BONE FRACTURE DETECTION USING ACTIVE CONTOUR MODEL WITH PRIOR SHAPE

By Yun Jia

B.Eng., Chengdu, 1997

P.R. China,

A thesis

presented to Ryerson University

in partial fulfillment of the

requirement for the degree of

Master of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2006

Author's Declaration

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Author's signature: _____

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Author's signature: _____

Borrower's Page

Ryerson University requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Name	Signature	Address	Date

Abstract

Bone Fracture Detection

using Active Contour Model

with Prior Shape

© Yun Jia 2006

Master of Applied Science

Department of Electrical and Computer Engineering

Ryerson University

In this research, an image segmentation method based on active contouring model was studied, which incorporates the prior shape into the active contour evolving process as the global constraint. The active contour model is implemented based on the level set method. The prior shape regulates the behavior of the active contour and keeps it from leaking out of the weak edges. The goal of this research is to determine the displacement and alignment between two fractured pieces of a bone which is encased in the cast material by segmenting them out and calculating their axes difference. The noise introduced by the cast material makes this task difficult. Morphological operations of dilation and erosion are deployed in this research as the noise reduction and edge detection tool. Experiment results are obtained successfully by applying this method upon the X-ray images of patients' fractured arm.

Acknowledgements

Looking back upon the years at Ryerson University for my Master studies, my heart is filled with immense gratitude towards all the professors, researchers, physicians, friends and family who have supported and encouraged me against difficulties.

I would like to express my sincere thanks to Dr. Rachel Jiang, my supervisor at the Department of Electrical and Computer Engineering, Ryerson University, who supports me through my Master years and provides me the great opportunity to work on this research project under her supervision. I feel fortunate of having her guidance since the beginning of my studies.

Many sincere thanks also give to Professor Lin Guan, my co-supervisor at the Department of Electrical and Computer Engineering, Ryerson University, who gave me tremendous help, as well as inspirations and encouragements.

My thanks also give to my classmates and friends, for their friendship and supports through the valuable two years.

I would like to thank the National Science and Engineering Research Council (NSERC) of Canada for financial support to this research project. Thanks also give to Dr. Paul S. Babyn, the Radiologist-in-Chief at Hospital for Sick Children and an associate Professor at Department of Medical Imaging, University of Toronto, who provides us the testing X-ray images and valuable advices.

Finally, my thanks give to my wife and my family for their great love and faith in me for all these years.

Table of Contents

List of Figures

List of Appendices

Chapter 1

Introduction

- Aim of the study

This research project focuses on the study of applying an active contour model segmentation method to X-ray images of fractured bones encased in cast material. Segmenting fractured bones and the alignment calculation of bone fractures on X-ray images using a computer will help radiologists in their diagnoses. This process will reduce the amount of time required in assessing the success of a fracture treatment. The aim of this research is to develop a computer aided diagnostic tool for analyzing bone fractures and determining their alignment. The key part of designing this tool is to develop a method to effectively segment the bone fractures from the X-ray image. Therefore, the first step of this project is to study image segmentation methods.

Image segmentation is an important task in the field of image processing and computer vision. Image segmentation is a technique that divides the image into several regions according to their homogeneities, such as pixel grayscale/color values. It has been an active area of research for many years, but there is still no generic solution for segmentation due to the diversity of the image processing applications.

The difficult part of this segmentation task is to overcome the noisy background caused by the overlaying cast. The X-ray images with cast material, especially in the

areas around metaphysic, are extremely blurry and have very low contrast. Metaphysic and the background can barely be separated by traditional segmentation techniques in medical images; the inconsistent intensity pattern that changes from one X-ray to the next causes difficulties and failures in the current segmentation processes.

- Active contour models

Active contour models [1] [2] have been widely applied in image segmentation applications over the past decades. Their robustness against the noise and the irregularities of the segmenting object, plus the incorporation of priori knowledge of objects of interest, make the active contour models an attractive approach for image segmentation applications. The active contour models can also be called deformable models because the active contour deforms itself towards the object of interests during the image segmentation process. There are two major branches of deformable models, the parametric one (Snakes) and the geometric one (Level set methods). The main difference between these two models is their different representations of the active contours. Parametric models are explicit, while the geometric models are defined implicitly as a property of a higher level function.

In this research study, a level set based active contour model is used in the segmentation process. An initial level set contour is placed inside the object of interest and develops itself towards the boundary of that object. A priori shape model of the object of interest is introduced into the evolving process in order to regulate the behavior of the evolving curve. Along with the curve evolving, the priori shape model

adjusts itself to match the boundary of the object of interest. The final result from the matching process is then passed to the alignment calculation process of the fractures, where the axes of the fractures are drawn and their angle displacement is calculated.

- Organization of the thesis

The rest of this thesis is organized as follows.

In **Chapter 2** a brief introduction is given to Medical Imaging and the different modalities widely used, followed by some example images. The remaining part of this chapter details the traditional image segmentation techniques, both boundary based and region based, and then explains segmentation methods using deformable models.

Chapter 3 discusses the methods used in this research study. First the edge detection techniques are examined here. Several edge detection methods are implemented comparably in this research in order to obtain better edge maps. By examining morphological operations, the morphological gradients can highlight the sharp gray-level transitions (edges) in the input image and depend less on edge directionality, thus it is the right choice for edge detection rather than the tradition methods in this research. The proposed segmentation method using level set methods based deformable models is detailed here. Mutual Information is also covered by this chapter. Finally shape moments that are used to calculate the priori shape model and the evolving curve center location and orientation are detailed here.

Chapter 4 examines the experimental simulation system and lists the experimental results.

Chapter 5 concludes the thesis with the post-experiment discussions and the possible research extensions.

Chapter 2

Medical Imaging and Image Segmentation

2.1 Introduction

Image segmentation [3] [4] is a fundamental task of the modern Medical Imaging field. More and more, physicians rely on segmentation results to overcome the difficulties of attempting to obtain accurate information from the blurry medical images and to determine the treatment process of their patients. Medical image segmentation is used to classify different anatomy features (such as bones, muscles, and soft tissues) and to extract a predefined region of interest from a single or set of medical images. Many researches [5-9] have been done on medical image segmentation during the past decades, yet there are no general segmentation methods that can be applied to every kind of medical image. This is because the segmentation methods applied upon medical images vary widely according to specific requirements for specific applications and various modalities. For example, segmenting a bone fracture has different requirements from segmenting the soft tissue of a liver. In addition, medical image noise and patients' movement also increase the segmentation difficulties and degrade the performance of a segmentation algorithm. Specialized segmentation algorithms to specific applications can often achieve better performance by taking into account prior knowledge. This research project is a classic example of the medical image segmentation.

2.2 Medical Imaging

Medical Imaging [10] plays a key role in the today's clinical analysis. Medical imaging helps the physicians and the radiologists to diagnose and cure the diseases more efficiently and accurately. With the assistance of computers, the inside of the patients can be examined on site or be taken as medical images. The images can be used for later diagnosis. Medical images have many modalities according to different image acquiring methods, such as X-rays, Ultra Sound, Computer Tomography (CT), Magnetic Resonance (MR), and Positron Emission Tomography (PET), or Single Photon Emission Computed Tomography (SPECT).

The example X-rays/Ultrasound/CT/MR/PET images are shown in the figure below.

Fig1 X-rays, Ultrasound, CT, MR and PET images

2.3 Image Segmentation

Image segmentation implements one of the general tasks of computer vision. Image segmentation partitions an image into constituent numbers of non-overlapping regions with respect to their homogeneous characteristics such as intensity or texture background. Mathematical representation of image segmentation can be described as follows.

$$I = \bigcup_{k=1}^{K} S_k,\ S_i \cap S_j = \phi, \text{for} i \neq j$$

Here I represent the input image, S_k represents the segmented region.

Numerous methods [4] [11] have been developed during the past decades, among those most can be grouped into two basic categories: boundary based approaches, region based approaches.

2.3.1 Boundary Based approaches

The boundaries are defined as separation indicators (luminance transitions) between regions inside an image, or edges. In the real world, mammals use the edge information to perceive or recognize objects with special meanings; in the computer world, accurate edge information can be used as input for higher level processes of an image processing application, such as image segmentation and pattern recognition. There are two key components of the boundary based segmentation methods: edge detection and edge linking.

Edge detection

Edge detection [12] refers to the process of identifying and locating the sharp discontinuities or abrupt intensity level changes in an image. In computer vision it is traditionally implemented by the convolving the input image with some form of linear filters, which usually approximates a first or second derivative operator. First derivative and second derivative methods are two major categories of edge detection.

- The first derivative edge detectors try to find local maxima in the gradient of an image. Gradient is the gray level change with directions; it can be computed in two directions, horizontal and vertical. It can be expressed by $\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$ with its magnitude of $\sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$ and direction of $\tan^{-1}(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x})$.

  Robert kernels (see Table 1) approximate the first derivatives with respect to the two diagonal directions. But in practice Robert kernels are to small to reliably find edges especially with the presence of noise.

  Another approach Prewitt kernels (see Table 1) are based on the idea of the central difference, but it is also sensitive to noise. By taking the average in both x and y directions, noise reductions is achieved to some degree.

  As with the Prewitt kernels, the Sobel kernels (see Table 1) rely on the central differences, but it gives more weight to the central pixels when doing the

averaging. The Sobel kernels can also be thought of as 3x3 approximations to first-derivative-of-Gaussian kernels.

	Row Gradient	Column Gradient
Pixel Difference	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Separated pixel difference	$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$
Prewitt	$\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
Roberts	$\begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Sobel	$\frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
Frei-Chen	$\frac{1}{2+\sqrt{2}}\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix}$	$\frac{1}{2+\sqrt{2}}\begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix}$

Table 1 some popular gradient operators in 3x3 mask

- The basic idea of the second derivative edge detection [13] is to find places where the second derivative is zero, because the local maximum of the first derivative (edges) is equivalent to the zero in second derivative.

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. It highlights regions of rapid intensity change and is therefore often used for edge detection. The second derivative is defined as $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

In discrete form,

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y)$$

The following are two commonly used Laplacian kernels.

0	1	0
1	-4	1
0	1	0

Kernel (1)

-1	-1	-1
-1	8	-1
-1	-1	-1

Kernel (2)

Fig2 Two commonly used Laplacian kernels

Although the above Laplacian kernels are able to find edges, but they are noise sensitive. The way the Laplacian operators are often used is to combine the use of a Gaussian smoothing operator by convolving them to form a single edge-finding operator. This is called Laplacian of Gaussian (LoG).

$$LoG(x, y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2 + y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

0	0	3	2	2	2	3	0	0
0	2	3	5	5	5	3	2	0
3	3	5	3	0	3	5	3	3
2	5	3	-12	-23	-12	3	5	2
2	5	0	-23	-40	-23	0	5	2
2	5	3	-12	-23	-12	3	5	2
3	3	5	3	0	3	5	3	3
0	2	3	5	5	5	3	2	0
0	0	3	2	2	2	3	0	0

Fig3 Discrete LoG kernel ($\sigma = 1.4$)

Edge linking

Edge linking is the second step of a boundary based image segmentation process; it groups the detected boundary elements to form lines or edges. Edge linking methods try to ensure that all the true edges obtained from the edge detection step have no breaks. However, because of the presence of noise, fragmented edge elements cannot be avoided during the edge detection process.

One way to minimize the break edges and link them all together is to analyze the properties (such as the spatial location, gradient direction and gradient magnitude) of a potential edge point and its surrounding area. If the properties are the same as other edge points, then these points can be linked to form an edge and remove the breaks. This method can be referred to as the local linking method.

Another way of edge linking is the global method, such as Hough Transform (HT) [14]. HT transforms the edge elements into a parameter space (Hough space), which is a joint histogram of the parameters of the model of line or curve being detected.

The peaks in this histogram then indicate the presence and location of the lines or curves being detected. For example, consider an edge point (x_i, y_i) obtained from the previous edge detection step, any line passing through this point can be expressed as,

$$y_i = ax_i + b$$

This can be rewritten as

$$b = -ax_i + y_i,$$

$$\text{or } x_i \cos\theta + y_i \sin\theta = \rho.$$

Here ρ is the distance of the line from the origin and θ is the angle between this line and x-axis. The diagram below shows the basic idea of Hough Transform.

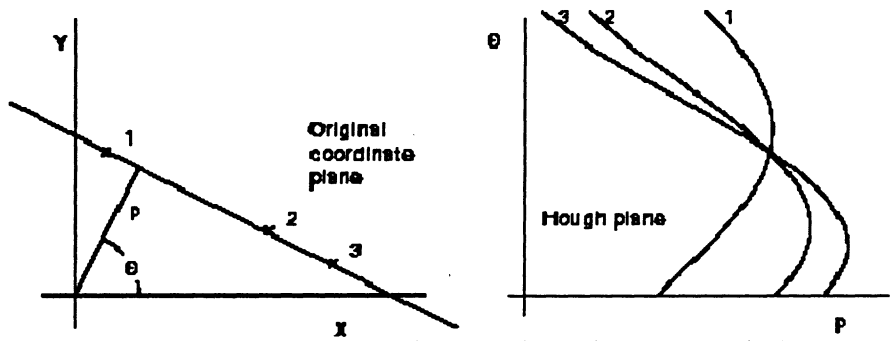


Fig4 Hough Transform: Original coordinate plane and Hough plane

2.3.2 Region Based approaches

Unlike the boundary based approaches, region based segmentation methods attempt to group regions according to common image properties such as intensity values, region texture, or patterns. The following describes some commonly used region based segmentation methods.

Thresholding

The thresholding technique makes decisions based on local pixel information, uses an estimated or experimental value, *threshold*, to group image pixels into different regions and can be very efficient when object intensity levels are totally different from the background. For an example, an 8 bit gray image has 256 intensity levels from 0 to 255, if an object inside the image has intensity levels of all its pixels between 80 and 200, and background has its intensity levels below 50, then it is easy to segment this object out from the image by setting the threshold value between 50 and 80. The basic concept of thresholding technique can be described as follows.

$$\text{If } I(x, y) \geq t, I(x, y) = object = 1,$$
$$\text{Else } I(x, y) = background = 0$$

Where $I(x, y)$ represents the intensity level of the image pixel at position of *(x,y)*, t represents the threshold value.

One of the simplest thresholding techniques is the histogram thresholding method, in which the threshold is chosen from the image histogram. Image histogram describes the occurring frequencies of the image intensity levels; the histogram peaks are the number of pixels that have the same intensity level. Normally most of the pixels inside an object have similar intensity levels. The number of pixels with similar intensities can be indicated as peaks in the image histogram, which means the object can be represented in the image histogram by the peaks. Thresholding the peaks is in fact segmenting the image object.

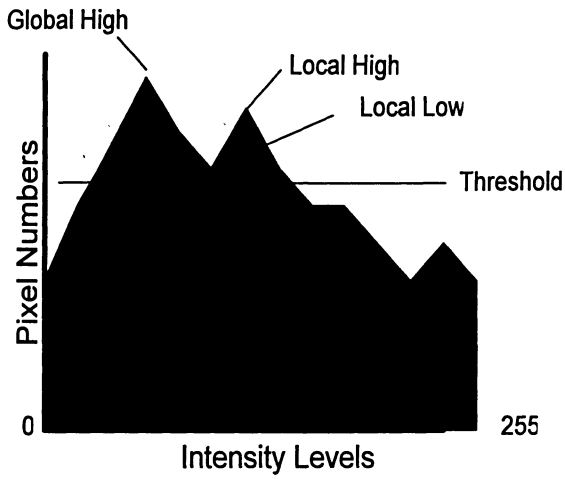


Fig5 Histogram Thresholding

Segmenting objects by histogram thresholding is a simple and easy solvable problem when there are no overlapped objects to be segmented. However when objects overlapping in the histogram domain, the histogram peaks are contributed by pixels from both overlapped image objects, and because histogram is 'global' and contains no positional information, it is difficult to separate these overlapped objects. An extension of Histogram Thresholding is the Local Thresholding. This method analyzes the local area around the global peaks, while the peaks in the local area could be considered as different objects. As shown in Fig2 above, the Global High value was found by global thresholding; all the peaks above the Threshold will be considered as one object; by doing local thresholding the Local High and the Local Low are analyzed and are identified as different objects.

