^{y}_{hor,m}(k), k = 1,2,\cdots,\#\left(S^{y}_{hor,m}\right) - 1\}, \tag{5.12}$$

and

$$DS^{x}_{ver,n} \equiv \{c_k \mid c_k = S^{x}_{ver,n}(k+1) - S^{x}_{ver,n}(k), k = 1,2,\cdots,\#\left(S^{x}_{ver,n}\right) - 1\}, \tag{5.13}$$

where $S^{y}_{hor,m}(k)$ and $S^{x}_{ver,n}(k)$ are the k^{th} elements in $S^{y}_{hor,m}$ and $S^{x}_{ver,n}$, respectively; $\#\left(S^{y}_{hor,m}\right)$ and $\#\left(S^{x}_{ver,n}\right)$ denote the total numbers of elements in $S^{y}_{hor,m}$ and $S^{x}_{ver,n}$, respectively.

Then we construct two series of two-element sets $P_{hor,m}$, $m = 1,2,\cdots,\sigma_h$ and $P_{ver,n}$, $n = 1,2,\cdots,\sigma_v$ such that

$$P_{hor,m} \equiv \{(x_{\inf} + m\delta_h, S^{y}_{hor,m}(k_1')),\ (x_{\inf} + m\delta_h, S^{y}_{hor,m}(k_1' + 1))\}, \tag{5.14}$$

and

$$P_{ver,n} \equiv \{(S^{x}_{ver,n}(k_2'), y_{\inf} + n\delta_v),\ (S^{x}_{ver,n}(k_2' + 1), y_{\inf} + n\delta_v)\}, \tag{5.15}$$

where $k_1' = \arg\max_k \left(DS^{y}_{hor,m}(k)\right)$ and $k_2' = \arg\max_k \left(DS^{x}_{ver,n}(k)\right)$; $DS^{y}_{hor,m}(k)$ and $DS^{x}_{ver,n}(k)$ denote the k^{th} elements in $DS^{y}_{hor,m}$ and $DS^{x}_{ver,n}$, respectively.

5.1.8 Selecting Vertices for the Quadrilateral as the Initial Contour

Once we obtain the two series sets as described in Eqs. (5.14) and (5.15), we can find the vertices for the initialization quadrilateral. First, let's rewrite $P_{hor,m}$, $m = 0,1,\cdots,\sigma_h$ and $P_{ver,n}$, $n = 0,1,\cdots,\sigma_v$ as

$$P_{hor,m} \equiv \{(x_{\inf} + m\delta_h, y^{1}_{hor,m}),\ (x_{\inf} + m\delta_h, y^{2}_{hor,m})\}, \tag{5.16}$$

and

$$P_{ver,n} \equiv \{(x^{1}_{ver,n}, y_{\inf} + n\delta_v),\ (x^{2}_{ver,n}, y_{\inf} + n\delta_v)\}. \tag{5.17}$$

Then we create a new set containing all the *y*-values on the top coordinate in every set defined by Eq. (5.16) such that

$$P_{north} \equiv \left\{ y^2_{hor,m} \mid m = 1,2,\cdots,\sigma_h ; y^2_{hor,m} - y^1_{hor,m} > d_{th} \right\}. \qquad (5.18)$$

where d_{th} is the minimum distance required between the top and bottom points in $P_{hor,m}$. Similarly, we create another set containing all the *y*-values on the bottom coordinate in every set defined by Eq. (5.16) such that

$$P_{south} \equiv \left\{ y^1_{hor,m} \mid m = 1,2,\cdots,\sigma_h ; y^2_{hor,m} - y^1_{hor,m} > d_{th} \right\}. \qquad (5.19)$$

In the same way, we may create two more sets containing all the *x*-values on the left and right coordinates in each set defined by Eq. (5.17) as

$$P_{west} \equiv \left\{ x^1_{ver,n} \mid n = 1,2,\cdots,\sigma_v ; x^2_{ver,n} - x^1_{ver,n} > c_{th} \right\}. \qquad (5.20)$$

and

$$P_{east} \equiv \left\{ x^2_{ver,n} \mid n = 1,2,\cdots,\sigma_v ; x^2_{ver,n} - x^1_{ver,n} > c_{th} \right\}. \qquad (5.21)$$

where d_{th} is the minimum distance required between the left and right points in $P_{ver,n}$. Thus, we can achieve the four vertices of the quadrilateral for the initial contour as

$$\vec{v}_1 \equiv \left(x_{\inf} + m'\delta_h, y_{north} \right), \text{ where } y_{north} = \min\left(P_{north}\right) \text{ and } \left(x_{\inf} + m'\delta_h, y_{north} \right) \in \vartheta, \qquad (5.22.a)$$

$$\vec{v}_2 \equiv \left(x_{\inf} + m''\delta_h, y_{south} \right), \text{ where } y_{south} = \max\left(P_{south}\right) \text{ and } \left(x_{\inf} + m''\delta_h, y_{south} \right) \in \vartheta, \qquad (5.22.b)$$

$$\vec{v}_3 \equiv \left(x_{west}, y_{\inf} + n'\delta_v \right), \text{ where } x_{west} = \max\left(P_{west}\right) \text{ and } \left(x_{west}, y_{\inf} + n'\delta_v \right) \in \vartheta, \qquad (5.22.c)$$

$$\vec{v}_4 \equiv \left(x_{east}, y_{\inf} + n''\delta_v \right), \text{ where } x_{east} = \min\left(P_{east}\right) \text{ and } \left(x_{west}, y_{\inf} + n''\delta_v \right) \in \vartheta, \qquad (5.22.d)$$

5.1.9 Connecting the Vertices to Form a Proper Quadrilateral

Since the four vertices $\vec{v}_1$, $\vec{v}_2$, $\vec{v}_3$ and $\vec{v}_4$ are in arbitrary constellation, we have to develop a sophisticated scheme to connect them properly, i.e., no two edges can intercept with each other in the resulting quadrilateral. Figure 5.3 illustrates the proper and improper connections for a set of four vertices.

