Department of Mathematics
University of Bergen

Master Thesis in Applied and Computational Mathematics

Supervised Continuous Max-flow for Segmentation of Images with Intensity Inhomogeneities

Junjie Guo

supervised by
Prof.Xue-Cheng Tai

May 26, 2016

Acknowledgements

The thesis represents two years of hard work away from home. In these two years I have met many remarkable people, mathematic wise and personal, to whom I would like to express my deep gratitude.

I would like to thank my supervisor, Professor Xue-Cheng Tai, for guiding my way with good mathematical ideas, and for encouraging me to go further.

I would like to thank my friend, Shi Yan, for his support given, for his prompt answers to all my puzzles, and for his constructive suggestions on my writing of the thesis.

I would also like to thank Associate Professor Wei Wang at the department of chemistry for his support and guidance in life, and also for revising the thesis.

I am mostly grateful for all the support and encouragement received from my parents through the years. I thank them for their altruistic love and financial support.

Junjie Guo

Bergen, May 2016

Abstract

In this thesis we propose a regional adaptive active contour model for segmenting images with intensity inhomogeneities effectively. This model includes a regularization term, a global energy term and a local energy term. The regularization term is used to control the length of the border of the characteristic function. At the early stage, the global term which is larger than the local term can give us a good initial segmentation of the image quickly. In the later phase of the process, the local energy term dominating the whole function energy can localize the precise position of the object of interest in image with intensity inhomogeneities. An algorithm based on supervised continuous max-flow method is developed to solve this problem robustly and validly. A large number of experiments are presented to show how this proposed method behaves on different images.

Contents

1 Introduction 1

2 Preliminaries 3

2.1 Image processing basics 3

2.1.1 Digital images 3

2.1.2 Mathematical operations used in digital image processing 7

2.1.3 Image segmentation 9

2.2 Optimisation theory 10

2.2.1 Gateaux differential 10

2.2.2 Unconstrained optimization 10

2.2.3 Constrained optimization 13

3 Max-flow and min-cut 16

3.1 Min-cut 16

3.2 Max-flow 17

3.3 Max-flow min-cut theorem 18

3.4 Continuous max-flow and min-cut 19

3.5 Supervised continuous max-flow and min-cut 21

4 Segmentation models 24

4.1 Chan-Vese segmentation model 24

4.1.1 Chan-Vese segmentation model without priori given information 24

4.1.2 Chan-Vese segmentation model with priori given information 28

4.2 Local region-based Chan-Vese model 29

4.3 Regional adaptive active contour model 30

5 The proposed method 31

5.1 Minimising w.r.t. u 32

5.2 Minimising w.r.t. c 36

5.3 Updating d 37
5.4 Updating λ and μ 37
5.5 The main steps of the algorithm 38
5.6 Implementation issues 39
5.6.1 Approximating the derivative 39
5.6.2 Approximating the divergence 40
5.6.3 Approximating the integral 40
5.6.4 Smoothing the two indicator functions 41

6 Experimental results **42**
6.1 Number of max-flow iterations 42
6.2 Trials on some images 44

7 Summary and conclusion **53**

Chapter 1

Introduction

Image processing is a special kind of signal processing in which the input is an image or a series of images and the output is an image or features related with the image. The purpose of image processing is to get an enhanced image or to extract meaningful information from the image by performing suitable operations on it. Nowadays, image processing has become a rapidly growing technology, which benefits from the development of computer. It forms core research area within industry, meteorology, law enforcement, defence and so on. For example we can found image processing being applied in inspecting products missed in a factory, predicting weather conditions, recognizing the face of criminals automatically and developing unmanned planes.

For many image analysis problems, the first step to deal with the research image is to separate the object of interest from its background in order to make the problem easier. Thus image segmentation has always been a fundamental task in image processing and its result has a great influence on the rest of the process. Because of the complexity and importance of image segmentation, an increasing number of researches have been done in the last decade. Image segmentation process is usually needed in fields such as machine vision, medical imaging, object detection, traffic control problem and many others. For example, so as to automatically recognize the face of criminal, we need to isolate the face in the given image and compare the result with the face which we have stored in our database.