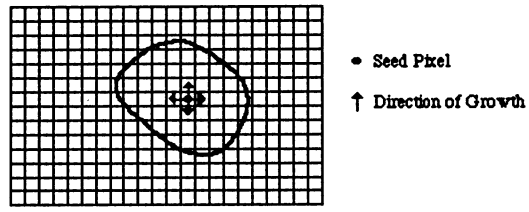
Region Growing

Region Growing extracts regions from an image based on predefined criteria such as intensity information and edges. The basic idea of a simple region growing method is

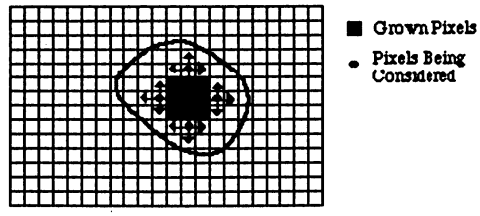
to manually select a seed point and all the pixels connected to the seed are extracted if they have the same intensity value as the initial seed point. The connected pixels then are considered as seeds for next step of region growing. The major drawbacks of this method are the manual interaction of obtaining the seed points and its sensitivity to noise. As noise affects the intensity values of neighbor pixels around the seed point, those neighbor pixels would not be connected when the region grows and a hole will be formed in that region.

$$P(R_i) = True : if \mid f(p_{neighbor}) - f(p_{seed}) \mid < T$$

Here $f(p_{seed})$ is the intensity level of the seed pixel and $f(p_{neighbor})$ is the intensity level of the neighbor pixel. When the absolute difference between the two intensity levels is less than a small value T, the neighbor pixel can be connected and included in the region R_i, $P(R_i)$ means Pixel of Region. The figure below describes the basic idea of the seeded region growing method.



(a) Start of Growing a Region



(b) Growing Process After a Few Iterations

Fig6 Region Growing

K-Means Clustering

Unlike the region growing method, the K-Means Clustering method [15] is an iterative process; it groups image pixels with similar properties, such as intensity values into segments or regions. The K-Means Clustering method takes the following steps.

- At the beginning, random pixels are selected from the image as the *centroids* for different image clusters. The *centroid* of a cluster is defined as the average intensity value of the cluster pixels.

$$C_c = \frac{1}{N}\sum_{i=1}^{N} f(p_i)$$

  Where C_c is the cluster *centroid*, $f(p_i)$ is the intensity value of the cluster pixel p_i.

- Each pixel is examined during the iterative process and is assigned to the cluster with the closest *centroid* matching its intensity value.
- After all pixels are group into different clusters, the *centroids* of clusters are recalculated, then the second step and this step are repeated until one of the clusters remains unchanged, which means no more pixels are grouped into or out from this cluster.

Because the K-Means Clustering method needs to start by providing an initial value of K, the concern of how to obtain a good initial value of K arises. However, this is also the key to success for this method. The optimal K value can be obtained from comparing the results of using different K values according to [16]. Noticing that intensity is not the only choice when grouping pixels into clusters, other criteria such

as position and orientation can also be used. Different criteria can be combined in the K-Means Clustering method to achieve better performance.

There are many region-growing techniques and their mathematical definitions can be found in the literature.

Deformable models

Deformable models [17] are physics-based models that deform under the laws of Newton mechanics, in particular, by the theory of elasticity expressed in the Lagrange dynamics. Deformable models provide a robust foundation for the representation, segmentation, and manipulation of complex objects in an image. Late developments in segmentation prefer using deformable models due to some outstanding properties of these models. By specifying the constraints in the model, deformable model can overcome some limitations of the traditional segmentation methods. For example, segmentations using boundary based approach tend to fail when the brightness between the object and background are weakly defined in the image. Region growing algorithms for segmentation might cause either over or under segment when the region within the same object are poorly presented by the image. Deformable models can also adapt to the object's shape, size, and color in the segmentation process.

Deformable models can be broadly classified into two competing categories: parametric deformable models (snakes) and geometric deformable models (level set methods).

- Parametric deformable models or snakes [18] consist of an elastic curve (or surface) which can dynamically conform to object shapes in response to internal forces (elastic forces) and external forces (image and constraint forces). It is more intuitive than the implicit models. Its mathematical formulation makes it easier to integrate image data, an initial estimated, desired contour properties and knowledge-based constraints, in a single extraction process.
- Geometric deformable models [19] consist basically of embedding the contour as the zero level set of a higher dimensional function and to solve the corresponding equation of motion. Such methodologies are best suited for the recovery of objects with complex shapes and unknown topologies. However, due to the higher dimensional formulation, implicit models are not as convenient as the parametric ones, for shape analysis and visualization, or for user interaction.

Snakes

The most known parametric deformable model is the active contour model, or snake [18], was first introduced by Kass et al., as a powerful tool for image segmentation and has been successfully applied in a variety of computer vision and image processing problems [20-26]. The basic idea of snake models is that an initial elastic curve deforms itself towards the object of interest according to the internal and external forces. The external force is responsible for the deformation and the internal force is responsible for the curve smoothness.

The main advantage of snake models is that it can overcome several photometric abnormalities such as contour gaps by integrating the initial contour estimation and

desired contour properties into the segmentation process. The snake seeks energy minima which stops the curve deformation.

It is parametrically defined as $v(s) = (x(s), y(s))$, where $x(s)$, $y(s)$ are x, y co-ordinates along the contour and s is from (0, 1). The energy functional to be minimized is:

$$E^*_{snake} = \int_0^1 E_{snake}(v(s))ds = \int_0^1 [E_{internal}(v(s)) + E_{external}(v(s))]ds$$

The internal energy can be written as:

$$E_{internal} = \alpha(s)|\frac{dv}{ds}|^2 + \beta(s)|\frac{d^2v}{ds^2}|^2$$

Where $\alpha(s), \beta(s)$ specify the *elasticity* and *stiffness* of the snake.

The external energy function $E_{external}$ is derived from the image so that it takes on its smaller value at the features of interest, such as boundaries.

However, the original snake suffers from the strong sensitivity to the initial contour position and can not deal with topological changes. Some modified snake models can overcome these kinds of problems. For example, T-Snake [27] embeds the snake model in the framework of a simple decomposition of the image domain. It uses region-based statistics to weigh an inflation force; Dual Active Contour Model [28] makes use of two contours to avoid the local minima. The inner one expands from the inside of the target and the other contracts towards the target from outside. The two contours are inter-linked to provide a balanced technique with an ability to reject weak local energy minima.

Level Set Methods

Geometric deformable models [19] implemented using level set methods have several advantages over parametric models due to their intrinsic behavior, parameterization independence and ease of implementation.

- Independent from parameterization of the evolving curve. The level set model is parameterized only when the level set function evolution is complete. It means that the evolving contour gets parameterized when it stops evolving. While the parametric model has to adjust itself during the process, such as adding or removing nodes from the curve and adjusting the spaces between nodes.
- Accuracy of computation. Because of the discrete nature of the parametric model, the calculations, such as curvature and normal vectors remain rough or inaccurate. From the level set function, these drawbacks can be improved.
- Level set contours can avoid the topological obstacles of the snakes; the contours can be merged or split to adapt to the topology changes.

The level set approach was introduced in 1988 by S.Osher and J.A. Sethian [1]. Level sets are designed to handle problems in which the evolving interfaces can develop sharp corners and cusps, change topology and become very complex. It provides a mathematical formulation for tracking the motion of a curve which can be recast as front propagation problems. The deformation of the level set method depends on the evolving process of the initial curve; while the evolving process depends on the evolving speed of the front, and the speed again relies on the external force applied to the evolving curve. Most of the challenges in level set approach result from the need

to construct an adequate model for the speed function. The classical level set speed function models rely on the edge gradient information to stop the curve evolving; these models can detect objects with edges defined by gradient.

Level set method offers a powerful approach to image segmentation since it can handle any of the cavities, concavities, splitting and merging. It has been applied in a variety of fields including the medical image segmentation and has achieved much success. In [29] an efficient adaptive multigrid level set method for front propagation purposing in 3D medical image processing and segmentation was proposed and it successfully solved the problem of segmentation with non sharp boundaries. [30] proposed an alterative image segmentation method which combines the morphological watershed transform and level set methods. A new segmentation method was proposed in [31] in order to segment a moving object against a still background, which relies on the level set method to handle topological changes while providing closed boundaries.

The shape provides important knowledge about the segmenting object and it can be incorporated into the segmenting process. Several methods of incorporating prior shape information into the boundary determination of level set have been developed. Staib and Duncan [32] introduced a parametric point model based on an elliptic Fourier decomposition of the landmark points. The parameters of their curve are calculated to optimize the match between the segmenting curve and the gradient of the image. Wang and Staib [33] applied a statistical point model for the segmenting curve by using principal component analysis to the covariance matrices that capture

the statistical variations of the landmark points. Leventon et al. [34] incorporated shape information as a prior model to restrict the flow of the geodesic active contour. Their shape model is derived by performing principle component analysis on a collection of signed distance maps of the training shape. The curve evolves according to two competing forces: the gradient force and the force exerted by the estimated shape where the parameters of the shape are calculated based on the image gradient and the current position of the curve. Chen [35] proposed a model that uses the geodesic contour model and an "average shape" as the prior shape which defines a term in the evolving function of the model. This approach showed potential for image segmentation incorporating a shape that can be collected before hand.

In this research project, a prior shape is incorporated into the active contour model as the global constraint to do the bone fracture detection. Bone fractures are common occurrence, manual inspection of bone fractures is tedious and time consuming and their presence can often be missed during X-ray diagnosis. Several methods have been developed for bone fracture detection during past decades. Tian et al. [36] developed a method to detect bone fractures by computing the angle between the neck axis and shaft axis, the bone fracture detection makes uses of Canny edge detection technique with Hough transform linking method and snake model to extract the femur boundary and to form a continuous contour of the femur respectively. This method could fail in cases that the snake points are not accurately located at bone boundaries. M. Donnelley and G. Knowles [37], introduced a method detecting fractures in long bones takes three steps: first, extracting bone edges using a

morphological scale space approach to smooth the image and preserve the important boundary information; the second step is the edge lines approximation using Hough Transformation; the last step is to do the gradient analysis for determination of the presence of fractures. It allows the abnormal regions including the fractures to be highlighted and detected. This method is based on the accuracy of the edge detection to calculate the composite gradient measure, if some edges are missing or very weak, the false positive results will be produced. Furthermore, the fractures parallel to the bone edge are not detected as well as those perpendicular ones.

The ACM with prior shape method studied in this research is based on level set active contour model, which detects the bone fractures by using an active contour deforming itself to find the bone boundary; a prior shape incorporated in the process regulates the behavior of the active contour from evolving out of weak edges. The active contour propagates with a velocity function defined based on the image gradients and the prior shape information. The propagation stops when the active contour arrives at high gradients or closely matches the prior shape. It segmented out the fractured bone pieces and then calculated their axes in order to display their alignment.

Chapter 3

Methods

In this chapter, the methods used to segment the fractured bone in this thesis project are introduced sequentially from the mathematical point of view.

3.1 Edge Detection

Several edge detection techniques are implemented comparably in this thesis project in order to obtain an optimal edge map from the original image.

3.1.1 Robert Edge Detector

Robert edge detector is one of the simplest 2D spatial gradient edge detectors; it highlights the high frequency components which often correspond to edges. Robert edge detector consists of 2 convolution kernels (2x2) with directions of one orthogonal to the other (see figure below).

1	0
0	-1

Gx

0	1
-1	0

Gy

Fig7 Robert Cross Convolution kernels

The kernels *Gx* and *Gy* are designed to respond to edges at diagonal directions ($\pm 45^{\circ}$).

When convolving them with an input image, the gradient of a corresponding point *(x,y)* is given by

$$|G| = \sqrt{Gx^2 + Gy^2}$$

or its approximation $|G| \approx |Gx| + |Gy|$.

The angle of orientation is given by

$$\theta = \arctan(Gy / Gx) - 3\pi / 4$$

The reason of subtracting the angle by 135 degrees is that the kernels are convolved in the diagonal directions, not in the *x* and *y* directions.

The gradient magnitude can be approximated by combining the two kernels into one, which gives the approximate magnitude $|G| \approx |P_1 - P_4| + |P_2 - P_3|$.

P_1	P_2
P_3	P_4

Fig8 Approximation kernel of Robert Edge Detector

The Robert edge detector is very computation efficient, because there are only 4 pixels are involved in the calculation to determine the output pixel value; on the other hand, it is sensitive to noise due to its small size and it can only perform well when the edges are very sharp.

3.1.2 Sobel Edge Detector

Sobel edge detector consists of a pair of 3x3 convolution kernels. Similar with Robert kernels, the two kernels are orthogonal to each other; unlike the Robert kernels, it performs convolution in the direction of x and y.

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

Gx

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

Gy

Fig9 Sobel Edge Detector kernels

The gradient is calculated by

$$|G| = \sqrt{Gx^2 + Gy^2}$$

or $$|G| = |Gx| + |Gy|.$$

The angle of orientation is given by

$$\theta = \arctan(Gy / Gx)$$

The gradient magnitude can be approximated by convolving the kernel below with an input image.

P_1	P_2	P_3
P_4	P_5	P_6
P_7	P_8	P_9

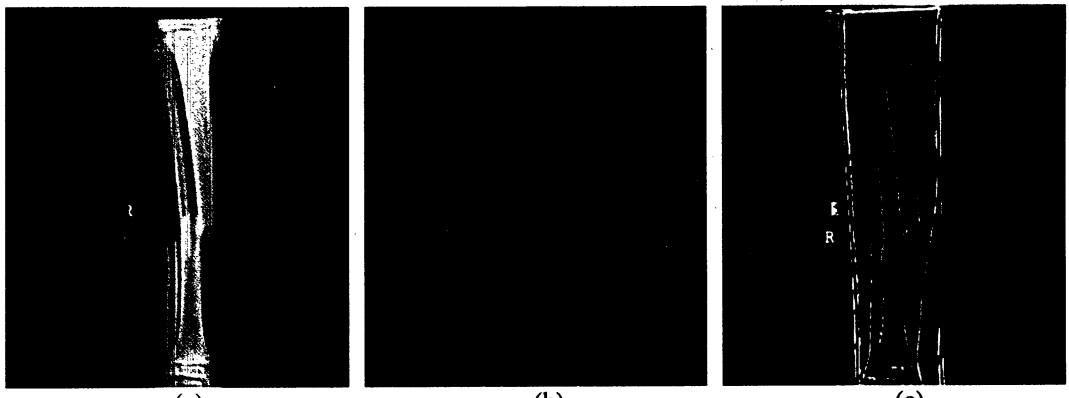
Fig10 Approximation kernel of Sobel Edge Detector

The approximated magnitude is given by

$$|G| = |(P_1 + 2P_2 + P_3) - (P_7 + 2P_8 + P_9)| + |(P_3 + 2P_6 + P_9) - (P_1 + 2P_4 + P_7)|$$

Compared with the Robert edge detectors, the Sobel is slower in computation, but it is less sensitive to noise because of its bigger kernel size.