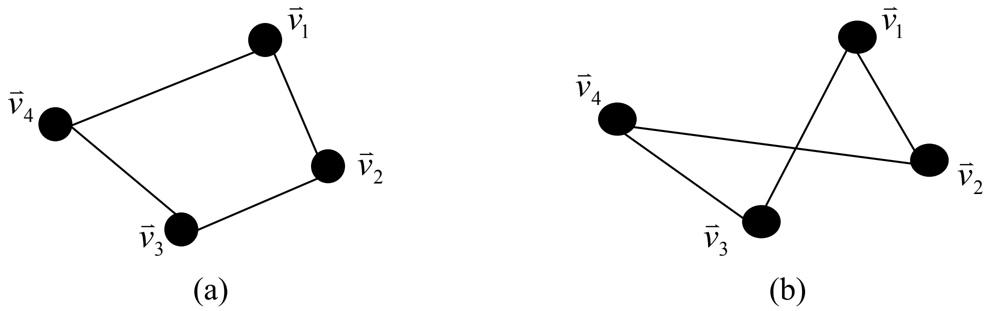


Figure 5.3. Connection among the vertices: (a) resulting in a proper contour (a regular quadrilateral), (b) resulting in an improper contour.

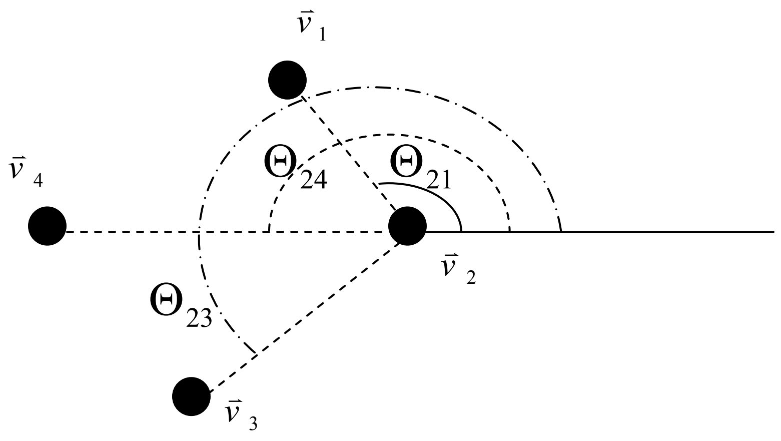


Figure 5.4. Relative angles Θ_{2n}, $n = 1,3,4$ for a set of four vertices.

To establish a proper connection, we need to compare the relative angles with respect to a reference vertex $\vec{v}_m$ (say $\vec{v}_2$ as depicted in Figure 5.4). We compute the angles Θ_{mn} at which $\vec{v}_n$, $n = 1,2,3,4$, $n \neq m$, are inclined to $\vec{v}_m$ such that

$$\Theta_{mn} = \cos^{-1}\left(\frac{\vec{v}_m{}^T(\vec{v}_n - \vec{v}_m)}{\|\vec{v}_m\|\,\|\vec{v}_n - \vec{v}_m\|}\right),\ n = 1,2,3,4,\ n \neq m. \tag{5.23}$$

For each m=1, 2, 3, 4, we can obtain three Θ_{mn} values and connect $\vec{v}_m$ to $\vec{v}_n$ with the two smallest Θ_{mn}. Consequently, we can form a proper contour as a regular quadrilateral.

5.1.10 Iterative Initialization Procedure to Avoid Interceptions with the Object

According to Eq. (5.5), we can compute the numbers of interceptions along the four edges of the quadrilateral with the object, namely λ_1, λ_2, λ_3, λ_4. Then we can check if $\lambda_{total} \le \lambda_{th}$. If so, we can use the resulting quadrilateral as the initial contour for snake to evolve. If $\lambda_{total} > \lambda_{th}$, we find those $\lambda_i > 0$, i =1, 2, 3, 4. Then, we need to increment σ_h or σ_v or both and repeat the steps described in Sections 5.1.5-5.1.10 to draw a new quadrilateral for reducing λ_{total}. For example, assume that $\lambda_1 > 0$ corresponds to the number of interceptions with the object for the edge between $\vec{v}_1$ and $\vec{v}_2$ as given by Eq. (5.22). According to Eq. (5.22), we may increase σ_h to redo the initialization for a new quadrilateral and reduce λ_{total}.

The procedures in Sections 5.1.5 to Section 5.1.10 can be repeated again and again until $\lambda_{total} \leq \lambda_{th}$. Thus, the initialization points resulting from our new scheme can be used as an effective initial contour. The snake can therefore start from this initial contour to deform according to Euler equations as discussed in Chapter 3.