The active contour model [1] which has been proven successful in solving image segmentation problems in recent years can be divided into edge-based model [2] and region-based model [3]. The active contour model based on edge which use gradient of image is not able to segment the target with blur edges or uneven gray correctly. The area based active contour model which ignores edges completely can work well for those images in which edges can not be found by the gradient information. One of the most famous models is the Chan-Vese (CV) model [4] which can optimally fits a two-phase piecewise constant model to a given image. The CV model is an effective method for segmenting images made of two regions with distinct mean of pixel intensity. We should note that the CV model is based on the assumption that the intensities of each region are uniform. However, most of the images from the complex real word can not hold this assumption, which holds back its applications.

To solve the problem caused by the intensity inhomogeneities, Shigang Liu and Yali Peng [5] proposed a new region based active contour model, named local region-based Chan-Vese (LRCV) model. By considering the local information, LRCV model can segment an image with intensity inhomogeneity effectively. However, to some extent this model is sensitive to the initial contour, which limits its practical applications.

In order to improve the robustness to initialization, a new model named regional adaptive active contour (RAAC) model is presented in [6] by Xing Hui, Yali Peng, et all. In this model, they considered to combine the local intensity information and the global intensity information. At the beginning, the global energy term overwhelming gives us a rough but good and quick initial segmentation. In the later stage of evolution, the exact location of the object of interest can be detected by the local energy term which is dominating the whole function energy.

For solving the RAAC model, the traditional level set method [7] is used in [6]. However, we find that one sub-problem can be formulated as a continuous min-cut problem. The purpose of this work is to use the supervised continuous max-flow method to solve this sub-problem and define the global weight function and the local weight function in LRCV model specifically.

The rest of this thesis is organised as follows: In chapter 2 we recall some basic knowledge about image processing and optimisation theory. Then, we explain the supervised continuous max-flow method in chapter 3. Chapter 4 details several segmentation models. The short discussion on implementation is given in chapter 5 followed by a variety of results in chapter 6.

Chapter 2

Preliminaries

In this chapter we review some basic knowledge which establish the foundation for the remainder of this thesis. In the first section, the definition of digital image and some image processing tools are recalled before introducing the segmentation task. The second section about optimisation theory includes three parts: introduction of Gateaux differential, unconstrained optimization and constrained optimization.

2.1 Image processing basics

2.1.1 Digital images

An image can be represented by a two-dimensional function, $f(x, y)$, where x and y are spatial coordinates, and the value of $f(x, y)$ at any pair of coordinates (x, y) is called the intensity or gray level of the image at that location. We call the image where the coordinates x, y and the function value of f are all finite, discrete quantities a digital image[8].

There are many ways to acquire images [8], essentially, the objective is to generate digital images from sensed data which is a continuous voltage waveform. So we can create a digital image after converting the continuous sensed data into digital form. This task can be completed by two processes: sampling and quantization. Sampling is to digitize the coordinate values while quantization is to digitize the amplitude values.

Suppose that the continuous image is sampled into a 2-D array, $f(x, y)$, consisting of M rows and N columns, where (x, y) are discrete coordinates. For notational clarity and convenience, we use integer values for these discrete coordinates [8]: $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$. For example, $f(0, 0)$ denotes the value of the digital image at the origin and $f(1, 0)$ is the value of the next coordinate along the first column. The number of the rows and the columns is related to fidelity of the digital image. The larger the number is, the smaller the difference between the digital image and the original image is. The spatial domain is defined by the section of the real plane spanned by the coordinates of an image. We refer to x and y as spatial variables or spatial coordinates.

There are three basic ways [8] to describe $f(x, y)$. one way is to represent it as a plot of function with two axes determining spatial location and the third axis being the values of f as a function of the two spatial variables x and y. Sometimes, the images are too detailed and hard for us to interpret from its plot. Showing $f(x, y)$ as it would appear on a monitor or photograph is the second way to represent $f(x, y)$. Here the intensity of each point is proportional to the value of f at that point. The third representation is just to display the intensity values as an matrix. The representation of an $M \times N$ numerical array by the third way can be written as [8]

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \tag{2.1}$$

This kind of representation can be used for processing and algorithm development. we call each element of the matrix above an *image element, picture element, pixel, or pel.* Conventionally, the origin of a digital image is at the top left, with the positive *x-axis* extending downward and the positive *y-axis* extending to the right. This is based on the fact that many image displays sweep an image starting at the top left and moving to the right one row at a time. Another more important reason is that the first element of a matrix is at the top left of the array, so putting the origin of the digital image at that location makes sense mathematically.