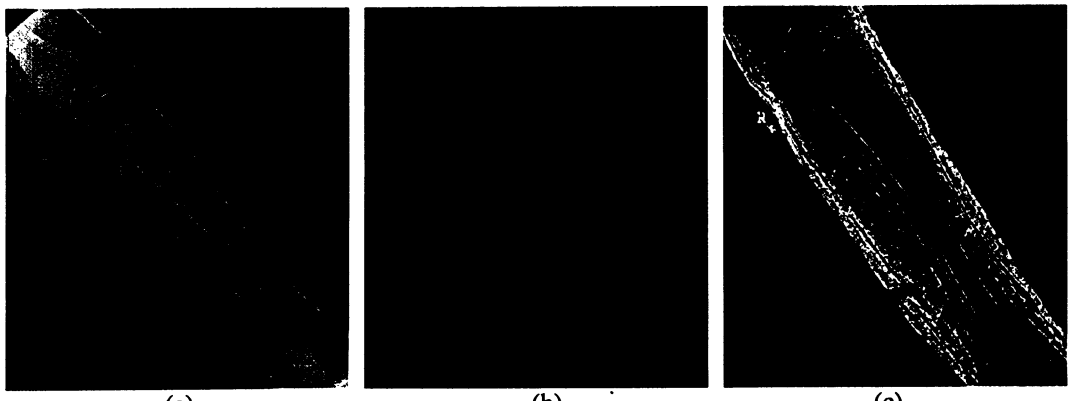
The following are some visual results for the edge detectors and their comparison:



(a) (b) (c)

Fig11 Robert/Sobel Edge Detector results (1)

(a) Original Fractured arm X-ray picture without cast material applied
(b) Robert edge detector result
(c) Sobel edge detector result



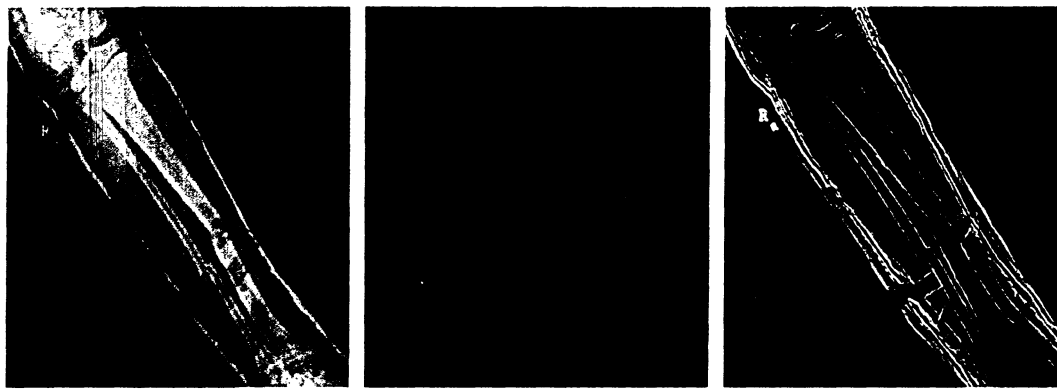
(a) (b) (c)

Fig12 Robert/Sobel Edge Detector results (2)

(a) Original Fractured arm X-ray picture with cast material applied
(b) Robert edge detector result
(c) Sobel edge detector result

When cast material is applied to the X-ray images, the edge detection results are poor due to the noise introduced by cast material and gradient edge detectors are sensitive

to noise. Therefore before edge detection the input image is always processed by a noise reduction process; the noise reduction is achieved by using Gaussian Noise Filter (GNF) in this thesis project. The following pictures show the results of Gaussian + edge detectors.



(a) (b) (c)

Fig13 GNF + Robert, GNF + Sobel results

(a) Fractured arm X-ray picture with cast material applied
(b) Gaussian Noise Filter + Robert edge detector result
(c) Gaussian Noise Filter + Sobel edge detector result

The Gaussian noise filter used here has the kernel as below.

$$\frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Fig14 Gaussian Noise Filter Kernel ($\sigma = 1.4$)

3.1.3 Canny Edge Detector

Canny's edge detector [38] improved the edge detection algorithms described above. The canny edge detection algorithm consists of the following steps.

- Noise reduction. Filtering out noise is the most first step that many image processing applications always take, and so is canny edge detector. By applying Gaussian filter using standard convolution methods, the Gaussian filter kernel is sliding over the image. The larger the width of the Gaussian mask, the lower is the detector's sensitivity to noise.

$$g(m,n) = f(m,n) * G_\sigma(m,n)$$

$$\text{Where, } G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{m^2+n^2}{2\sigma^2}\right]$$

  The filter kernel used is the one shown in Fig15.

- Edge strength calculation by gradient edge operator. This step performs a 2-D spatial gradient measurement on the image (the Sobel operator is used in this research project). Once done, the edge strength (approximated gradient magnitude) of each point can be found. The Sobel operators are one pair of 3x3 convolution masks and are applied upon *x* and *y* directions respectively.

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

Gx

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

Gy

The gradient magnitude or edge strength is calculated by $|G| = |Gx| + |Gy|$.

- Edge direction. The edge direction is calculated by the inverse tangent of the gradient sum along the directions y and x. Due to the possible zero value of x direction gradient, the error will occur. To avoid this, when x direction gradient is zero we assign the edge direction to be 0 or 90 degrees according to the value of the y direction gradient. If Gy is 0, edge direction is 0 degree, otherwise it is 90 degrees.

  Edge_direction_theta = inverse_tangent (sumGy/sumGx)

- Edge direction trace. This step performs an operation relating the calculated edge direction with the pixels inside the image, which means we group the edges into four directions: horizontal (yellow, 0 to 22.5 & 157.5 to 180 degrees), positive diagonal (green, 22.5 to 67.5 degrees), vertical (blue, 67.5 to 112.5 degrees) and negative diagonal (red, 112.5 to 157.5 degrees). See Fig19 below.
- Non-maximum suppression. The non-maximum suppression is used to trace along the edge in the edge direction and suppress any pixel value that is not considered to be an edge.
- Dashed edge repair. The dashed edge could occur because the edge pixels gradients are below and above one threshold. Thus when threshold applies, some points value are set to zero. To avoid this, Canny adopted the two-threshold (low and high) method. If the magnitude is below the first threshold, it is set to zero; if it is above the high threshold, it is made an edge; and if the magnitude is between the 2 thresholds, it is set to zero unless there is a path from this pixel to a pixel with a gradient above the high threshold.

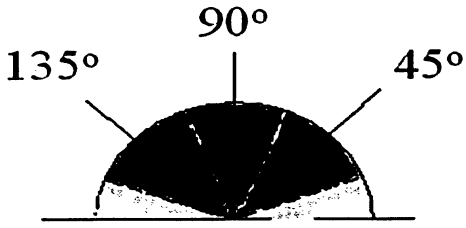


Fig15 Canny Edge Direction Trace

The figure below shows the steps taken for a Canny edge detection process.

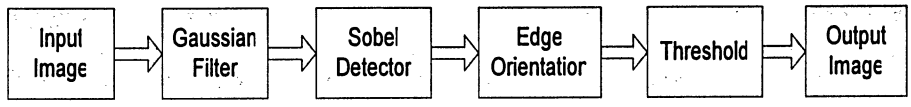
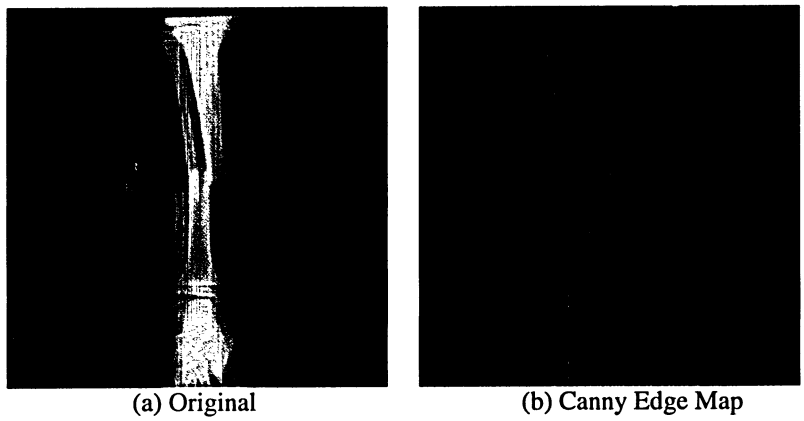
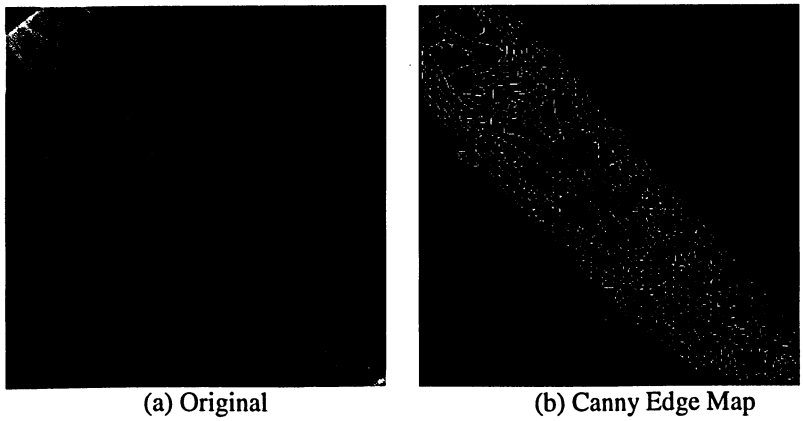


Fig16 Canny Edge Detection Steps

The following are some visual results from Canny Edge Detector.



(a) Original (b) Canny Edge Map



(a) Original (b) Canny Edge Map

Fig17 Canny Edge Detector results

3.1.4 Laplacian Edge Detection

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection. The second order derivative of an image is defined as,

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

This formulation consists of two different derivatives, which represent x direction and y direction respectively, the discrete form can be expressed as below.

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y)$$

It also can be expressed as Kernel (1),

0	1	0
1	-4	1
0	1	0

Kernel (1)

-1	-1	-1
-1	8	-1
-1	-1	-1

Kernel (2)

Fig18 Two commonly used Laplacian kernels

Another slightly different version from Kernel (1) incorporates two diagonal terms, which introduces Kernel (2).

The core of Laplacian edge detection is the Laplacian of Gaussian (LoG), since the Laplacian edge detector alone is sensitive to noise. Traditionally, a Gaussian smoothing filter is applied before Laplacian in order to achieve the noise reduction. The drawback is that two convolution steps need to be performed separately, it is computation consuming. Because of the associative property of the convolution process, the two convolution steps can be combined together, thus a kernel of LoG can be formed as below.

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1-\frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\nabla^2(G \otimes I) = \nabla^2 G \otimes I$$

Where ∇^2 represents the Laplacian operation, and G represents the Gaussian smoothing operation, I is the input image. The discrete approximation of LoG kernel can be found in Fig3.

The figure below shows the visual results of the LoG edge detector.

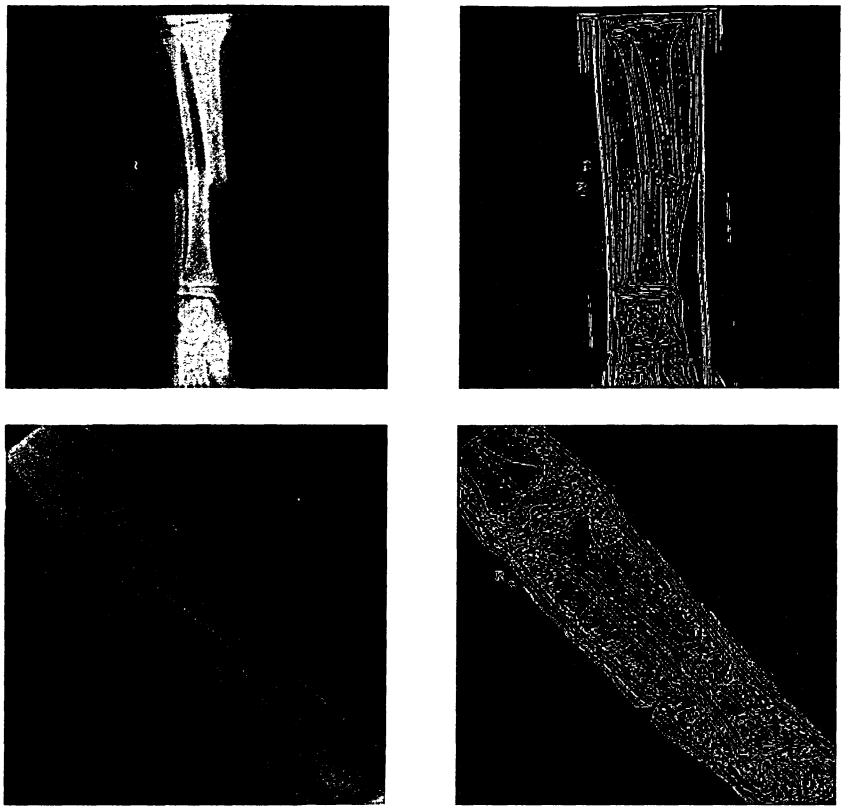


Fig19 Laplacian of Gaussian results

3.2 Morphological Operations

3.2.1 Set Theory

Mathematical Morphology [3] is based on Set Theory; it offers a unified and powerful approach to numerous image processing problems. Sets in mathematical morphology represent objects in an image. The basic set theory operations can be represented by the Figures below.

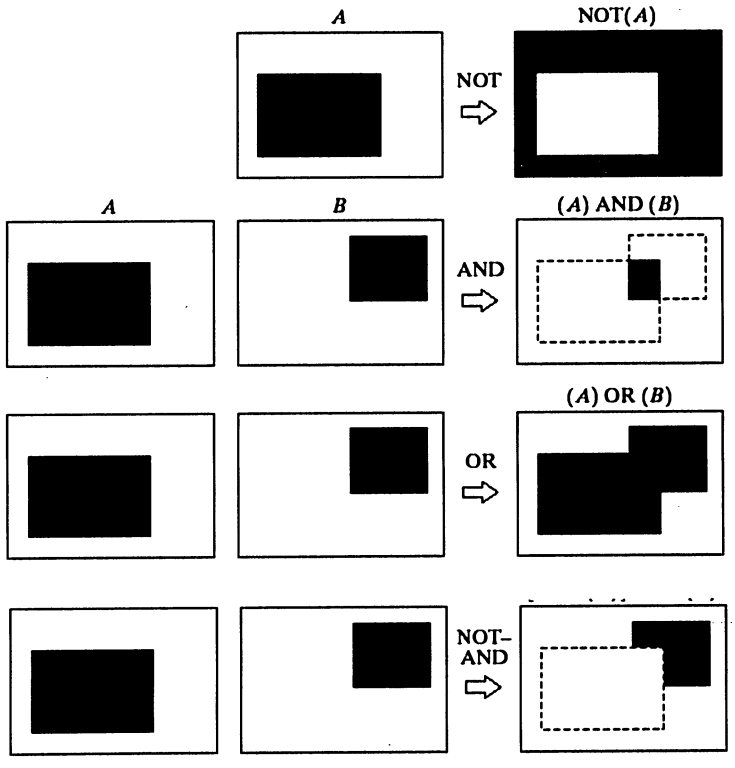


Fig20 Basic Set Theory Operations

Fundamentally morphological image processing is very like spatial filtering and the structuring element is moved across every pixel in the original image to give a pixel in a new processed image. A structuring element is a matrix consisting of only 0's and 1's that can have any shape and size.

3.2.2 Mathematical Morphological Operations

The value of this new pixel depends on the operation performed. There are 4 basic morphology operations: Dilation, Erosion, Opening and Closing, and their definitions are described below.

- Dilation:

  Dilation of image *f* by structuring element *s* is given by

  $$f \oplus s = \{z \mid [\hat{(s)}_z \cap f] \subseteq f\}$$

  The structuring element *s* is positioned with its origin at *(x, y)* and the new dilation image is determined by: Translating the origin of the reflection of the structuring element by point *z* inside the original image *f*, the translated result then intersects with the original image *f*, the point *z* is in the dilation image if it satisfies that the intersection result is totally included inside the original image *f*.

- Erosion:

  Erosion of image *f* by structuring element *s* is given by

  $$f \ominus s = \{z \mid (s)_z \subseteq X\}$$

  The structuring element s is positioned with its origin at *(x, y)* and the new erosion image is determined by: Translating the origin of the structuring element along the point *z* inside the original image *f*. The point z is in the erosion image if it satisfies that the translated result in included by the original image *f*.