5.2 Algorithm Flowchart

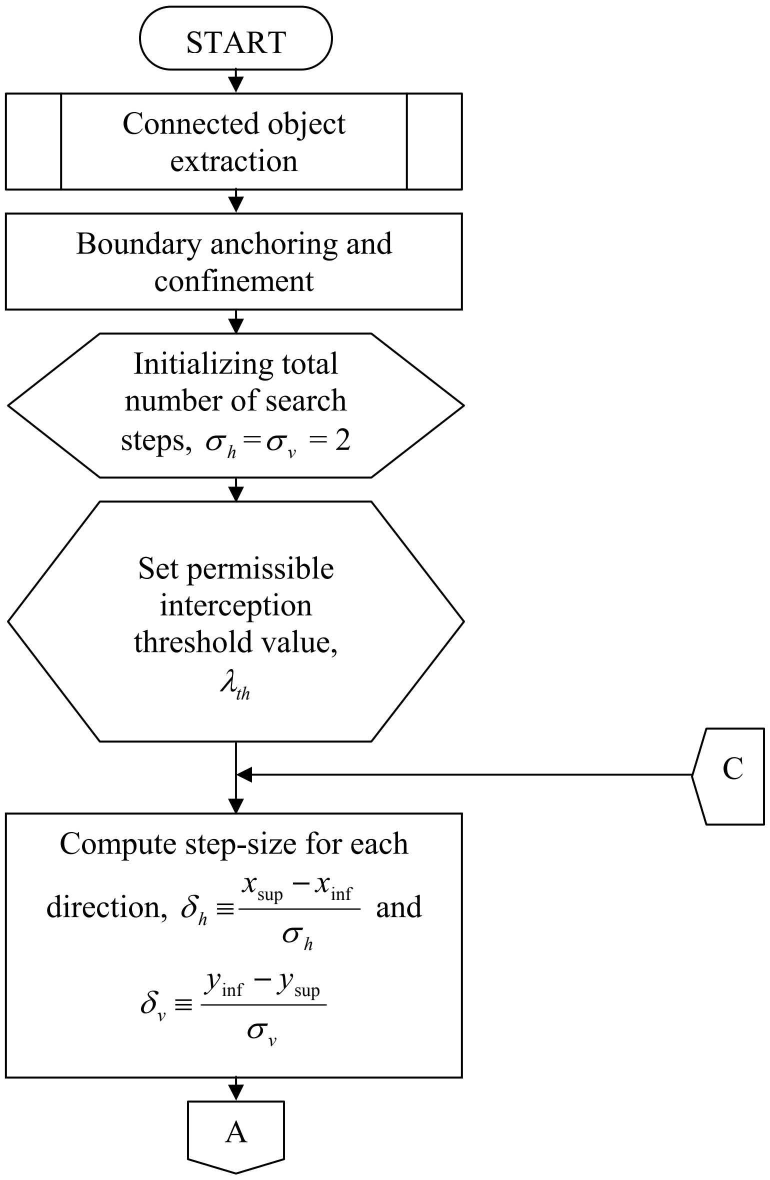
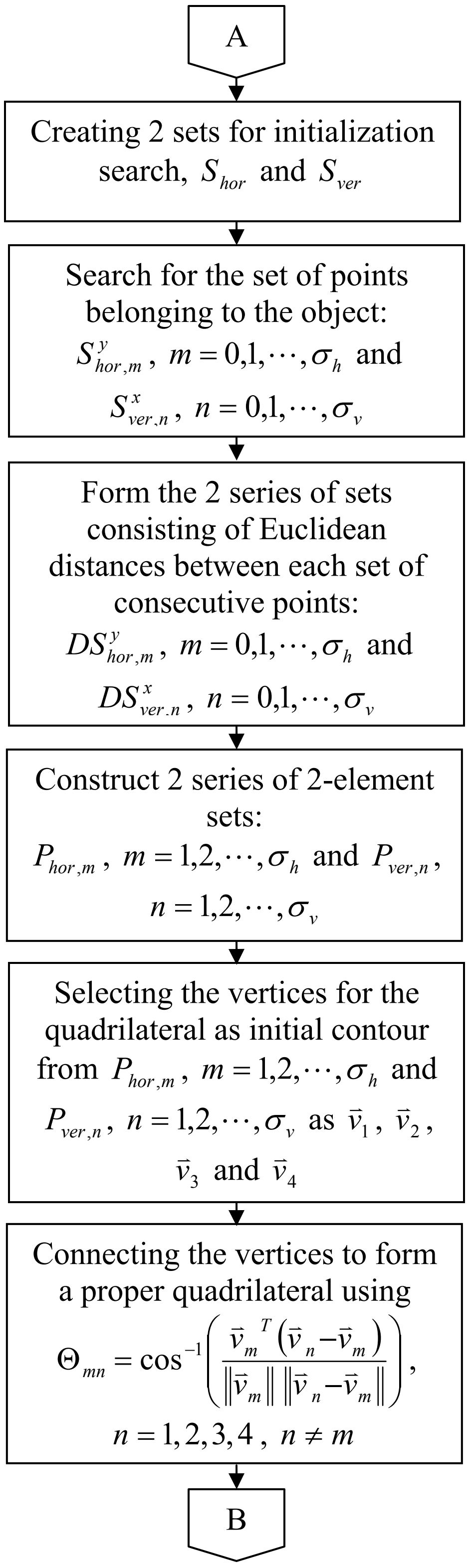
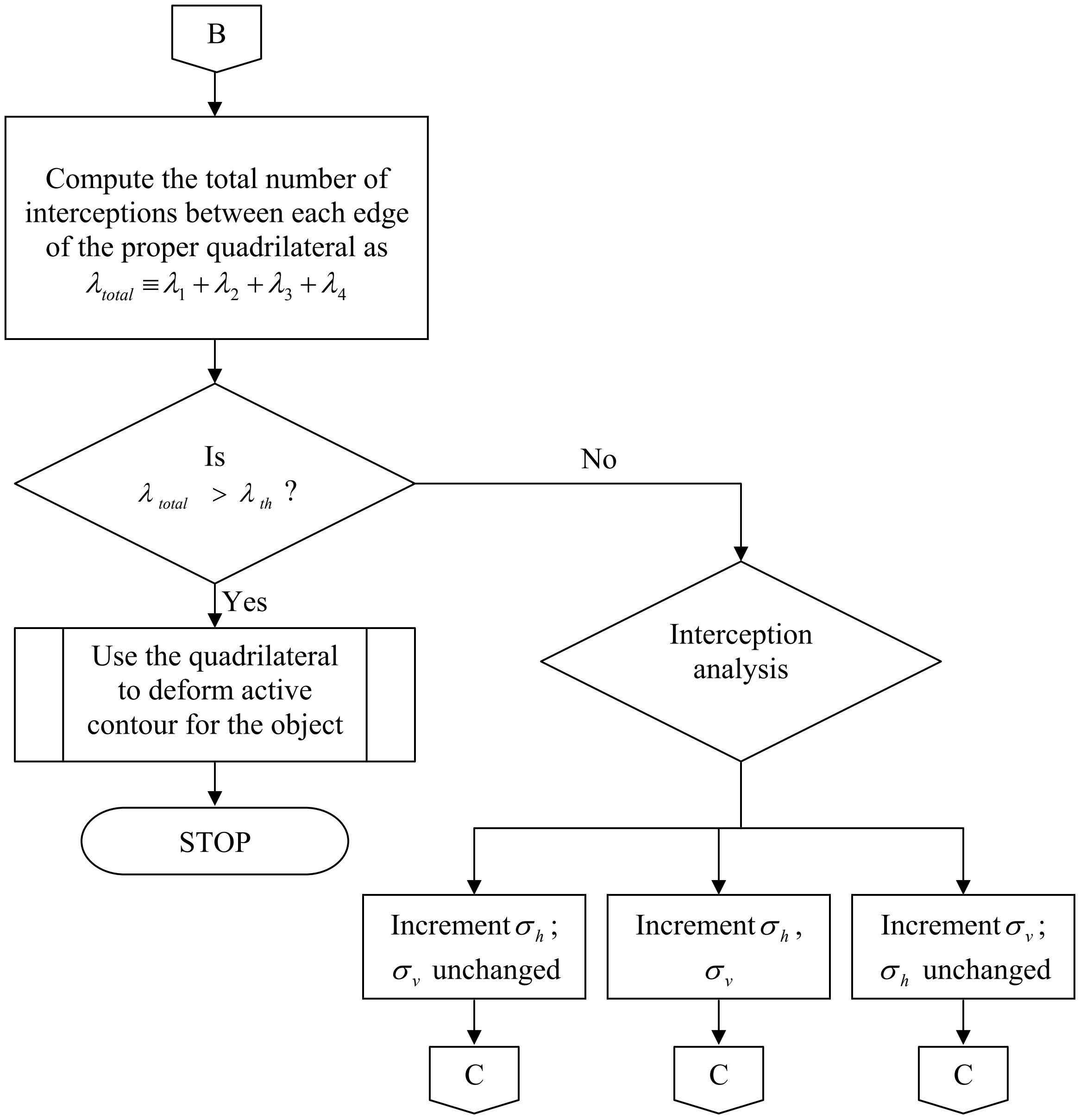