Sometimes, it is useful to express sampling and quantization in more formal mathematical terms. Let Z and R denote the set of integers and the set of real numbers, respectively. The process of sampling can be regarded as partitioning the xy-plane into a grid where the coordinates of the center of each cell is a pair of elements from the Cartesian product Z^2 [8], which is the set of all orded pairs of elements (z_i, z_j), with z_i and z_j being integers from Z. Hence, if (x, y) are chosen from Z^2 and f is a function that assigns an intensity value to each distinct pair of coordinates (x, y), we can call $f(x, y)$ a digital image.

Digitising the function values, quantisation, is done by picking equally spaced values along intensity scale and assigning one of these to each point depending on whose sensed value is closest to discrete quantity [8]. Due to storage and quantizing hardware considerations, we chose an integer power of 2: 2^k as the number of intensity levels. We call an image with 2^k intensity levels a k-bit image. For example, an image with 256 possible discrete intensity values is called an 8-bit image. Such an image has intensity levels that lie in the interval [0,255]. Sometimes, we normalise the intensity to the interval [0,1] just by dividing the original discrete number by 255. Here, 0 represents black and 1 represents white. The range of values spanned by gray scale is referred to informally as the *dynamic range*. The *dynamic range* of an imaging system is defined as the ratio of the maximum measurable intensity to the minimum detectable intensity level in the system.

Now we use an example to show how a digital image is expressed by a matrix. Here, we regard the part which has the same size with the white

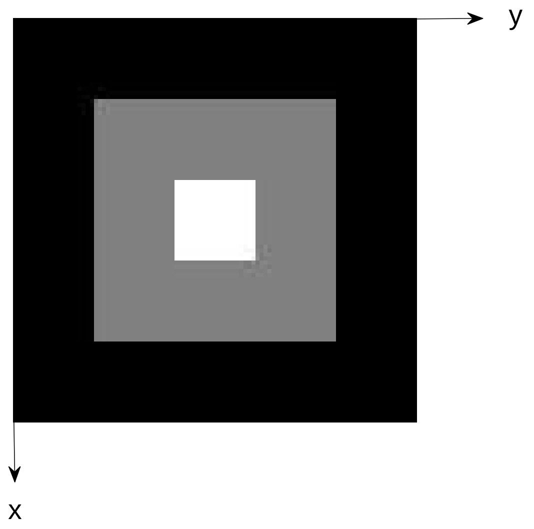
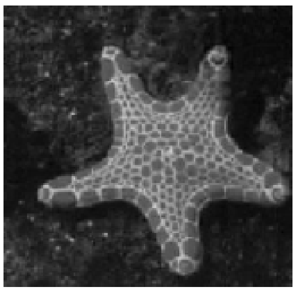


Figure 2.1: An digital image in x-y plane

part in the center as one pixel. The corresponding 2-D array of figure 2.1 is following:

$$f(x,y) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0.5 & 1 & 0.5 & 0 \\ 0 & 0.5 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.2}$$


(a)

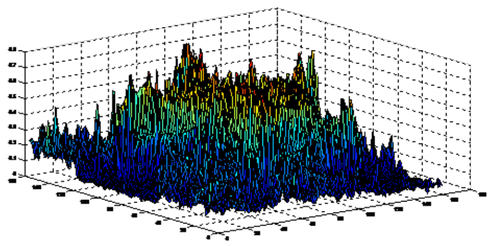
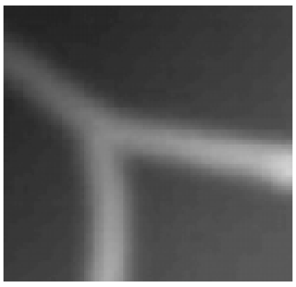

(b)

Figure 2.2: The image with homogeneous intensity and its corresponding gray scale 3-D plot.


(a)

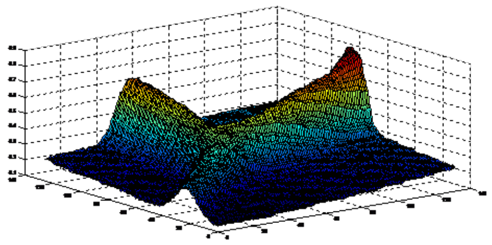

(b)

Figure 2.3: The image with inhomogeneous intensity and its corresponding gray scale 3-D plot.