- Opening

  The opening of image *f* by structuring element *s*, denoted by $f \circ s$ which is simply an erosion followed by a dilation.

  $$f \circ s = (f \ominus s) \oplus s$$

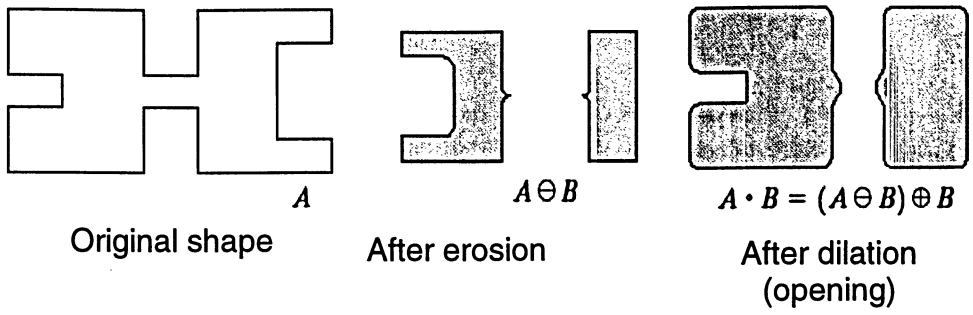


Fig21 Opening Operations

- Closing

  The closing of image f by structuring element s, denoted by $f \bullet s$ which is simply a dilation followed by an erosion.

$$f \bullet s = (f \oplus s) \ominus s$$

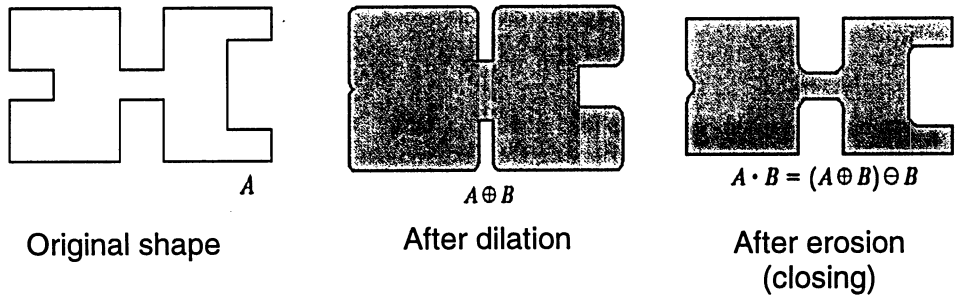


Fig22 Closing Operations

The TopHat Transform is an extension of morphology operations, which is defined as the difference between the original image and the closing/opening image. Thus there are two versions of TopHat Transforms.

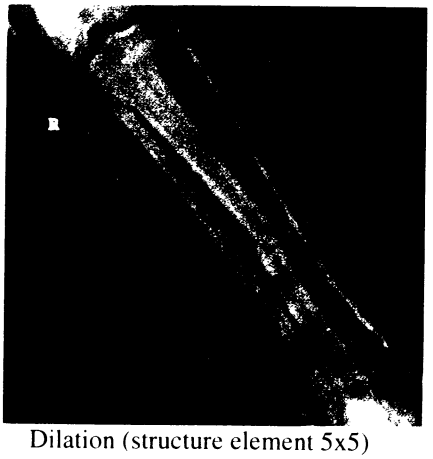
- $TopHat(A,B) = A - (A \circ B)$, the difference between the original image and opening image
- $TopHat(A,B) = (A \bullet B) - A$, the difference between the closing image and original image

The purpose of morphological processing is primarily to remove imperfections added during segmentation. The morphological operations are used in this thesis project as low level image processing techniques of noise removal and edge detection.

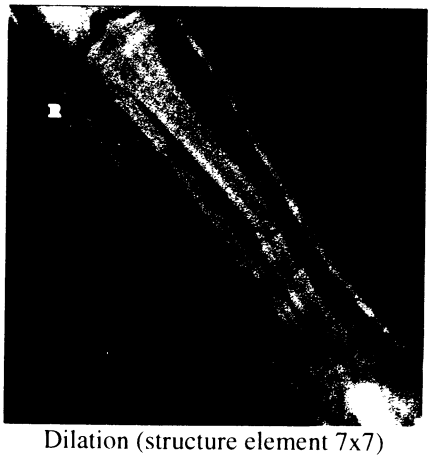
3.2.3 Morphological Operation Visual Results

The following are some visual results of the Morphological operations.

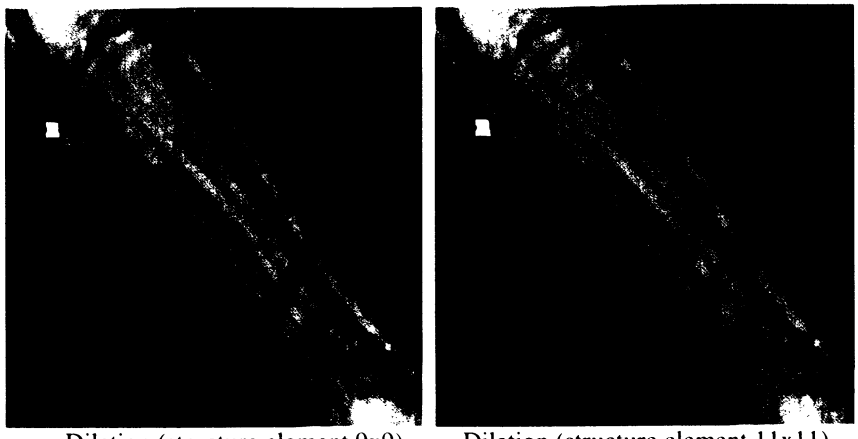
Original X-ray



Dilation (structure element 5x5)



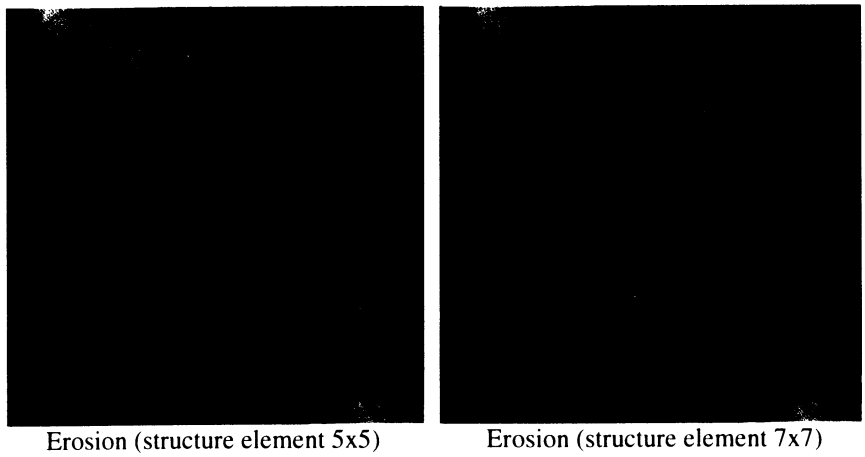
Dilation (structure element 7x7)



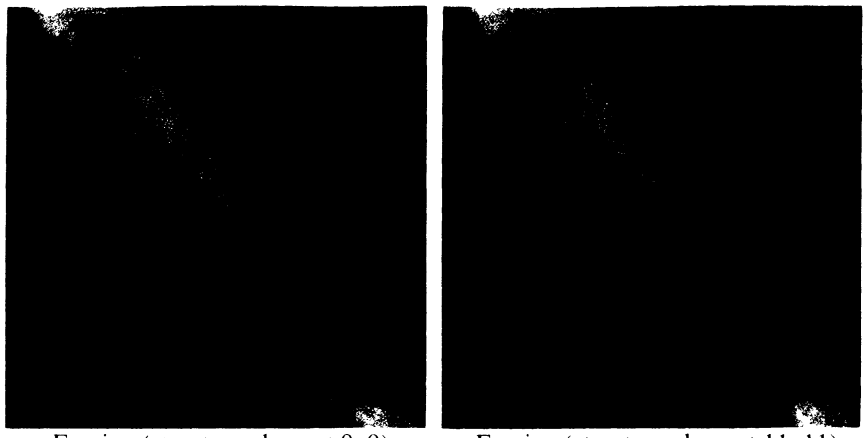
Dilation (structure element 9x9) Dilation (structure element 11x11)

Fig23 Dilation results (structuring elements size of 5x5, 7x7, 9x9, 11x11)

The above figures show the results of dilation operation with different sizes of structuring elements. The following figures show the results of erosion operation with different sizes of structuring elements.



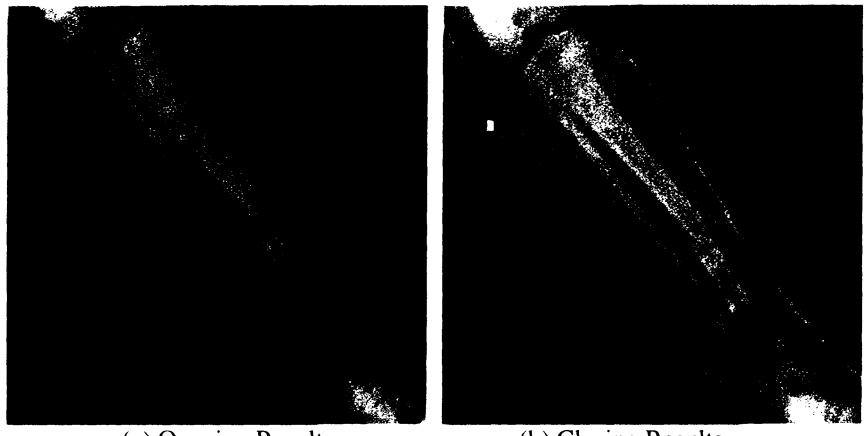
Erosion (structure element 5x5) Erosion (structure element 7x7)



Erosion (structure element 9x9) Erosion (structure element 11x11)

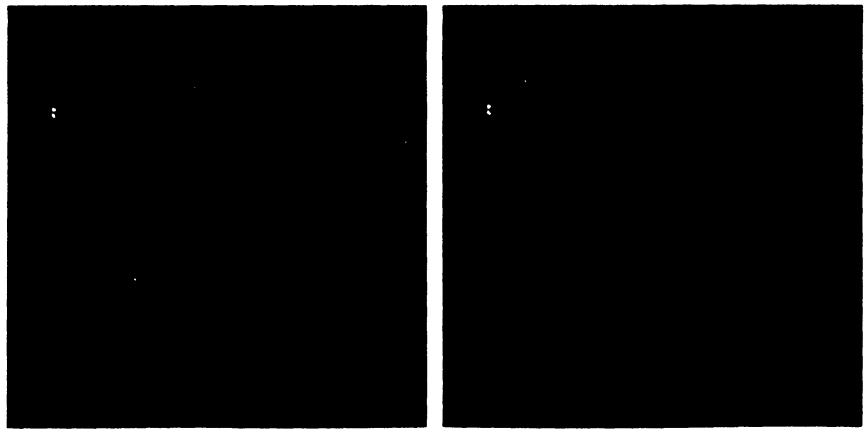
Fig24 Erosion results (structuring elements size of 5x5, 7x7, 9x9, 11x11)

The results of opening, closing operations and TopHat operations are shown below.



(a) Opening Results (b) Closing Results

Fig25 Opening and Closing Results



(a) TopHat1 $A - (A \circ B)$ (b) TopHat2 $(A \bullet B) - A$

Fig26 TopHat1 and TopHat2 Results

The following results are obtained from subtracting the dilation image with the erosion image.

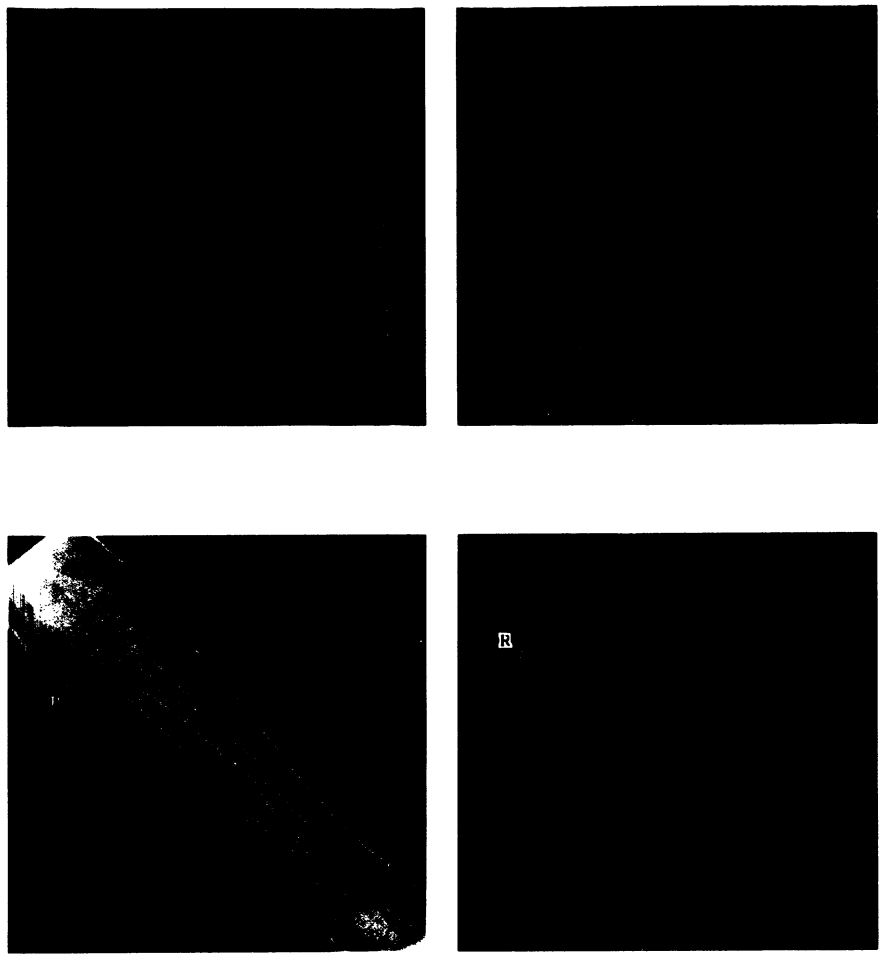


Fig27 Morphological Operation Results

3.3 Mutual Information

Mutual information [39] concerns about the measurement of the information carried by a variety of media, such as telegraphs, voice, and images. It is derived from Information Theory. The study of *Entropy*, a literature term for *measure of information*, originates from the communication theory, which cares about the broadcast of a message from a sender to a receiver. Hartley [40] first defined a measure of information of a message in 1928 by representing a message by a string of symbols; each symbol has different possibilities of occurring in the message.

Let M be a message, n represents the number of symbols in the M, and s represents different possibilities for each one of n symbols, thus there are s^n different possible messages. The key problem is to seek a measure H as for the information the message contains. Hartley first gave H the definition as,

$$H = n \log s = \log s^n$$

Shannon introduced an adapted information measure; it overcame the drawback of Hartley information measure, which considered the occurrences of symbols in a message equally. By Shannon's information measure, the weighted symbols are measured by their occurrences in a message, or their probabilities. The Shannon's entropy [41] definition is given below,

$$H = \sum_i p_i \log \frac{1}{p_i} = -\sum_i p_i \log p_i$$

The p_i represents the probability of occurrence for the event i. The higher the value of H, the more the uncertainty, and more information the message contains.

In the information theory, the Mutual Information (MI) [42] [43] of two random variables is a quantity that measures the mutual dependence of the two variables. MI measures the information about X that is shared by Y. In image processing field, MI tells the amount of uncertainty about image X minus the uncertainty about X when Y is known (vice versa). In mathematics, there are 4 MI definitions shown as below.

$$I(X, Y) = H(X) - H(X|Y)$$

$$I(X, Y) = H(Y) - H(Y|X)$$

$$I(X, Y) = H(X) + H(Y) - H(X, Y)$$

$$I(X,Y) = \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

The second definition points out that maximization of the MI are equivalent to minimize the joint entropy. And the third definition describes a measure of the Kullback-Leibler distance between the joint distribution of the images' gray values $p(x,y)$ and the joint distribution in case of independence of the two images, $p(x)p(y)$.