START
Connected object extraction
Boundary anchoring and confinement
Initializing total number of search steps, $\sigma_h = \sigma_v = 2$
Set permissible interception threshold value, λ_{th}
C
Compute step-size for each direction, $\delta_h \equiv \frac{x_{\sup} - x_{\inf}}{\sigma_h}$ and $\delta_v \equiv \frac{y_{\inf} - y_{\sup}}{\sigma_v}$
A

Figure 5.5. The flowchart for our novel automatic initialization algorithm.

(fig. 5.5 contd.)


A
Creating 2 sets for initialization search, S_{hor} and S_{ver}
Search for the set of points belonging to the object: $S^{y}_{hor,m}$, $m = 0,1,\cdots,\sigma_h$ and $S^{x}_{ver,n}$, $n = 0,1,\cdots,\sigma_v$
Form the 2 series of sets consisting of Euclidean distances between each set of consecutive points: $DS^{y}_{hor,m}$, $m = 0,1,\cdots,\sigma_h$ and $DS^{x}_{ver,n}$, $n = 0,1,\cdots,\sigma_v$
Construct 2 series of 2-element sets: $P_{hor,m}$, $m = 1,2,\cdots,\sigma_h$ and $P_{ver,n}$, $n = 1,2,\cdots,\sigma_v$
Selecting the vertices for the quadrilateral as initial contour from $P_{hor,m}$, $m = 1,2,\cdots,\sigma_h$ and $P_{ver,n}$, $n = 1,2,\cdots,\sigma_v$ as $\vec{v}_1$, $\vec{v}_2$, $\vec{v}_3$ and $\vec{v}_4$
Connecting the vertices to form a proper quadrilateral using $\Theta_{mn} = \cos^{-1}\left(\frac{\vec{v}_m^{\,T}(\vec{v}_n - \vec{v}_m)}{\|\vec{v}_m\|\,\|\vec{v}_n - \vec{v}_m\|}\right)$, $n = 1,2,3,4$, $n \neq m$
B

(fig. 5.5 contd.)

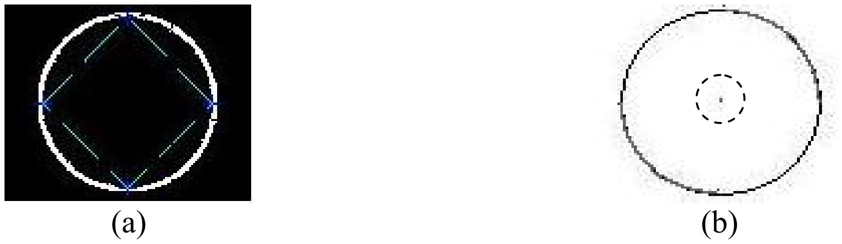

B
Compute the total number of interceptions between each edge of the proper quadrilateral as $\lambda_{total} \equiv \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4$
Is $\lambda_{total} > \lambda_{th}$?
No
Yes
Use the quadrilateral to deform active contour for the object
STOP
Interception analysis
Increment σ_h ; σ_v unchanged
Increment σ_h , σ_v
Increment σ_v ; σ_h unchanged
C
C
C

5.3 Simulations

Since we would like to explore the applicability of our new algorithm for the ultrasonic nondestructive evaluation of material samples, we collected some C-scan ultrasonic images through the equipment shown in Figure 5.6, which is facilitated at Mechanical Engineering Department, Southern University. The scanning frequencies of the collected image samples ranged from 5 to 20 MHz. The tested material samples were the metal pieces with some defects.