Images with homogeneous intensity

A homogeneous intensity image has a good property which is that the variation of the intensity in same part of the image is low. The difference between the intensities of two pixels in the same region is small no matter how far these two pixels is. The plot with higher values in figure 2.2(b) represents the starfish in figure 2.2(a) and the part left corresponds to its surroundings. As we can see from the figure 2.2(b), the plot inside the starfish and the plot outside the starfish seem to be two flat plains.

Images with inhomogeneous intensity

For images with inhomogeneous intensity, the variation of the intensity inside at least one region is very high, which is shown in figure 2.3. Even though two pixels are in the same region, their difference could be very large if they are far away from each other. From figure 2.3(b), we can see that the vessel looks like a bumpy mountain ridge.

2.1.2 Mathematical operations used in digital image processing

Arithmetic operations

Even though an digital image is represented by a matrix, there are many situations in which arithmetic operations should be performed. The arithmetic operations are carried out between corresponding pixel pairs. For example [8], we consider the following 2×2 images:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

The array product of these two images is given by

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} \\ a_{21}b_{21} & a_{22}b_{22} \end{bmatrix}$$

On the other hand, the matrix product is

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

In image segmentation, array operations play a very important role. For example, if our goal is to extract regions of interest from an image, we can just multiply the image with a binary mask image which consists of ones in the regions which we are interested in and zeros in the remaining parts of the image. Array operations are also applied in other areas including: noise reduction and shading correction. We assume array operations throughout this thesis, unless stated otherwise. When we refer to dividing an image by another, we mean that the division is between corresponding pixel pairs; when we refer to raising an image to a power, we mean that each individual pixel is raised to that power, and so on.

Spatial filtering

Spatial filtering can be used in a broad spectrum of applications. In fact, we borrow the name *filter* from frequency domain processing, where filtering refers to accepting or rejecting certain frequency components [8]. However spatial filters are more versatile, because they can be used for nonlinear filtering, something we cannot do in the frequency domain.

A spatial filter consists of (1) a *neighborhood*, and (2) a *predefined operation* that is operated on the image pixels encompassed by the neighborhood.

We can get the new filtered image by moving the center of the spatial filter of the neighborhood from pixel to pixel in the input image and doing the predefined operation to generate the result at the coordinate of the neighborhood center. If the predefined operation is linear, we call the filter a *linear spatial filter*. Here, we only talk about linear filtering.

Correlation and *convolution* are two closed related concepts in linear spatial filtering. Correlation is the process of moving a filter mask over the input image and computing the sum of products at each location. Convolution is the same except that the filter is rotated by 180°.

Correlation of a filter $w(x, y)$ of size $m \times n$ with an image $f(x, y)$, denoted as $g(x, y)$ is given by:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

where $m = 2a + 1, n = 2b + 1$, and x and y are varied so that each pixel in w visits every pixel in f. In a similar manner, the *convolution* of the $w(x, y)$ and $f(x, y)$ is given by:

$$k(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

Sine every element of w must visit every pixel in f, the input image has to be padded.

Smoothing images and sharping images are two common applications of linear spatial filtering. A smoothing filter can also be called an averaging filter, since it computes the wighted average of the pixels in its neighborhood. When all the coefficients of the mask are all the same, the result from the averaging filter is the standard average. If we want to give more importance to the middle pixel, we can construct a filter with bigger number in the center and smaller number in other positions. The uses of smoothing filters range from blurring to getting the local intensity information. The purpose of sharpening image is to highlight transitions in intensity. Because averaging is analogous to integration, it is logical to know that sharping can be accomplished by spatial differentiation. In fact, the strength of the response of a derivative operator can reflect the degree of intensity discontinuity of the image. The bigger the differentiation of the intensity is, the quicker the intensity transition is. Thus, differentiation filters can enhance edges and other discontinuities and deemphasize areas with slowly varying intensities. Image sharpening can be used in many tasks such as electronic printing, medical imaging and industrial inspection.