The MI has the following properties.

- $I(X,Y) = I(Y,X)$, symmetric property.
- $I(X,X) = H(X)$, the information image X contains about itself is equal to the information (entropy) of image X.

- $I(X,Y) <= H(X), I(X,Y) <= H(Y)$, the information the images contain about each other will never greater than the information in the images themselves.
- $I(X,Y) >= 0$
- $I(X,Y) = 0$, if and only if X and Y are independent.

The MI is used in image registration as a criterion that states the images are geometrically aligned by the transformation T_α for which $I(X,Y)$ is maximal. It is assumed that the amount of information that X contains about Y is the maximal in the registered position. In this research project, the MI between the priori shape model and the evolving curve was calculated as an attempt of measuring their shape similarity.

3.4 Deformable Models

3.4.1 Snakes (Parametric deformable models)

Snakes, or active contour models, first introduced by Kass et al in 1988, can be viewed as *Lagrangian* geometric formulations wherein the boundary of the model is represented in a parametric form. The deformation energy function is minimized with 'internal' and 'external' energies along its boundary. The geometric information is considered to be internal energy and image gradients are external energy. The snakes act like elastic bodies that stabilize when the energy function is minimized.

The snake is parametrically defined as $v(s) = (x(s), y(s))$, where $x(s), y(s)$ are x, y co-ordinates along the contour and s is from [0, 1].

The energy functional to be minimized is:

$$E^*_{snake} = \int_0^1 E_{snake}(v(s))ds = \int_0^1 [E_{internal}(v(s)) + E_{external}(v(s))]ds$$

The internal energy can be written as:

$$E_{internal} = \alpha(s)\left|\frac{dv}{ds}\right|^2 + \beta(s)\left|\frac{d^2v}{ds^2}\right|^2$$

Where $\alpha(s), \beta(s)$ specify the *elasticity* and *stiffness* of the snake. The internal energy provides the snake a smoothness determination and makes the snake tend to be more continuous and smooth. On the contrary, the external energy drags and attracts the snake towards a priori known object shape and close to the object boundary.

3.4.2 Level set methods (Geometric deformable models)

Theory

The main idea of the level set method describes a closed curve Γ in the image plane as the zero level set of a higher dimensional function $\phi(x,t)$ in $\Re^3$, the value of ϕ at some point x is defined by

$$\phi(x,t=0)=\pm d\ .$$

Here d is the distance from x to the zero level set curve Γ, and the sign is chosen depending on whether the point x lies inside or outside the curve. Γ is propagating in the plane in a direction normal to itself with speed F. The closed curve Γ can be expressed by the zero level set of the level set function

$$\Gamma(t)=\{x\in R^2 \mid \phi(x,t)=0\}\ .$$

It gives the position of the front at any time t.

The evolving process of $\phi(x,t)$ can be modeled as:

$$\frac{\partial\phi}{\partial t}+F\parallel\nabla\phi\parallel=0 \text{ with } \phi(x,t=0)$$

The figure below shows the level set function transformation in 2D and 3D.

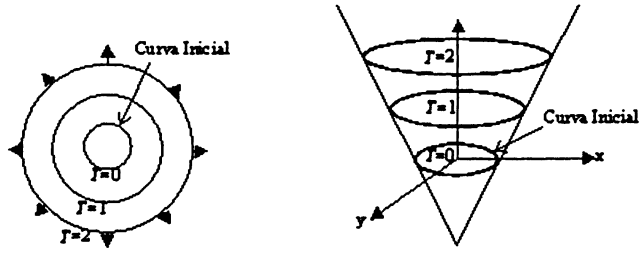


Fig28 Transformation of front motion in 2D and 3D

Signed distance function and Narrow Band

Consider an image plane Ω, the zero level set curve Γ divides it into two sub domains (as shown in the figure below): interior and exterior. The level set function is initialized as the signed distance function:

$$\phi(x) = \begin{cases} d(x,\Gamma) \\ -d(x,\Gamma) \end{cases}$$

When x belongs to the interior domain, the sign of the distance function is positive; while x belongs to the exterior domain, the sign is negative; the distance is zero when x is on the boundary.

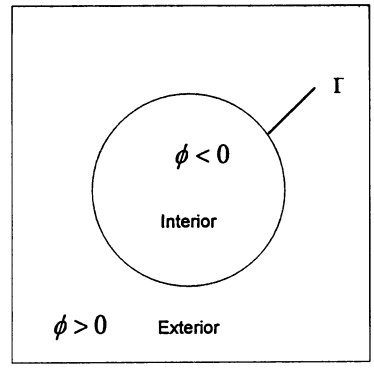


Fig29 Zero level set curve Γ and image plane Ω

Narrow band Level set method stems from observing that only the values of the level set function near its zero level set are essential. Thus only the values at the grid points in a narrow band around the zero level set have to be calculated. As the zero level set moves, the signed distance function in the narrow band must be maintained.

Re-initialization

The initial level set function is defined as a distance function and during the evolving process the level set function could be changed into a non-distance function. This requires that we need to preserve the level set function as a distance function while the front curve evolves. Re-initialization of distance function provides a method of preserving the level set function as distance function. The idea behind the re-initializing distance function is simple by stopping a level set calculation at some point in time and rebuilding a level set function corresponding to the signed distance function. Several methods have been developed to accomplish this re-initialization. Choop's method [44] reattached level sets to a bounding wire frame and simply recomputed the signed distance function at each grid point in a narrow band and find the distance to the front. Sussman, Smereka and Osher [45] developed a different technique for the distance function re-initialization without explicitly finding the zero level set. But it is relatively crudeness because of the sign checking for the distance function of this iterative re-initialization method. Another method introduced by Sethian [46] is to avoid the re-initialization as much as possible and use extension velocities for replacement, but the construction process of the extension velocities is kind of time-consuming.

Segmentation with priori shape model

Shape is a powerful property to distinguish an object from its surroundings in an image. It is commonly used to complete the information provided by local properties of an image. In medical imaging, geometric shape models provide extrinsic

information about objects and are often incorporated explicitly, especially for the segmentations where prior shape information can be collected.

3.5 Shape Moments

The priori shape model needs to adjust itself to match the object of interest because the initial position of the priori shape model is often far from the matching position. Rotation and Translation are applied to the priori shape model after each evolving step. By rotation, the priori shape model makes use of the axis angle difference between the evolving curve and itself; by translation, the distance between their centroids is used. All the axis angles and centroids calculations are based on the shape moments [47], which obtain the shape center location and the orientation from object boundary information.

Important information about a shape such as its size, center location and orientation are all moment based attributes. For example, the size of a shape is the $(0,0)$th order moment of the shape. The (p,q)th moment of continuous 2D function $f(x,y)$ and discrete 2D array $p(i,j)$ are defined as below.

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)dxdy$$

$$m_{pq} = \sum_{i=1}^{M}\sum_{j=1}^{N} i^p j^q p(i,j)$$

Traditional shape moments calculation is using the occupancy array of a shape, every single pixel in the array is used. The calculation is not efficient especially when the occupancy array of an object is big. The method introduced here makes use of the shape boundary information to do the shape moments calculation. Shape corner

points are used to form numbers of triangles, the moments of these triangles are computed first, and then the moments of the shape are derived from the triangles' moments.

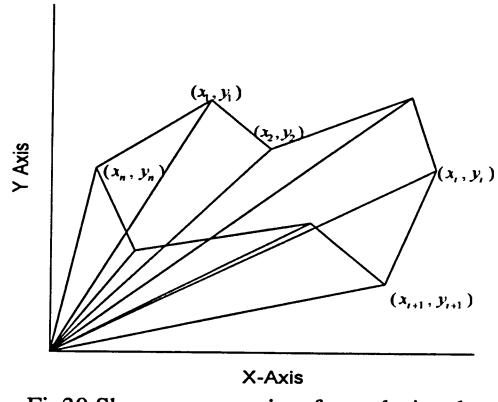


Fig30 Shape corner points formed triangles

For a triangle T with corner points of $(0,0), (x_1, y_1)$ and (x_2, y_2), the corner points form three triangles, and their *(p,q)th* moment are represented by $m_{pq,1}, m_{pq,2}, m_{pq,3}$ respectively. The triangle T's *(p,q)th* can be calculated by

$$m_{pq,T} = m_{pq,1} + m_{pq,2} - m_{pq,3}$$

Because the priori shape model and the evolving curve are represented by their boundary points, these points can form numbers of triangles; the priori shape model and the evolving curve's moments can be derived from the moments of these triangles.

$$m_{pq,S} = \sum_{i=1}^{n} m_{pq,T_i} \cdot sign(i)$$

Where the $sign(i)$ is the sign associated with triangle T_i and has the value of 1 when $\tan^{-1}\left(\frac{y_i}{x_i}\right) \geq \tan^{-1}\left(\frac{y_{i+1}}{x_{i+1}}\right)$, otherwise -1.

The shape center or the center of gravity (c_x, c_y) of a shape S can then be derived by

$$c_x = \frac{m_{10,S}}{m_{00,S}}, \quad c_y = \frac{m_{01,S}}{m_{00,S}}$$

When the origin of the coordinate system is moved to the center of the shape, we can obtain the *central moments* of the shape $\mu_{(p,q),S}$, and the shape orientation θ_S can be derived from it.

$$\mu_{(p,q),S} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-c_x)^p(y-c_y)^q f(x,y)dxdy$$

$$\mu_{11,S} = \frac{m_{01,S}m_{02,S}}{m_{00,S}}, \mu_{02,S} = \frac{m_{01,S}^2}{m_{00,S}}, \mu_{20,S} = \frac{m_{10,S}^2}{m_{00,S}}$$

$$\theta_S = \frac{1}{2}\tan^{-1}\frac{2\mu_{11,S}}{\mu_{20,S}-\mu_{02,S}}$$

Other shape attributes such as spread, elongation can also be derived from the shape moments.

Chapter 4

Experiment Results

In this chapter, results for the experiments are provided for the proposed segmentation method [48], which incorporates the priori shape model into the segmenting process using the level set active contour model. The experiments were conducted upon the X-ray images of the patients' fractured arm.

4.1 Simulation System Architecture

A simulation system is designed and developed using Microsoft visual C++. The architecture is shown below.

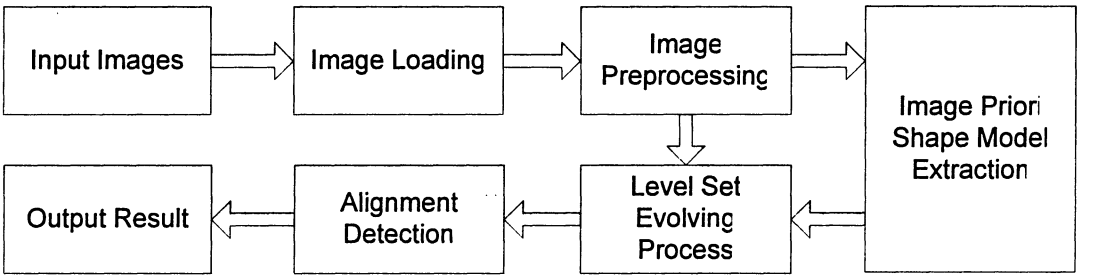


Fig31 Simulation System Architecture

There are two types of input images, the non-cast X-ray image and the cast X-ray image. The non-cast X-ray image is taken before applying the cast material on the patients' fractured arms and the cast X-ray image is taken with cast material applied. Image Loading reads in the two types of input images separately and passed them to Image Preprocessing. Image Preprocessing handles noise reduction and edge detection for the input images. Priori shape model extraction operation is carried out on the preprocessed

image results of the non-cast X-ray images because the bone boundary is much clearer than the one from the cast X-ray image's edge map. The extracted priori shape model is then inputted to the Level Set Evolving Process which starts the segmentation process by evolving an initial curve inside the fractured bone pieces. After that, the axes of the fractured bone pieces are drawn on the screen and their angle difference can be calculated for alignment diagnosis. The output results are then used by the radiologists to determine the displacement and the miss-alignment between the fractured bone pieces.

Noise reduction operation is first applied to the patient's fractured bone X-ray images. This is done by using morphological operation of Erosion with the structuring element size of 5x5 and its shape of square. An illustration is shown as below.

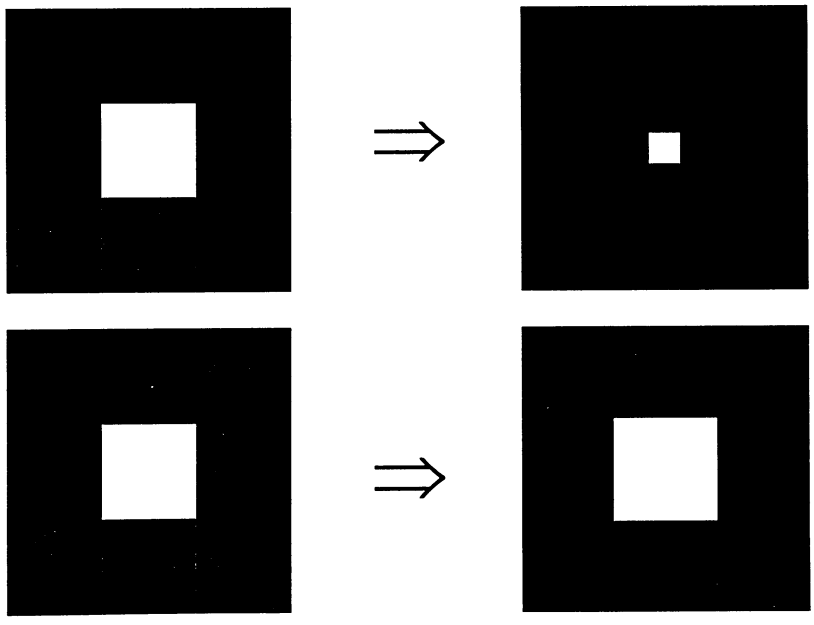


Fig32 Erosion and Dilation

Erosion removes noise by specifying its structuring element size larger than the noise shape size.

$$E = X \ominus S = \{z \mid (S)_z \subseteq X\}$$

Here X represents the input X-ray image, and S represents the structuring element of the erosion operation. The origin of the structuring element S is translated by the point z and makes sure the structuring element S is contained inside X. The result image E consists of all the point z.

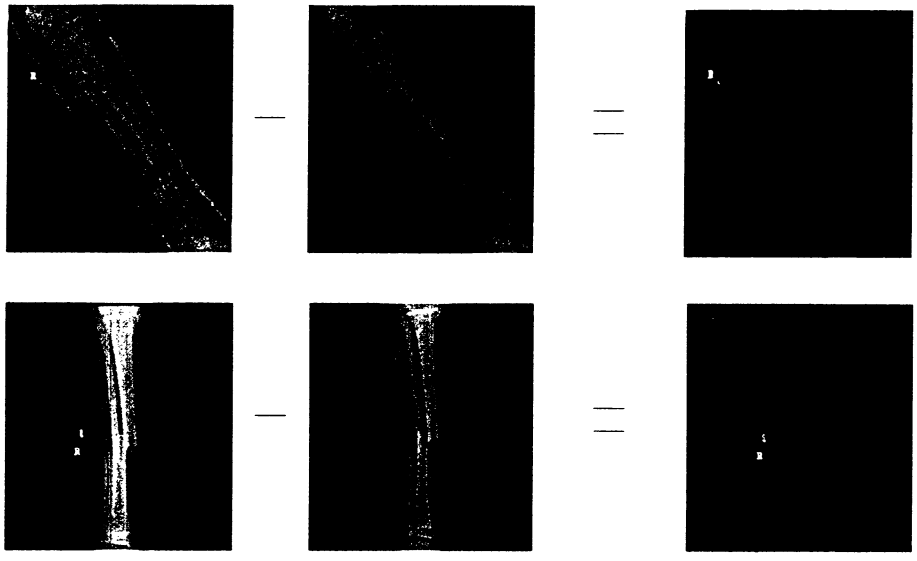


Fig33 $B = D - E$

After the erosion operation, an easy way of obtaining the edge map B is to subtract the dilation result D of the original X by the erosion result image E (see Fig32).

$$D = X \oplus S = \{z \mid [(\hat{S})_z \cap X] \subseteq X\}$$

The dilation result D is obtained by translating the origin of the reflection of the structuring element S by the point z to make sure the original X is included.