Figure 5.6. Ultrasonic imaging equipment for nondestructive evaluation of materials.

To test our algorithm, we first consider a simple object, namely a circle. We compare two automatic initialization schemes here, namely our new scheme described in Section 5.1 and the center-of-divergence method in [2]. According to Figure 5.7, it is obvious that our scheme results in an initial contour lying within the object boundary, so does the center-of-divergence algorithm. However, it needs the sophisticated effort to determine the appropriate size of the initial contour (the diameter of the dashed circle around the center as depicted in Figure 5.7 (b)) to assure the initial contour lying within the object boundary and also as close to the boundary as possible. According to Figure 5.7, it can be observed that our new algorithm can lead to a much bigger initial contour than the center-of-divergence method. It means that although we have a relatively computationally heavy initialization algorithm, we can have a much faster snake convergence in reward, compared to other existing initialization methods.



(a) (b)

Figure 5.7. Comparison of results between: (a) our new initialization algorithm and (b) the center-of-divergence method (solid line: circle object, dashed line: initial contour).

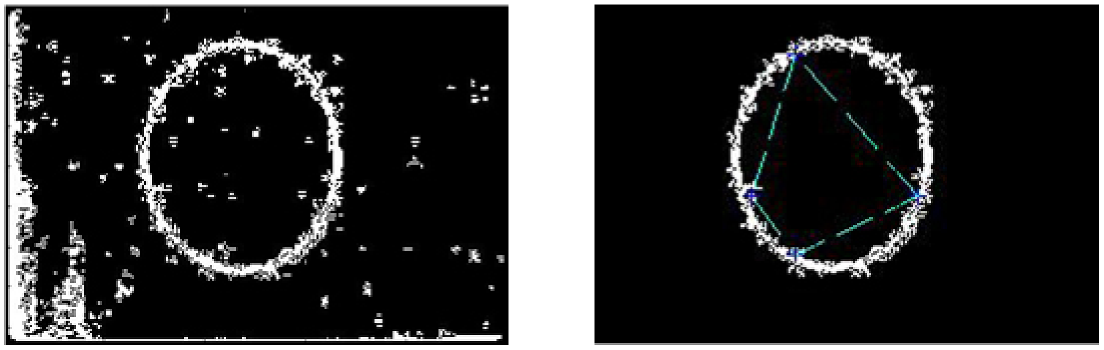

(a) (b)

Figure 5.8. Initial contour for a noisy image: (a) original image, (b) initial contour (blue dashed quadrilateral) drawn by our algorithm.

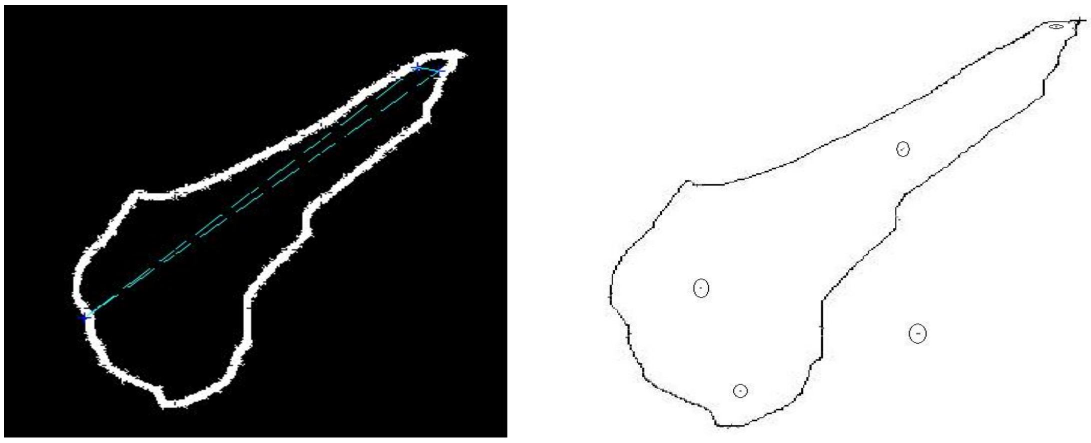

(a) (b)

Figure 5.9. Comparison of initial contours: (a) a single initial contour resulting from our new algorithm (blue dashed quadrilateral), (b) multiple initial contours resulting from the center-of-divergence algorithm [2].

To test the robustness of our new initialization algorithm, we tested a noisy object in Figure 5.8 (a). The initial contour achieved by our algorithm is depicted in Figure 5.8 (b). The number of iterations for this initial contour is 5. It shows that the initial contour resulting from our algorithm is very promising even for noisy objects. Another advantage of our new algorithm over the existing center-of-divergence method can be illustrated in Figure 5.9, where (a) denotes the single initial contour achieved by our new algorithm and (b) denotes multiple initial contours resulting from the center-of-divergence algorithm. In our method, we used the interception threshold $\lambda_{th} = 13$ and the distance threshold $d_{th} = c_{th} =$ 5. The initial contour resulted from 36 iterations. Note that an arbitrary object as which we hand-drew in Figure 5.9 will often lead to the multiple initial contours and the spurious centers-of-divergence that would induce a heavy computation burden and demand a sophisticated spurious contour elimination procedure ultimately but our algorithm can always determine a single initial contour.