2.1.3 Image segmentation

In this part, we will introduce the purpose of this thesis which is image segmentation. Segmentation is one of the most difficult tasks in image processing and it is also a very important basis for image analysis. The accuracy of the segmentation result determines the eventual result of image analysis procedures. So we should put more attention on this mission. Image segmentation is to divide an image into its constituent regions or objects. How the subdivision of this image is carried out is based on the problem being solved. Let R denotes the whole spatial region. The image segmentation [8] can be viewed as a process that splits the whole region R into n subregions, $R_1, R_2, \ldots, R_n$ such that

(i) $\bigcup_{i=1}^{n} R_i = R$.

(ii) R_i is a connected set, $i = 1, 2, \ldots, n$.

(iii) $R_i \cap R_j = \varnothing$ for all i and j, $i \neq j$.

(iv) $Q(R_i) =$ TRUE for $i = 1, 2, \ldots, n$.

(v) $Q(R_i \cup R_j) =$ FALSE for any adjacent regions R_i and R_j.

Where $Q(R_k)$ is a logical predicate defined over the points in set R_k, $\varnothing$ is the null set, and the symbols $\cup, \cap$ represent set union and intersection respectively.

Condition (i) says that all the subregions can join together to become the whole region. Condition (ii) indicates that all the pixels in one same subregion must be connected in some predefined criteria. Condition (iii) means that the subregions can not have common part with each other. Condition (iv) requires that pixels in the segmented region must satisfy the properties predefined by the logical predicate Q. At last, condition (v) means that the pixels belonging to two adjacent regions can not be the same in the sense of predicate (Q).

How to segment the image depends on how to define the logical predicate. Basically, most of the segmentation algorithm are based on either discontinuity or similarity of intensity values. The first approach based on the discontinuity is to partition image by finding the edges which always have abrupt changes of intensity values. The second method based on similarity of intensity values divides the image into regions where some sort of measure is similar within each region. For example, one can use the average intensity value as a measure to find regions with similar grayscale or the standard deviation to find regions with similar texture [9].

2.2 Optimisation theory

Because any maximisation problem can be transformed to a minimisation problem, in the following part, we only discuss the definitions and theorems of minimisation problem for convenience.

2.2.1 Gateaux differential

The definition of derivative of a function of real variable is familiar to us . This kind of derivative measures the sensitivity to change of function value which is determined by another quantity [10].

Definition 2.2.1. *The derivative of $f : \mathbb{R}^n \to \mathbb{R}$ with respect to x is the function $\nabla f(x)$ and is defined as*

$$\nabla f(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}$$

However, there are lots of cases where the function $f(x)$ is a function of a function. Then the definition of the derivative in the elementary calculus cannot be used. The Gateaux differential [11] is used to generalize the concept of directional derivative in differential calculus. The definition of Gateaux differential is very similar with the definition of derivative in elementary calculus.

Definition 2.2.2. *We say that $f'(x)$ is the Gateaux differential of $f(x)$ if for all $d \in C$ the directional derivative of f at x in the direction d is*

$$f'(x; d) = \lim_{h\to 0} \frac{f(x+hd) - f(x)}{h} = \langle f'(x), d\rangle$$

where $\langle \cdot, \cdot \rangle$ is a proper inner product on the function space and C is a set from a general function space.

2.2.2 Unconstrained optimization

Unconstrained minimization problem considers the problem of minimizing an objective function that depends on real variables with no restrictions on their values. Its mathematical model can be formulated as

$$\min_{x\in\mathbb{R}^n} f(x). \tag{2.3}$$

where $n \geq 1$ and $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth function.

Main optimality conditions

Proposition 2.2.1. *(Necessary Optimality Conditions) Let x^* be an unconstrained local minimum of $f : \mathbb{R}^n \to \mathbb{R}$, and assume that f is continuously differentiable in an open set S containing x^*. Then*

$$\nabla f(x^*) = 0. \qquad\qquad (First\ Oder\ Necessary\ condition)$$

If in addition f is twice continuously differentiable within S, then its second order derivative $\nabla^2 f(x^*)$ is positive semidefinite:

$$\nabla^2 f(x^*) \geq 0. \qquad\qquad (Second\ Oder\ Necessary\ condition)$$

Proposition 2.2.2. *(Second Order Sufficient Optimality Conditions) Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable in an open set S. Suppose that a vector $x^* \in S$ satisfies the conditions*

$$\nabla f(x^*) = 0, \qquad\qquad \nabla^2 f(x^*) : positive\ definite.$$

Then, x^* is a strict unconstrained local minimum of f. In particular, there exist scalars $\gamma > 0$ and $\epsilon > 0$ such that

$$f(x) \geq f(x^*) + \frac{\gamma}{2} \|x - x^*\|^2, \qquad \forall x\ with\ \ \|x - x^*\| \leq \epsilon.$$