The above figure shows two edge maps for the cast X-ray image and the non-cast X-ray image respectively.

4.2 Priori Shape Model Extraction

The edge map for the non-cast X-ray image is passed to Priori Shape Model Extraction process to extract the priori shape model; the edge map for the cast X-ray image is passed to Level Set Evolving Process for segmentation.

A level set initial curve is manually placed inside the fractured bone piece, and it evolves towards the boundary of the fractured bone piece automatically. The evolving process is shown in Fig34 (see Figure below).

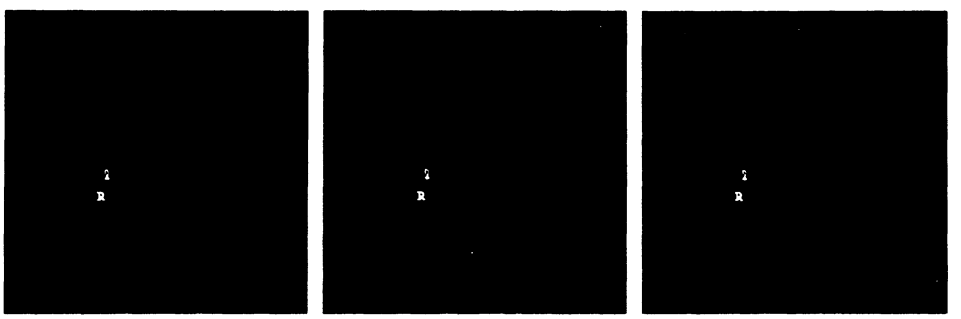


Fig 34 Priori shape model extraction

The curve evolution function is as below,

$$\frac{\partial C}{\partial t} = |\nabla C| \, div(g(I)\frac{\nabla C}{|\nabla C|})$$

$$= g(I)|\nabla C| \, div(g(I)\frac{\nabla C}{|\nabla C|}) + \nabla g(I) \cdot \nabla C$$

$$= g(I)|\nabla C| k + \nabla g(I) \cdot \nabla C$$

The energy function is given by,

$$E(C) = \overset{\min}{C(q)} \int g(|\nabla I(C(q))|) |C'(q)| \, dq$$

By minimizing the energy *E(C)*, the curve C will be stabilized on the boundaries of objects in the image. The image gradient is often given by

$$g(|\nabla I|) = \frac{1}{1+|\nabla I|^2}$$

The final blue curve is the extracted priori shape model and is saved to a .txt file. The extracted priori shape model is then loaded and passed to the Level Set Evolving Process together with the previously obtained edge map for the cast X-ray image (see figure below).

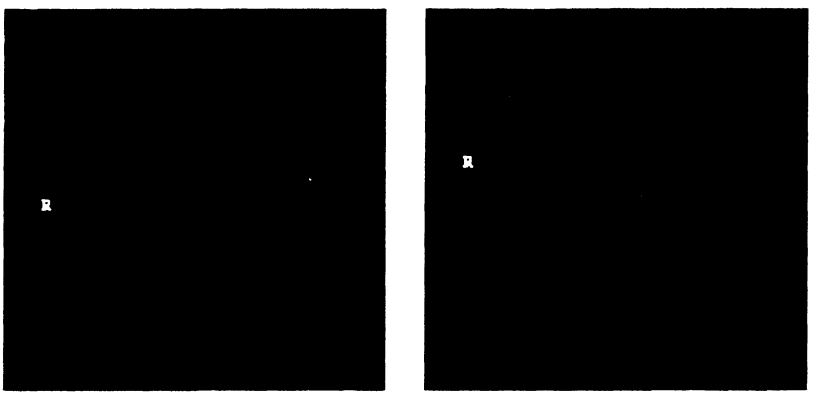


Fig35 Priori shape model loading

The right figure above shows the initial curve is placed inside the fractured bone piece and the priori shape then adjusts its centroid to the position where the initial curve is placed.

4.3 Level set evolving process with priori shape model

The initial curve evolves based on the level set evolution function that a priori shape model is embedded. The energy minimization function with priori shape model is changed as,

$$\min_{C(q)} \int \{g(|\nabla I(C(q))|) + \frac{\lambda}{2} d^2(sRC(q)+T)\} |C'(q)| dq$$

Here $\lambda > 0$ is a constant, and d is the distance from a point (x,y) to the priori shape model . The minimization problem now can be solved by finding steady state solutions to the following system:

$$\frac{\partial C}{\partial t} = -vn, C(0,q) = C_0(q),$$
$$\frac{\partial \mu}{\partial t} = -\lambda \int d\nabla d \cdot RC \mid C'(q) \mid dq, \mu(0) = \mu_0,$$
$$\frac{\partial \theta}{\partial t} = -\lambda\mu \int d\nabla d \cdot (\frac{dR}{d\theta} C) \mid C'(q) \mid dq, \theta(0) = \theta_0,$$
$$\frac{\partial C}{\partial t} = -\lambda \int d\nabla d \cdot \mid C'(q) \mid dq, T(0) = T_0,$$
$$v = \nabla g \cdot n + gk + \lambda s(d\nabla d) \cdot (Rn) + \lambda d^2 k$$

Here n is the outward unit normal to C, and k is the curvature of the curve C.

After each evolving step, the priori shape model adjusts itself by means of rotation and translation. In rotation, the priori shape model rotates the angle equal to the angle difference between the axes of the evolving curve and priori shape model. In translation, the priori shape model adjusts its centroid to the position where the evolving curve's centroid is. Both the angles of their axes and their centroids are calculated based on the shape moment methods which are described below.

Let $C_e(x_e, y_e)$ represent the centroid of the evolving curve E and $C_m(x_m, y_m)$ represent the centroid of the priori shape model M.

$$C_e(x_e, y_e): \quad x_e = \frac{m_{10,E}}{m_{00,E}}, \quad y_e = \frac{m_{01,E}}{m_{00,E}}$$

$$C_m(x_m, y_m): \quad x_m = \frac{m_{10,M}}{m_{00,M}}, \quad y_m = \frac{m_{01,M}}{m_{00,M}}$$

The $m_{00,E}$, $m_{01,E}$ and $m_{10,E}$ represent the (0,0), (0,1) and (1,0) moment of shape E (the evolving curve). The $m_{00,M}$, $m_{01,M}$ and $m_{10,M}$ represent the (0,0), (0,1) and (1,0) moment of shape M (the priori shape model).

The axis of shape S is calculated based on the central moment $\mu_{(p,q),S}$ of the shape. This can be done by moving the origin of the coordinate system to the center of the shape.

Let θ_E and θ_M represent the shape orientations of the evolving curve and the priori shape model respectively.

They are given by,

$$\mu_{11,E} = \frac{m_{01,E} m_{02,E}}{m_{00,E}}, \mu_{02,E} = \frac{m_{01,E}^2}{m_{00,E}}, \mu_{20,E} = \frac{m_{10,E}^2}{m_{00,E}}$$

$$\theta_E = \frac{1}{2} \tan^{-1} \frac{2\mu_{11,E}}{\mu_{20,E} - \mu_{02,E}}$$

$$\mu_{11,M} = \frac{m_{01,M} m_{02,M}}{m_{00,M}}, \mu_{02,M} = \frac{m_{01,M}^2}{m_{00,M}}, \mu_{20,M} = \frac{m_{10,M}^2}{m_{00,M}}$$

$$\theta_M = \frac{1}{2} \tan^{-1} \frac{2\mu_{11,M}}{\mu_{20,M} - \mu_{02,M}}$$

Here $\mu_{11,E}$, $\mu_{20,E}$ and $\mu_{02,E}$ represent the (1,1), (2,0) and (0,2) central moments of the evolving curve E; $\mu_{11,M}$, $\mu_{20,M}$ and $\mu_{02,M}$ represent the (1,1), (2,0) and (0,2) central moments of the priori shape model M.

Let M_{new} represent the rotated and translated priori shape model. The rotation and translation process can be expressed as below,

$$M_{new} = MRs + T$$

$$R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \quad \theta = \theta_M - \theta_E$$

$$T = [x_d \quad y_d], \quad x_d = x_m - x_e, y_d = y_m - y_e$$

Here s is the scaling factor for the priori shape model. For a point (x,y) on the priori shape model M, the point (x^*, y^*) on the M_{new} can be expressed by,

$$x^* = \frac{(x - x_d)\cos\theta + (y - y_d)\sin\theta}{s},$$
$$y^* = \frac{-(x - x_d)\sin\theta + (y - y_d)\cos\theta}{s}$$

The scale factor s is chosen based on min-distance between the priori shape model and the evolving curve. For two curves, priori shape model M and the evolving curve E, the min-distance algorithm is described as below.

for each point p on E,

calculate the distance between it and the centroid and sum them together,

for each point p on M,

calculate the distance between it and the centroid and sum them together,

calculate the difference between the two sums

for a range of model scaling factor s, scale the model and recalculate the distance between E and the scaled M . *if* at one scale position, the recalculated distance is the minimum, then record the position and adjust M to the new fit.

(Note: The main source codes for the evolving, translation, rotation and scaling are listed in Appendix A-D.)

Fig36 shows an example of the curve evolving process.

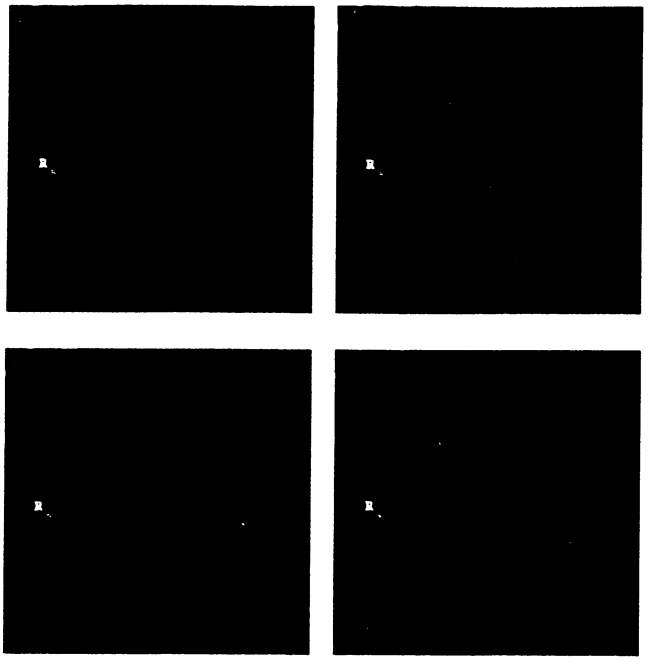


Fig 36 Curve evolving

After each evolving process, the priori shape model adjusts itself with the evolving curve. In the next iteration, the evolving process will reference the newly adjusted priori shape model. The same process goes through for another fractured bone piece.

The figures below show the two matching results and the final result.

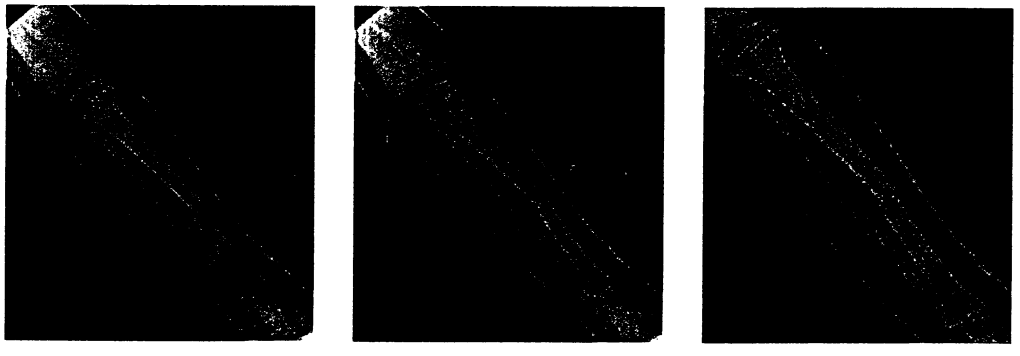
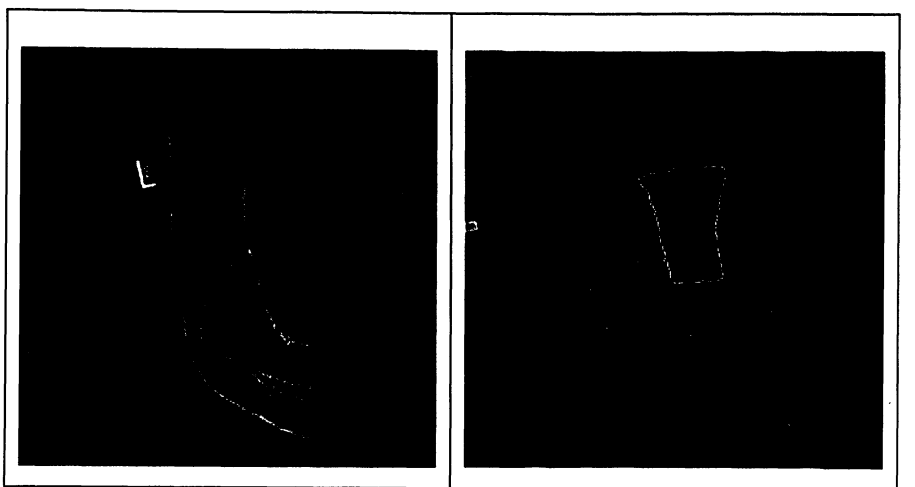


Fig37 Matching results

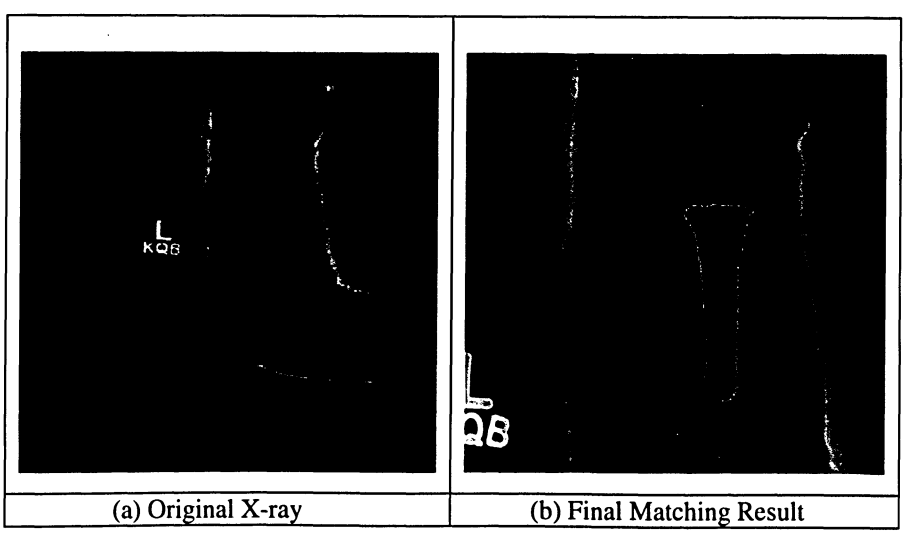
More experimental results are shown below. The blue curve represents the initial position of the priori shape model; the yellow curve represents the matched result.