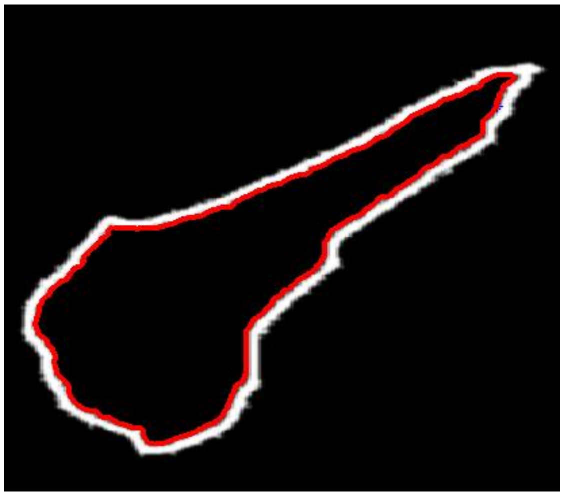


Figure 5.10. Final contour resulting from the snake algorithm using our new initialization scheme.

Once the initial contour is obtained in 5.9 (a), the snake deforms according to the Euler equations with tension constraint $\alpha = 0.5$ and bending constraint $\beta = 0.05$. Figure 5.10 shows the ultimate contour after the snake takes 50 iterations to deform. The total computational time including the automatic initialization contour is 103.78 seconds. Hence the complete snake algorithm with our new initialization is relatively computationally efficient.

Chapter 6: Conclusion

In this thesis, we tackle the automatic initialization problem for the active contour model. We design a novel robust automatic initialization algorithm for the evolution of the snake algorithm. According to numerous experiments, our scheme outperforms other existing algorithms such as center-of-divergence method, center-of-strong/weak-divergence method, and automatic initialization for computer-tomography scans of lungs. In our algorithm, we search for the four vertices which are located at the object boundary and then form a regular quadrilateral. Moreover, we have an iterative procedure to assure the resulting quadrilateral periphery does not intercept with the object boundary. The resulting quadrilateral as an initial contour for snake deformation, although perhaps not optimal, is relatively much bigger than the initial contours generated by other existing automatic initialization schemes. On the other hand, our new automatic initialization algorithm is relatively computationally heavy, but it often leads to a much faster snake convergence in reward, compared to other existing initialization methods. Our new method makes possible the completely automatic contouring techniques in the future. Such techniques can avoid human errors or experts' manipulation in computer graphics applications.

References

1. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour models. International Journal for computer vision, vol 1, pp. 321 – 332, 1988.

2. Ge Xingfei, Tian Jie, An automatic active contour model for multiple objects, 2002 IEEE pp. 881 – 884.

3. Clovis Tauber, Hadj Batatia and Alain Ayache, "A general Quasi-Automatic Initialization for Snakes: Application to Ultra-sound images", IEEE International Conference on Image Processing, 2005, vol. 2, 11-14 sept. 2005, pp.: 806-9.

4. Alexandru P. Condurache, Edward A. Essah, Andreas Reske, Matthias Seiwerts, Harald Busse, Til Acah, Ulrich G. Hoffmann, "Robust Rib Cage Segmentation in CT image series using Active Contour Models"

5. www.ndt.org

6. "Non-destructive Testing", Louis Cartz Marquette University. College of Engineering, Milwaukee, WI, USA. ASM international.

7. www.ndt-ed.org/GeneralResources/MethodSummary/MethodSummary.htm

8. R. C. McMaster, *Nondestructive Testing Handbook: Liquid Penetrant Tests*, American Society for Nondestructive Testing, 1982.

9. M.G. Silk, Ultrasonic Transducers for NDT, Adam Hilger, 1984.

10. H G Tattersall, "The ultrasonic pulse-echo technique as applied to adhesion testing," J. Phys. D: Appl. Phys., vol. 6, pp.819-832, 1973.

11. J. Krautkramer, H. Krautkramer., Ultrasonic Testing of Materials, 2nd edition, Springer, Berlin, Heidelberg, and New York, 1977.

12. Jiang Jian, Zhou Xiao-jun, Guo Tian-tai, Wu Si-yuan, "Research on ultrasonic detection of complex surfaces", Journal of Zhejiang University SCIENCE, ISSN 1009 – 3095. http://www.zju.edu.cn/jzus

13. M. Berke, "Non destructive material testing with ultrasonics – introduction to the basic principles", NDT.net, vol. 5, no. 9, September 2000, http://www.ndt.net

14. D. Pagodinas, "Ultrasonic signal processing methods for detection of defects in composite materials", NDT.net, vol. 8, no. 7, July 2003, http://www.ndt.net

15. W. Hilger, "Ultrasonic imaging of internal defects in composites", NDT.net, vol. 2, no. 5, May 1997, http://www.ndt.net

16. P. Kalyanasundaram, C. Rajagopalan, C.V. Subramanian, M. Thavasimuthu and B. Raj, “Ultrasonic signal analysis for defect characterization in composite materials”, British Journal of NDT, vol. 33, no. 5, pp. 221 – 226, 1991.

17. Ming Yu, Ying-Chun Guo and Zhi-Tao Xiao. A novel edge detection method and its application in ultrasound images. Proceedings of the third international conference on Machine Learning and Cybernetics

18. C. Garbay. Image structure representation and processing. A discussion of some segmentation methods in cytology. IEEE transactions on Pattern Analysis and Machine Intelligence, PAMI – 8(2): 140 – 146, March 1986.