According to the necessary optimality condition and the sufficient optimality condition [12], a straightforward way to solve the unconstrained problem (2.3) can be concluded as following: First, find all the possible local minimum points by using the first order necessary condition ($\nabla f(x^*) = 0$); Then filter out those that do not satisfy the second order necessary condition ($\nabla^2 f$ is positive semidefinite); Thirdly, check all those remaining candidates if they satisfy the second order sufficient condition($\nabla^2 f$ is positive definite) to make sure that they are strict local minimum; Finally, chose the one which has the smallest function value among all the local minimums as the global minimum. However, to find the global minimum for most of the practical problems by using this way requires too much time and effort.

Since any local minimiser is a global minimizer for convex function [13], the minimisation problem with a convex function as the cost function is much easier to be solved.

Gradient method

According to discussion above, searching for points where $\nabla f(x) = 0$ is a natural idea for finding the minimum of a convex functional. However, not all convex functional can be solved analytically. Actually speaking, most of the practical problems are impossible to find the analytical solution. Gradient method [12] is a very fundamental and useful approach to finding the estimate solution.

Gradient method is based on an important idea, called *iterative descent*. Suppose the function that we are dealing with is $f : \mathbb{R}^n \to \mathbb{R}$. The *iterative descent* works as follows: We start at an initial point x^0 and generate sequence x^k, such that the function value of f decreases after each iteration,

$$f(x^{k+1}) < f(x^k), \qquad k = 0, 1, \ldots, \tag{2.4}$$

In doing so, we successively improve our current estimate result. Hopefully, the sequence generated by the description above will converge to the minimum.

The difference among algorithms based on the *iterative descent* lies in how to update every point in each iteration. In gradient method, the sequence is constructed by the following formula:

$$x^{k+1} = x^k - \epsilon^k \nabla f(x^k) \tag{2.5}$$

where ϵ^k is the step length. The reason why this method can promise $f(x^{k+1}) < f(x^k)$ can be explained by its Taylor expansion shown below:

The first order Taylor series expansion around x gives us that

$$\begin{aligned} f(x^{k+1}) &= f(x^k) + \nabla f(x^k)'(x^{k+1} - x^k) + o\left(\left\|x^{k+1} - x^k\right\|\right) \\ &= f(x^k) - \epsilon^k \left\|\nabla f(x^k)\right\|^2 + o\left(\epsilon^k \left\|\nabla f(x^k)\right\|\right) \end{aligned}$$

So we get

$$f(x^{k+1}) = f(x^k) - \epsilon^k \left\|\nabla f(x^k)\right\|^2 + o\left(\epsilon^k\right)$$

When ϵ^k is near zero, the term $\epsilon^k \left\|\nabla f(x^k)\right\|^2$ overwhelms $o(\epsilon^k)$. Obviously, the term $\epsilon^k \left\|\nabla f(x^k)\right\|^2$ is positive and $f(x^{k+1})$ is smaller than $f(x^k)$.

We can expand this method into a more general optimisation problem:

$$\min_{x \in C} f(x)$$

where $f : C \to \mathbb{R}$. When C is a subset of a function space, we have to use the Gateaux differential. So similarly as above the minimising sequence is constructed according to

$$x^{k+1} = x^k - \epsilon^k f'(x^k),$$

where $f'(x; d) = \langle f'(x), d \rangle$, $\forall d \in C$.

2.2.3 Constrained optimization

In this section, we talk about the constrained optimization problem

$$\text{minimize } f(x) \quad \text{s.t } x \in X,$$

where X is a nonempty and convex subset of $\mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function over X. We can see that the unconstrained problem is just a special case of constrained problem when $X = \mathbb{R}^n$. Usually, the convex set is constructed by equations and inequalities. In this section, we only focus on a special type where the set X consists of only equations.