Case #1



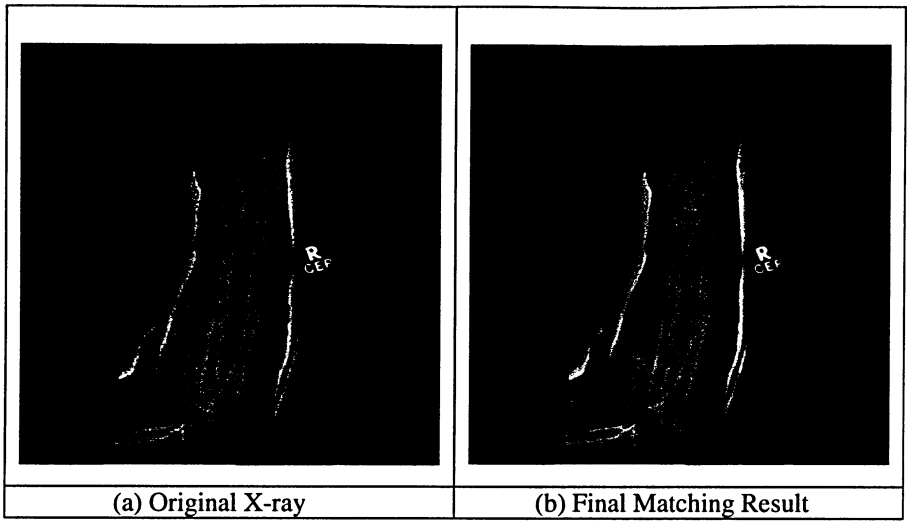
(a) Original X-ray | (b) Final Matching Result

Case #2



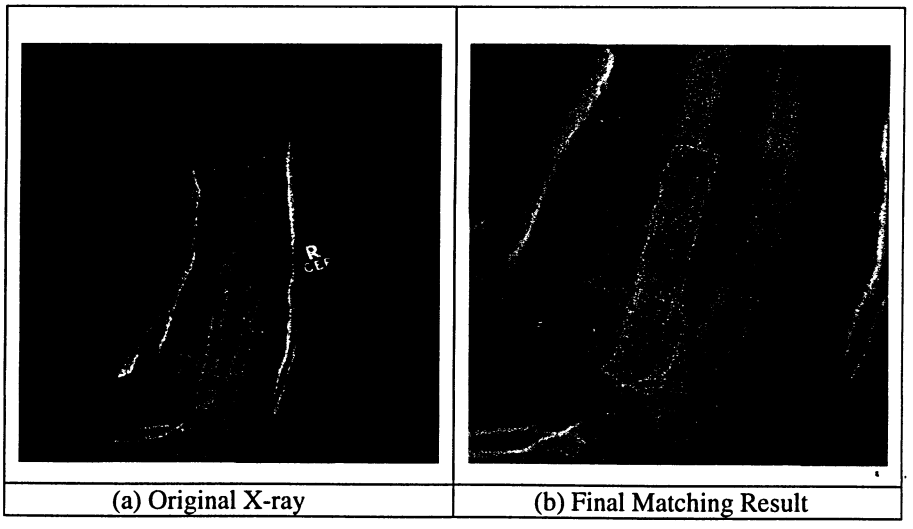
(a) Original X-ray | (b) Final Matching Result

Case #3



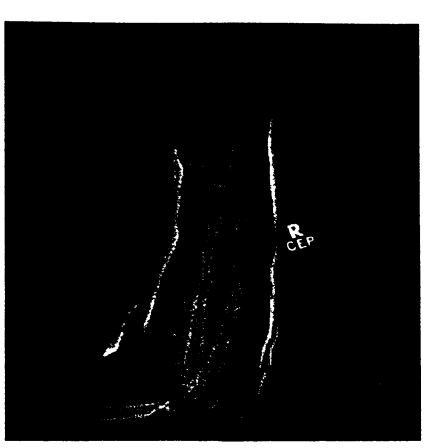
(a) Original X-ray | (b) Final Matching Result

Case #4



(a) Original X-ray | (b) Final Matching Result

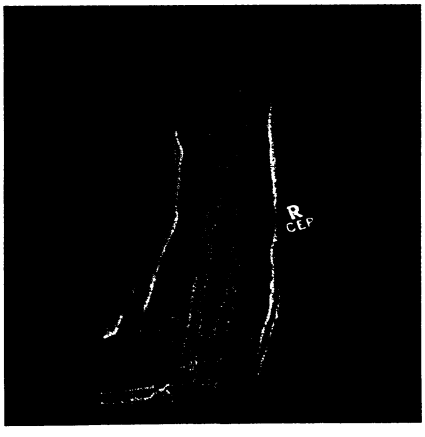
Case #5



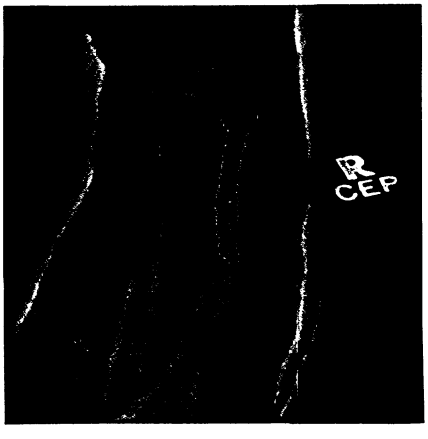
(a) Original X-ray

(b) Final Matching Result

Case #6

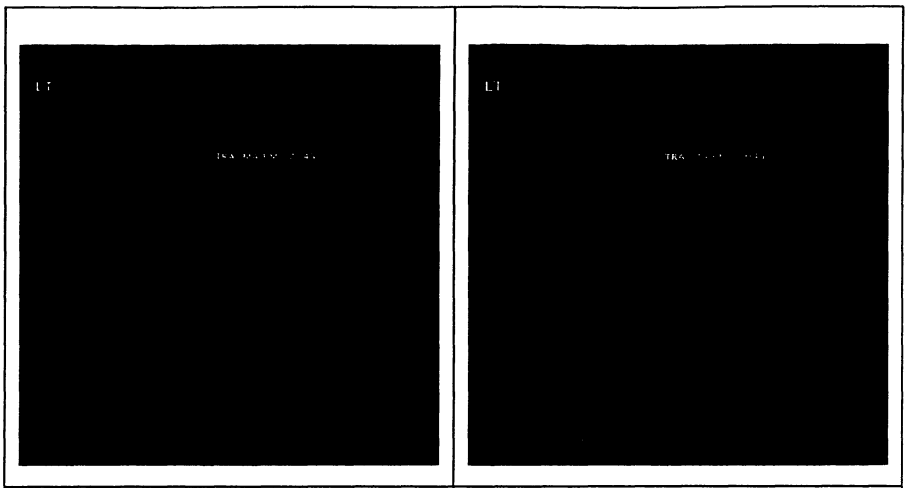


(a) Original X-ray



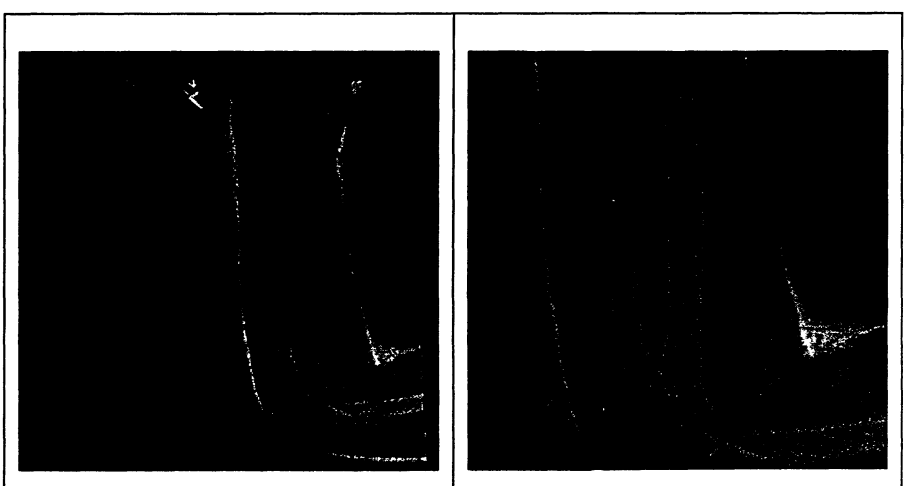
(b) Final Matching Result

Case #7



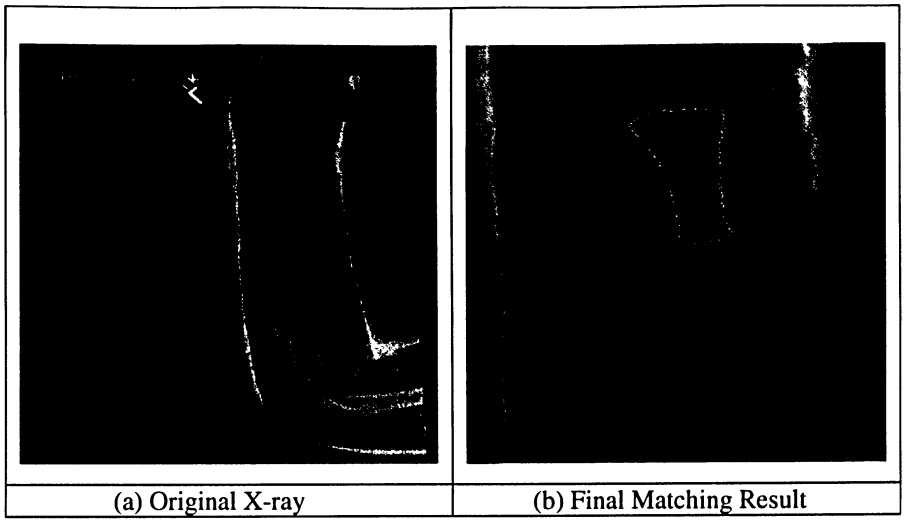
(a) Original X-ray | (b) Final Matching Result

Case #8



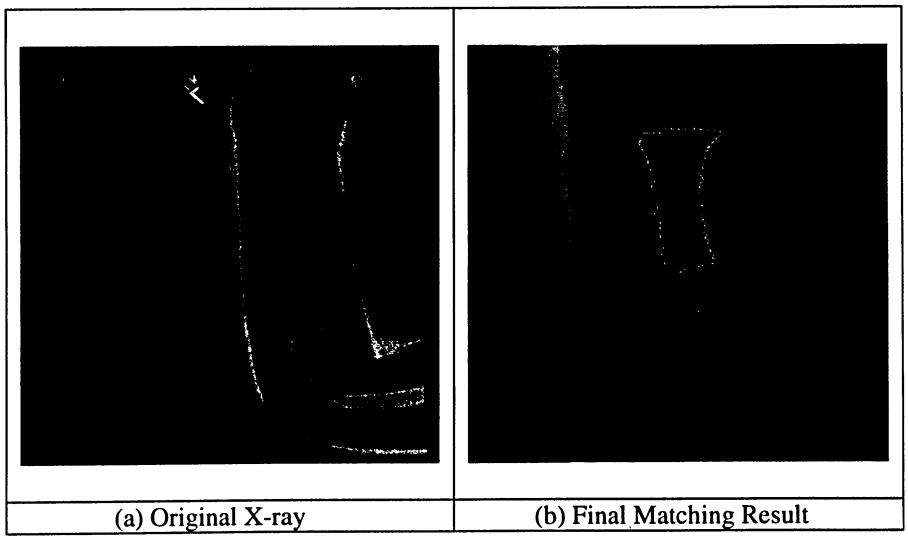
(a) Original X-ray | (b) Final Matching Result

Case #9



(a) Original X-ray | (b) Final Matching Result

Case #10



(a) Original X-ray | (b) Final Matching Result

Fig38 Other matching results

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In the research, an image segmentation method based on active contouring model was studied, which incorporated the priori shape model as the global constraint to regulate the evolving process of the active contour. The goal was to segment out the fractured bone pieces from the noisy background of the cast X-ray images and to calculate the misalignment of the fractured bone pieces. The active contouring model is implemented based on the level set method, in which the contour is propagated by evolving a time-dependent embedding function according to an appropriate partial differential equation. The curve evolves along the normal n with a speed F and stops evolving when a big gradient boundary is encountered. The difficulties of how to overcome the noise introduced by the cast material and how to prevent the active contour growing out of the area defined by the faint edges, have been solved. Morphological operations of erosion and dilation suppressed the noise by choosing the structuring element size (5x5) larger than the noise shape size. The operations also preserved the edge information very well. An edge map was obtained by subtracting the dilation result with the erosion result of the X-ray image which is better than using traditional edge detection techniques. Embedded into the evolving process, the priori shape model provided a global constraint to the evolving curve and kept it from leaking out of the weak edges due to the important shape information provided.

The overall performance of the segmentation method was satisfactory (as shown by the results in Chapter 4). However, there were non-perfect segmentation results produced by the method and they can be attributed to two reasons. The first reason is that the edge is really weak, e.g., two bones are overlapped. In this case, the area of the sharp intensity's discontinuity is large and the boundary is difficult to separate. The second reason is that only the global constraint (priori shape model) is not sufficient to guarantee the perfect segmentation result, more constraints need to be considered in above case.

5.2 Future Work

As was discussed above, the segmentation method needs to incorporate more information as constraints to the evolving process in order to perform better. For example, a possible constraint could be the intensity distributions inside the priori shape model. Another future extension is the automatic placement of the initial curve.

Bibliography

[1] S. Osher & J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations", *Journal of Computation Physics*, vol. 79, pages 12–49, 1988.
[2] D. Adalsteinsson, and J.A. Sethian, "The Fast Construction of Extension Velocities in Level Set Methods", 148, pp. 2-22, 1999.
[3] R. C. Gonzalez and R. E. Woods, "Digital Image Processing", *Second Edition*, 2002.
[4] R. M. Haralick and L.G. Shapiro, "Image segmentation techniques". *Comput. Vis. Graph. Im. Proc.*, 29:100-132, 1985.
[5] D. L. Pham, C. Xu, J. L. Prince, "A Survey of Current Methods in Medical Image Segmentation". *Annual Review of Biomedical Engineering,* Jan 19, 1998.
[6] Olabarriaga S.D., Smeulers A.W.M. "Interaction in the Segmentation of Medical Images: A Survey". *Medical Image Analysis,* 5:127-142, 2001.
[7] N. Ayache, P. Cinquin, I. Cohen, L. Cohen, F. Leitner, and O. Monga. "Segmentation of complex three dimensional medical objects: a challenge and a requirement for computer-assisted surgery planning and performance". *Computer-integrated surgery: technology and clinical applications*, pages 59-74, MIT Press, 1996.
[8] W. E. L. Grimson, G, J. Ettinger, T. Kapur, M. E. Leventon, W. M. Wells, et al. "Utilizing segmented MRI data in image-guided surgery". *Int. J. Patt. Rec. Art. Intel.*, 11:1367-1397, 1997.
[9] S. M. Larie and S. S. Abukmeil. "Brain abnormality in schizophrenia: a systematic and quantitative review of volumetric magnetic resonance imaging studies". *J. Psych.*, 172:110-120, 1998.
[10] H. Zaidi, "Medical Imaging: Current status and future perspectives". Division of Nuclear Medicine, Geneva University Hospital, 1997.
[11] N. R. Pal and S. K. Pal, "A review on image segmentation techniques". *Patt. Rec.*, 26:1277-1294, 1993.
[12] D. Marr and E. Hildreth, "Theory of Edge Detection". *Proceedings of the Royal Society London*, B207:187-217, 1980.
[13] R. M. Harlick, "Zero-Crossing of Second Directional Derivative Edge Operator". IEEE Trans. PAMI., 6:58-68, 1978.
[14] D. Ballard, "Generalising the Hough Transform". *Pattern Recognition*, 13:111-22, 1981.
[15] M. Ester, H. P. Kriegel, J. Sander, X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". *ACM Int'l. Conf. on Knowledge Discovery and Data Mining (KDD),* 1996.
[16] D.A. Forsyth, J. Ponce, "Computer Vision A Modern Approach". Prentice Hall, 2003
[17] T. McInerney and D. Terzopoulos. "Deformable models in medical image analysis: a survey", Medical Image Analysis, 1(2):91--108, 1996.
[18] M. Kass, A. Witkin, D. Terzopoulos, "Snakes: active contour models". *International Journal of Computer Vision* 1, 321—332, 1988.
[19] V. Caselles, F. Catte, T. Coll, and F. Dibos, "A geometric model for active contours". *Numerische Mathematik*, 66:1-31, 1993.