19. Tracking and Describing deformable objects using Active Contour Models. Dissertation by Frederic F. Leymarie, Feb 1990

20. V. Caselles, F. Catte, T. Coll, and F. Dibos. A geometric model for active contours. Numerische Mathematik, 66: 1 – 31, 1993.

21. L. D. Cohen. On active contour models and balloons. CVGIP: Image Understanding, 56(2): 242 – 263, September 1992.

22. B. Leroy, I. Herlin, L.D. Cohen, “Multi-resolution algorithms for active contour models”, in the 12th International conference Analysis and Optimization systems, 1996, pp. 58 – 65.

23. L.D. Cohen and I. Cohen, “Finite-Element methods for active contour models and balloons for 2-D and 3-D images”, IEEE transactions on Pattern Analysis Machine Intelligence, vol. 15, pp. 1131 – 1147, Nov. 1993.

24. C. Xu and J.L. Prince, “Gradient Vector Flow: A new external force for snakes”, IEEE Proc. Conf. on Computer Vision and Pattern Recognition (CVPR), 1997, pp. 66 - 71.

25. R. Courant and D. Hilbert. Methods of Mathematical Physics, volume I, Interscience, New York, 1953.

26. Chenyang Xu and Jerry L. Prince, “Snakes, Shapes, and Gradient Vector Flow”, IEEE Transactions on Image processing, vol. 7, no. 3, March 1998, pp. 359 – 369.

27. H.S. Seshadri and A. Kandaswamy, “Application of Contour Models for the detection of cancer tumors in breast tissue”, The Internet Journal of Medical Simulation

28. C. Xu and J.L. Prince, “Generalized gradient vector flow external forces for active contours”, Signal Processing 71 (1998) pp. 131 – 139.

29. M. Tridi, N. Nacereddine and N. Oucief, “Contour Estimation in Synthetic and Real Weld Defect Images based on maximum-likelihood”, Transactions on Engineering, Computing and Technology, vol. 9, November 2005.

30. R.C. Gonzales, R.E. Woods, Digital Image Processing (second edition), Pearson Education.

31. E. Gose, R. Johnsonbaugh, S. Jost, Pattern Recognition and image analysis, 1997.

Appendix: Chain Codes

Often, one wants to divide or segment an image into a number of regions, each of which is relatively uniform in some characteristic. A region can be defined as a collection of adjacent pixels that are similar in some way, such as brightness, color, or local visual texture. Chain codes are widely used to represent a boundary formed by a connected sequence of straight-line segments with specified length and direction [30]. Typically, this representation is based on a 4- or 8-connectivity of the segments. The direction of each segment is coded using a numbering scheme. Digital images are usually acquired and processed in a grid format with equal spacing in the *x*- and *y*-directions, so a chain code could be generated by following a boundary, say, along a clockwise direction and assigning a direction to the segments connecting every pair of pixels.

The chain code of a boundary depends on the starting point. However, the code can be normalized with respect to the starting point by a straightforward procedure where the chain code is treated as a circular sequence of direction numbers. We may redefine the starting point so that the resulting number sequence forms an integer of minimum magnitude. Two regions that overlap only at a corner can be considered as either a single region or two distinct regions on the choice between 4-connectivity or 8-connectivity.

Each pixel has four neighbors that border it along a side: the ones above, below, to the right, and to the left. A pixel has eight neighbors if we also count the four pixels that touch it only at one of its four corners. Two pixels are said to be 4-adjacent if they share a side or 8-adjacent if they share either a side or a corner as shown in figure APP.1.1. A region is said to be 4-connected if for every pair of pixels in the region, there is some finite chain of pixels that connects them such that each consecutive pair of pixels is 4-adjacent. A region is said to be 8-connected if for every pair of pixels in the region, there is some finite chain of pixels that connects them such that each consecutive pair of pixels is 8-adjacent.

8	4	8
4	0	4
8	4	8

Figure APP. 1.1. Chain Codes: illustrates 4- and 8-connectivity where the pixel denoted by 0 is 4-connected to its 4-neighbors (marked as 4) and 8-connected to all of its neighbors (marked as 4 or 8).

Thresholding can divide an image into two or more regions, but we must still identify which pixels belong to each of these specific regions so that we can measure various properties for each individual region such as shape, size, location, or color. The labeling algorithm for performing such analysis is described as follows. The labeling algorithm uses a list to keep track of pixels that are yet to be labeled. This list (also called a queue in computer science) has two operations: insert (*s*,*t*) which inserts the pixel (*s*,*t*) at the end of the list, and (*s*,*t*) ← remove() which removes the pixel from the front of the list and saves it for further use.

The labeling algorithm begins by scanning the image left to right, top to bottom. When an unlabelled pixel (*x*,*y*) is found, the algorithm will label all the pixels in the 4-connected region to which (*x*,*y*) belongs before it labels any pixel from other regions. Say, we first dispatch a new label *L* and then label (*x*,*y*) as *L* and add (*x*,*y*) to an initially empty list of pixels whose neighbors are to be checked later. Next, we remove pixel (*s*,*t*) placed next in the list. We next label with *L* each unlabelled 4-neighbor of (*s*,*t*) that has the same gray level as (*s*,*t*) and insert each such 4-neighbor into the list. (Such 4-neighbors belong to the same region as (*s*,*t*)). We then repeat this process. If the list is not empty, we remove from the list the pixel (*s*,*t*) least recently placed in the list. Whenever the list is empty, one entire region has been labeled and the process terminates and we resume scanning the image from left to right and top to bottom, looking for another unlabelled pixel. If such a pixel is found, we dispatch a new label and restart the labeling process.