The form of the equality constrained optimisation problem is shown as following:

$$\text{minimize } f(x) \quad \text{s.t } h_i(x) = 0 \quad \text{for } i = 1, \ldots, m,$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $h_i : \mathbb{R}^n \to \mathbb{R}$ are continuously differentiable functions. For notational convenience, we can write the constraints in a more compact form

$$h(x) = 0.$$

where

$$h^T = (h_1, \ldots, h_m).$$

There is one important function called *Lagrangian function* in the constrained optimization problem. The definition of this function $L : \mathbb{R}^{n+m} \to \mathbb{R}$ is following

$$L(x, \lambda) = f(x) + \lambda^T h(x).$$

where $\lambda^T = (\lambda_1, \ldots, \lambda_m)$ is a vector.

Main optimality conditions

Proposition 2.2.3. *(Necessary Optimality Conditions) if x^* is the local minimum of f subject to $h(x) = 0$, and the gradients of the constraints $\nabla h_1(x)$, $\ldots, \nabla h_m(x)$ are linearly independent. Then we can find a unique vector $\lambda^* = (\lambda_1^*, \ldots, \lambda_m^*)$ called a Lagrange multiplier, such that*

$$\nabla_x L(x^*, \lambda^*) = 0.$$

If f and h are twice continuously differentiable, we have

$$y' \nabla_{xx}^2 L(x^*, \lambda^*) y \geq 0, \qquad \textit{for all } y \in V(x^*)$$

where $V(x_*)$ is the subspace of the first feasible variations

$$V(x^*) = \{y \mid \nabla h_i(x^*)' y = 0, \ i = 1, \ldots, m\}.$$

Proposition 2.2.4. *(Sufficient Optimality Conditions) We assume that f and h are twice continuously differentiable, and let $x^* \in \mathbb{R}^n$ and $\lambda^* \in \mathbb{R}^m$ satisfy*

$$\nabla_x L(x^*, \lambda^*) = 0, \qquad \nabla_\lambda L(x^*, \lambda^*) = 0$$

$$y' \nabla^2_{xx} L(x^*, \lambda^*) y > 0, \qquad \textit{for all } y \neq 0 \textit{ with } \nabla h(x^*)' = 0$$

Then we can say that x^* is a strict local minimum of f subject to $h(x) = 0$.

The optimality conditions above shows that the gradient of the cost function $\nabla f(x^*)$ is a linear combination of the gradients of the constraints $\nabla h_i(x^*)$. If $\nabla f(x^*)$ is not a linear combination of $\nabla h_i(x^*)$, at least one projection of $\nabla f(x^*)$ along the tangent line of the intersection of $h_i(x) = 0$ is nonzero. Thus the cost function f can be increased in one direction without violating any of the constraints. This is a contradiction with the fact that x^* is the local minimum. So $\nabla f(x^*)$ must belong to the subspace spanned by the constraints gradients. Like the case in unconstrained problem, the analytical solution for some problems can be found by making use of these optimality conditions [12].

Augmented Lagrangian method

As we said in the unconstrained problem, there are some problems which can not be solved analytically. Here, we introduce one method: Augmented Lagrange method [12]. The idea behind this method is to transform the constrained problem into an unconstrained problem by eliminating some or all of the constraints. We can fulfill this purpose by adding to the cost function a penalty term that prescribes a high cost to infeasible points.

At the central position of this method is the augmented Lagrangian function $L_c : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ given by

$$L_c(x, \lambda, c) = f(x) + \lambda^T h(x) + \frac{c}{2} \|h(x)\|^2$$

Where $c > 0$ is called penalty parameter. The reason why minimizing $L_c(x, \lambda)$ can solve the constrained problem is based on two facts: one is that x^* is a strict local minimum of the augmented Lagrangian $L_c(x, \lambda^*)$ when c is large enough. This suggests that we can find a good approximation to x^* by minimizing the unconstrained function $L_c(x, \lambda)$, when λ is close to λ^*. The second fact is that because of the high cost for the infeasible points, the unconstrained minimum of $L_c(x, \lambda)$ should be nearly feasible. We also expect that $L_c(x, \lambda) \approx f(x)$ near the feasible set. Thus we expect that we can get a good approximation to x^* by finding the minimum of $L_c(x, \lambda)$, if c is big enough.