[20] J. L. Prince, Davatzikos, "Convexity Analysis of Active Contour Algorithms". *Image and Vision Computing*, 17(1):27-36, January 1999.
[21] T. J. McInerney, "Topologically Adaptable Deformable Models for Medical Image Analysis". *PhD thesis, Department of Computer Science, University of Toronto,* 1997.
[22] F. Leymarie and M. D. Levine, "Tracking Deformable Objects in the Plane using an Active Contour Model". *IEEE Trans. Patt. Anal. Mach. Intell.*, vol.15, No.6, 1993
[23] P. Fua and Y. G. Leclerc, "Model Driven Edge Detection". *Machine Vision and Applications*, vol.3, pp. 45-56, 1990.
[24] R. Bajscy, S. Kovacic, "Multiresolution Elastic Matching". *Computer Vision, Graphics, and Image Processing*, vol.46, pp. 1-21, 1989.
[25] I. Cohen, L. D. Cohen and N. Ayache, "Using Deformable Surfaces to Segment 3-D Images nd Infer Differential Structures". *CVGIP: IMAGE UNDERSTANDING*, vol.56, No.2, Setptember, pp. 242-263, 1992.
[26] Y. F. Wang and J. F. Wang, "Surface Reconstruction using Deformable Models with Interior and Boundary Constraints". *IEEE Trans. Patt. Anal. Mach. Intell.*, vol.14, No.5, pp. 572-579, 1992.
[27] T. McInerney and D. Terzopoulos, "T-snakes: Topology Adaptive Snakes". *Medical Image Analysis*, vol.4, No.2, pp.73-91, 2000.
[28] S. R. Gunn, "Dual Active Contour Models for Image Feature Extraction". PhD thesis, University of Southampton, May 1996.
[29] M. Droske, B. Meyer, M. Rumpf, and C. Schaller, "An adaptive level set method for medical image segmentation". *Proc. of the Annual Symposium on Information Processing in Medical Imaging*, Springer, Lecture Notes Computer Science, 2001.
[30] H. Q. Lu, T. X. Zhang, "An improved image segmentation approach based on level set and mathematical morphology". *Third International Symposium on Multispectral Image Processing and Pattern Recognition*, pp. 851-854, September 2003.
[31] E. Debreuve, M. Barlaud, G. Aubert, I. Laurette, and J. Darcourt, "Space-time segmentation using level set active contours applied to myocardial gated SPECT". *IEEE TRANSACTIONS ON MEDICAL IMAGING* , vol. 20 , pp. 643 -659 , 2001.
[32] L. Staib and L. Duncan, "Boundary finding with parametrically deformable models", *IEEE transactions on pattern analysis and machine intelligence*, 14: 1061-1075, 1992.
[33] Y. Wang and L. Staib, "Boundary finding with correspondence using statistical shape models". *Proceedings from IEEE Conference on Computer Vision and Pattern Recognition*, 338 – 345, 1998.
[34] M. E. Lenventon, W-E. L. Grimson and O. Faugeras, "Statistical shape influence in geodesic active contours". *Proceedings from IEEE Conference on Computer Vision and Pattern Recognition*, 316 – 323, 2000.
[35] Y. Chen, S. Thiruvenkadam, D. H. Tagare, et al., "On the incorporation of shape priors into geometric active contours". *Proceedings from IEEE Workshop on Variational and Level set method*, 145-152, 2001
[36] M. Donnelley, G. Knowles, "Computer aided long bone fracture detection", *Signal Processing and Applications, Proceedings of the Eighth International Symposium on*, pp. 175-178, August 28-31, 2005.
[37] T. P. Tian, Y. Chen, W. K. Leow, W. Hsu, T. S. Howe and M. A. Png, "Computing neck-shaft angle of femur for x-ray fracture detection", in Proc. Int. Conf. on Computer Analysis of Images and Patterns (LNCS 2756), pp. 82-89, 2003.

[38] J. Canny, "A Computational Approach to Edge Detection". *IEEE Trans. PAMI.*, 8:679-698, 1986.
[39] J. Pluim, J. Maintz, M. Viergever, "Mutual information based registration of medical images: a survey". *Medical Imaging*, 22 (8), pp. 986-1004, 2003.
[40] R. V. L. Hartley, "Transmission of information". *Bell Syst. Tech. J.*, vol. 7, pp. 535-563, 1928.
[41] C.E. Shannon, "A mathematical theory of communication", *Bell Syst. Tech. J.*, vol. 27, pp. 379-423/623-656, 1948.
[42] A. Collignon, "Multi-modality medical image registration by maximization of mutual information", Ph.D. dissertation, Catholic Univ. Leuven, Leuven, Belgium, 1998.
[43] P. A. Viola, "Alignment by maximization of mutual information", Ph.D. dissertation, Massachusetts Inst. Technol., Boston, MA, 1995.
[44] D. L. Chopp. "Computing minimal surfaces via level set curvature flow", *Journal of Computational Physics*, 106: pp. 77-91, 1993.
[45] M. Sussman, P. Smereka, S. Osher. "A level set method for computing solutions to incompressible two-phase flow". *Journal of Computational Physics*, 114: pp. 146-159, 1994.
[46] J. A. Sethian, "Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science". Second Edition. Cambridge: Cambridge University Press, 1999.
[47] J. G. Leu, "Computing a shape's moments from its boundary". *Pattern Recognition*, Vol.24, No.10, pp.949--957, 1991
[48] Y. Jia, Y. Jiang, "Active Contour Model with Shape Constraints for Bone Fracture Detection". *Computer Graphics, Imaging and Visualisation*, Sydney, Australia, Jul. 26-28, 2006

Appendices

Appendix A:

Curve Evolving Source Code

```
void CLoadMedBoneImgView::MS_evolving(int i) {
        CPixelCurve tempAcurve;
        CPixelCurve tempBcurve;
        CPixelCurve tempCcurve;
        CPixelCurve temp;

        CPixelCurve tCurve;
        int total, totalc;
        CPoint p;
        float CurvecentroidX, CurvecentroidY;
        int ii, sumX = 0, sumY = 0;


        total = m_currentCurve.curve.GetSize();
        for (ii = 0; ii < total; ii++) {
                sumX += m_currentCurve.curve[ii].x;
                sumY += m_currentCurve.curve[ii].y;
        }

        CurvecentroidX = (float)sumX / (float)total;
        CurvecentroidY = (float)sumY / (float)total;
        tempCcurve.Copy(cCurve);
        totalc = cCurve.GetSize();
        cCurve.RemoveAll();

        for (ii = 0; ii < totalc; ii++) {
                p.x = tempCcurve[ii].x - (int)(CurvecentroidX-m_modelP.centroidX);
                p.y = tempCcurve[ii].y - (int)(CurvecentroidY-m_modelP.centroidY);
                temp.Add(p);
        }
        cCurve.Copy(temp);//(tempCcurve);
        m_modelP.curve.RemoveAll();
        m_modelP.curve.Copy(temp);//(tempCcurve);
        tempCcurve.RemoveAll();

        for (ii = 0; ii < total; ii++) {
                p.x = aCurve[ii].x;
                p.y = aCurve[ii].y;
                tempAcurve.Add(p);
        }
        total = tempAcurve.GetSize();
        tempAcurve.Copy(m_currentCurve.curve);
        for (int k = 1; k <= i; k++) {
                clear_MaskImg();
                tempBcurve.Copy(traceBorder(tempAcurve, 1));
// New Code Add here...
                int tv = 3;
                set_maskImage_to_T(tempBcurve,tv);
                tempCcurve.RemoveAll();
                tempCcurve.Copy(traceBorder(tempBcurve, 1));
                tv ++;
                set_maskImage_to_T(tempCcurve,tv);
```

```
                tempBcurve.RemoveAll();
                tempBcurve.Copy(traceBorder(tempCcurve, 1));

                tCurve.RemoveAll();
                tCurve.Copy(
                        Evolve_to_ModelShape(tempAcurve));

                clear_MaskImg();
                tempBcurve.RemoveAll();
                tempBcurve.Copy(traceBorder(tCurve, 1));
// minpath algorithm start here...
                tv = 1;
                set_maskImage_to_T(tempBcurve,tv);
                tempCcurve.RemoveAll();
                tempCcurve.Copy(traceBorder(tempBcurve, 1));
                tempBcurve.RemoveAll();
                tv ++;
                set_maskImage_to_T(tempCcurve,tv);
                tempBcurve.Copy(traceBorder(tempCcurve, 1));

                // make the mask on the mask image...
                tCurve.RemoveAll();
                tCurve.Copy(minPathLine());

                // curve post process start here...
                tempBcurve.RemoveAll();
                tempBcurve.Copy(traceBorder(tCurve, 1));
                tv = 250;
                clear_MaskImg();

                set_maskImage_to_T(tempBcurve,tv);
                tempAcurve.RemoveAll();

                tempAcurve.Copy(processResultCurve(tCurve));
                tCurve.RemoveAll();
                tCurve.Copy(tempAcurve);
        }

        aCurve.RemoveAll();
        aCurve.Copy(cCurve);
        cCurve.RemoveAll();
        cCurve.Copy(tCurve);

        m_currentCurve.curve.RemoveAll();
        m_currentCurve.curve.Copy(tCurve);
        GetDocument()->UpdateAllViews(0, 0, NULL);
}
```

Appendix B:

Model Translation Source Code

```
void CLoadMedBoneImgView::Model_Translation(void)
{
        int i,totalm,totalc;
        CPoint p;
        CPixelCurve tempCurveM,tempCurveC;
        tempCurveM.Copy(m_modelP.curve);
        tempCurveC.Copy(m_currentCurve.curve);
        totalm = tempCurveM.GetSize();
        totalc = tempCurveC.GetSize();
        CPixelCurve tempCurve;

        double sumXm = 0,sumYm = 0, sumXc = 0, sumYc = 0;
        float dx,dy;

        dx = m_modelP.centroidX - m_currentCurve.centroidX;
        dy = m_modelP.centroidY - m_currentCurve.centroidY;

        m_modelP.centroidX = m_currentCurve.centroidX;
        m_modelP.centroidY = m_currentCurve.centroidY;

        for (i = 0; i < totalm;i++){
                p = m_modelP.curve[i];
                p.x -=(long) dx;
                p.y -=(long) dy;
                tempCurve.Add(p);
        }
        m_modelP.curve.RemoveAll();
        m_modelP.curve.Copy(tempCurve);
        cCurve.RemoveAll();
        cCurve.Copy(m_modelP.curve);
}
```

Appendix C:

Model Rotation Source Code

```
void CLoadMedBoneImgView::Model_Rotation_OnTheta(double theta)
{
        int i;
        double x,y;
        CPoint p;
        CPixelCurve tempCurve;
        double stheta, ctheta, temptheta;
        double dx,dy;
        int total;

        total = m_modelP.curve.GetSize();
        temptheta = 10*theta;

        stheta = sin(temptheta);
        ctheta = cos(temptheta);

        for (i = 0; i < total; i++){
                p = m_modelP.curve[i];

                dx = (p.x - m_currentCurve.centroidX);
                dy = (p.y - m_currentCurve.centroidY);
                x = (dx*ctheta - dy*stheta) + m_currentCurve.centroidX;
                y = (dx*stheta + dy*ctheta) + m_currentCurve.centroidY;
                p.x = (long)x;
                p.y = (long)y;
                tempCurve.Add(p);
        }
        m_modelP.curve.RemoveAll();
        m_modelP.curve.Copy(tempCurve);


}
```

Appendix D:

Model Scaling Source Code

```
double CLoadMedBoneImgView::Model_Scaling(double dAmin)
{
        int i, totalm, totalc, total;
        CPixelCurve tempCurve,tempCurveM, tempCurveC,tempMin;
        double i_minSx,i_minSy, d,dmin,iSx,iSy;
        double ic,im;
        double icR[360];
        CPoint p, p1;
        double x1, y1, x, y, dx, dy, ddc, ddm;
        double cx, cy, mx, my;
        double sumD;

        tempCurveM.Copy(m_modelP.curve);
        totalm = tempCurveM.GetSize();
        tempCurveC.Copy(m_currentCurve.curve);
        totalc = tempCurveC.GetSize();
        i_minSx = 1.0;
        i_minSy = 1.0;
        ic = (double)totalc/360.0;
        im = (double)totalm/360.0;
        total = 360;

        cx = (double)m_currentCurve.centroidX;
        cy = (double)m_currentCurve.centroidY;
        mx = m_modelP.centroidX;
        my = m_modelP.centroidY;

        sumD = 0;
        for (i = 0; i < 360; i++){
                //for curve...
                p1 = tempCurveC[((int)(i*ic))%totalc];
                x = (double)p1.x;
                y = (double)p1.y;
                dx = x - cx;
                dy = y - cy;
                ddc = sqrt(dx*dx + dy*dy);
                icR[i] = ddc;
                // for model...
                p1 = tempCurveM[((int)(i*im))%totalm];
                x = (double)p1.x;
                y = (double)p1.y;
                dx = x - mx;
                dy = y - my;
                ddm = sqrt(dx*dx + dy*dy);
                sumD += fabs(ddm - ddc);
        }
        dmin = sumD;

        for (iSy = 0.98;iSy < 1.08;iSy += 0.002){
                for (iSx = 0.98;iSx < 1.08;iSx += 0.002){
                        // find minSum on scales...
                        sumD = 0;
                        for (i = 0; i < total; i++){
                                p = tempCurveM[((int)(i*im))%totalm];
                                x1 = (double)p.x;
```

```
                        y1 = (double)p.y;
                        x = iSx*x1;
                        y = iSy*y1;
                        dx = x - (double)mx;
                        dy = y - (double)my;
                        ddm = sqrt(dx*dx + dy*dy);
                        sumD += fabs(ddm - icR[i]);
                }
                d = sumD;
                if (d < dmin){
                        dmin = d;
                        i_minSx = iSx;
                        i_minSy = iSy;
                }
        }
}

// change the scale to the new fit...
if ((i_minSx != 1)||(i_minSy != 1)){
        for (i = 0; i < totalm; i++){
                p = tempCurveM[i];
                x1 = (double)p.x;
                y1 = (double)p.y;
                x = (x1*i_minSx);
                y = (y1*i_minSy);
                p.x = (int)x;
                p.y = (int)y;
                tempCurve.Add(p);
        }
        // find the centroid aFTER THE scales adjusted...
        finding_newModel_centroid(tempCurve);
        int t_diffx, t_diffy;
        t_diffx = (int)(m_m_centroidX - mx);
        t_diffy = (int)(m_m_centroidY - my);
        tempCurveM.RemoveAll();
        totalm = tempCurve.GetSize();
        for (i = 0; i < totalm; i++){
                p = tempCurve[i];
                x = (double)(p.x + t_diffx);
                y = (double)(p.y + t_diffy);
                p.x = (int)x;
                p.y = (int)y;
                tempCurveM.Add(p);
        }
        // translate the scale adjusted model to the original centroid...

// finding the dmin from the tempCurve...
        d = 0;

        for (i = 0; i < total; i++){
                p = tempCurveM[((int)(i*im))%totalm];
                p1 = tempCurveC[((int)(i*ic))%totalc];
                x = (double)p.x;
                y = (double)p.y;
                dx = x - (double)p1.x;
                dy = y - (double)p1.y;
                d += sqrt(dx*dx + dy*dy);
        }
        if (d < dAmin){        //update the model energy...
```

```
                    m_modelP.curve.RemoveAll();
                    m_modelP.curve.Copy(tempCurveM);
                    dmin = d;
                    return(dmin);
                }
        }
        return (dAmin);
}
```