Pseudo-code for the region labeling algorithm [31]:

```
L ← -1 (initialize label)
Scan the image from left to right and top to bottom for all (x,y)
    If g(x,y) >= 1 then insert(x,y)
        While list is not empty do
```

```
            (s,t) ← remove()
            for each 8-neighbor (u,v) of (s,t) do
                if (u,v) is unlabelled and g(u,v) == g(x,y) then
                        insert (u,v)
                end if
            end for
        end while
        L ← L -1 (Get new label)
End scan
```

If an image has *n* pixels, the scanning part of the region-labeling algorithm takes *n* steps. Since each pixel goes into the list only once, the total number of times that the "while" loop is executed is *n*. Thus the "for" loop is executed at most for a total of 4*n* times. Therefore this algorithm terminates at most after *Cn* steps for some constant *C*. Figure APP. 1.2 shows an example of connected component analysis. Figure APP. 1.2 (a) shows the original image containing a number of cross-like symbols. Figure APP. 1.2 (b) shows an image where each component (a cross-like symbol) is labeled using a particular pixel value.

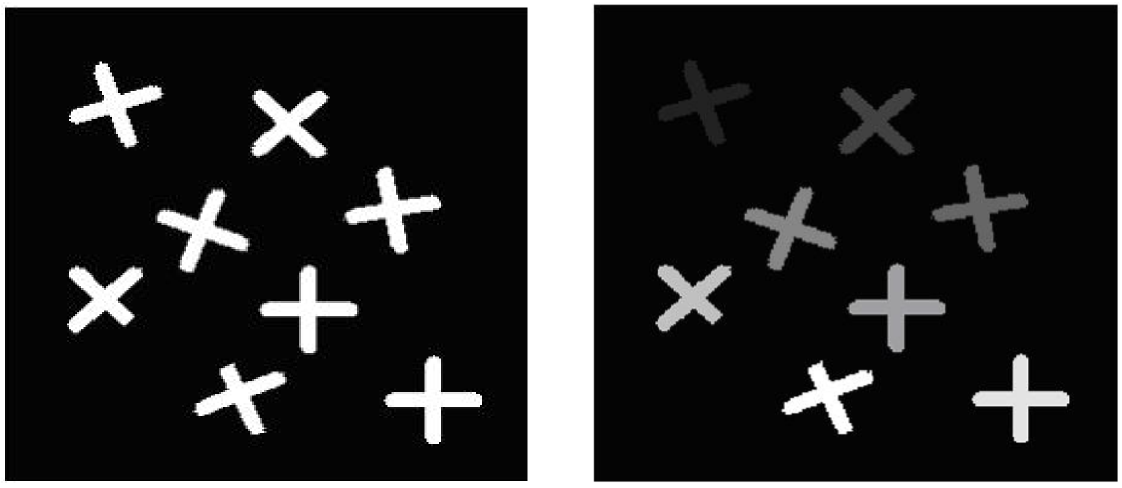
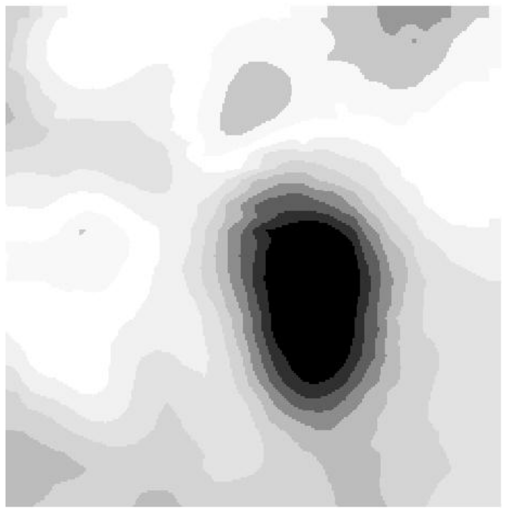

(a) (b)

Figure APP. 1.2. Chain Codes example: (a) original image, (b) image labeled into different pixel values after 8-connectivity analysis (shown in a gray color map).

In this thesis, instead of labeling each connected component with a separate pixel, we consider it as a region from which the active contour model algorithm can be established. We extract the regions from the original image as separate entities. For example, consider the image shown in Figure APP. 1.3 (a). It presents a C-scan image obtained from http://diwww.epfl.ch/w3lami/detec/detec2screen.html. The image components of interest as the separate objects are illustrated in Figure APP. 1.3 (b).


(a)

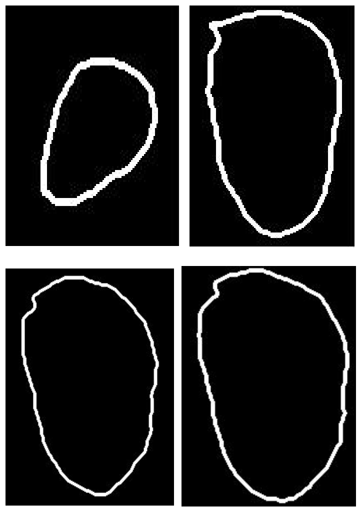

(b)

Figure APP. 1.3. Chain Code for an ultrasonic C-scan image: (a) original ultrasonic C-scan image, (b) image components of interest extracted from (a).

The labeling algorithm can be extended for three-dimensional images. In such a case, each voxel (volume pixel) has three types of neighbors: face neighbors, edge neighbors and corner neighbors. Therefore, at least three different definitions of connectivities exist. The employment of face neighbors only would tend to fragment regions, while the employment of all neighbors would tend to merge the regions together.

Vita

Srinath Vepathur Sitaraman was born on October 19, 1983, in Chennai, Tamil Nadu, India. He graduated from University of Madras, Chennai, India, with a degree of Bachelor of Engineering in Electronics and Instrumentation Engineering in May 2004. In August 2004, he came to Louisiana State University to pursue graduate studies in electrical engineering. He is currently a candidate for the degree of Master of Science in Electrical Engineering, which will be awarded in December 2006.