This method includes solving a sequence of problems as

$$\text{minimize } L_{c^k}(x, \lambda^k),$$

where $0 < c^k < c^{k+1}$ and $c^k \to \infty$. The global minimum of each problem in the sequence of problems above converges to a local minimum of the original constrained problem. In each iteration, c^k is increased and λ^k is updated by

$$\lambda^{k+1} = \lambda^k + c^k h(x^k). \tag{2.6}$$

The reason can be explained by the following observation:

$$\nabla_x L_a(x^k, \lambda^k, c^k) = \nabla f(x^k) + (\lambda^k + c^k h(x^k))^T \nabla h(x^k) \approx 0$$

and

$$\nabla_x L_a(x^*, \lambda^*) = \nabla f(x^*) + \lambda^{*T} \nabla h(x^*) = 0$$

The Lagrangian multiplier approximation λ^k is updated by (2.6) to approach λ^* at the same time when x^k converge to x^*.

The Augmented Lagrangian method is reliable and always converges to at least one local minimum. When the cost function is convex, we can get the global minimum by penalty function method.

Chapter 3

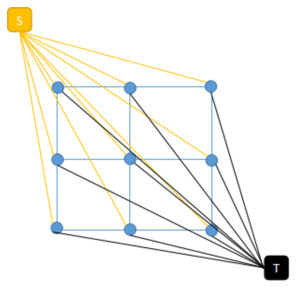
Max-flow and min-cut

Min-cut of a graph is to search for a cut that is minimal in some sense. Some energy minimisation problem can be transformed into a min-cut problem. Thus some optimization problem can be solved by the min-cut method. According to the theorem of min-cut and max-flow, we can solve the min-cut problem by maximising its corresponding flows instead. In this chapter, the definition of these two kinds of problems,their connection between each other are explained before introducing the continuous max-flow method and the supervised continuous max-flow method.

3.1 Min-cut

A graph consists of a set of vertices V and a set of edges E [9]. In order to correspond with our image processing context, we also need two terminal vertices: the source s and the sink t. Each pixel in the image is regarded as a vertice in the graph and each pixel is linked with its neighbours by spatial edges e_n, as well as to both terminal vertices by the two terminal edges e_s and e_t(see figure 3.1(a)). Our purpose is to divide all the pixels in the image into two disjoint groups. The group which has the pixels connecting with the sources is called foreground, and the left pixels form the background group. Then the min-cut problem can be thought of as cutting off all the spatial edges between pixels belonging to different groups and also removing one of the terminal edges from each pixel so that it either is connected with s or connected with t (see figure 3.1(b)).

Now the whole set V has been partitioned into two subset: one is the foreground subset V_s containing the terminal vertex s and the foreground pixels, the other one is the background subset V_t containing the vertices left. This cut partitioning the set into two parts is called an $s-t$ cut. Here, each


(a)

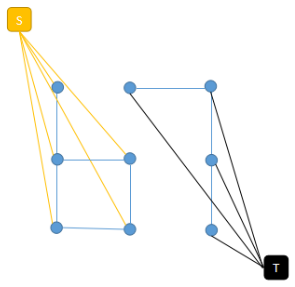

(b)

Figure 3.1: Figure showing a graph and containing 9 pixels, two terminal vertices and spatial and terminal edges. 3.1a is the whole part of the network. In 3.1b the 9 pixels are partitioned into two parts by cutting off some of the edges. The edges which are removed from the 3.1a are missing in 3.1b.

edge $e \in E$ has a non-negative cost $C(e) \geq 0$. The mathematical model of the min-cut problem is formulated by the following:

$$\min_{E_{st} \subset E} \sum_{e \in E_{st}} C(e), \text{where } E_{st} = \{e \in E \mid e = (v_1, v_2),\ v_1 \in V_s,\ v_2 \in V_t\},$$

Our task is to determine where we should make the cut to minimize the sum of the costs of the edges that are cut from the edge set E.

3.2 Max-flow

We can also look at this graph by a closely related way, which is thinking of the graph as a network, where each edge is reckoned as a pipe. Calculating the maximum amount of water that can flow through these pipes from the source s to the sink t is our purpose. Like the case in the min-cut problem, there are some constraints that must be fulfilled on those pipes: the flow p through a pipe can not be bigger than the pipe's capacity, and the amount of the water flowing into a vertex should be equal to the amount of the water flowing out of the vertex. Its mathematical formulation is shown as following:

- Source flows capacity:

$$0 \leq p_s(v) \leq C_s(v), \tag{3.1}$$

  where $p_s(v)$ and $C_s(v)$ are shorthand for $p(e_s(v))$ and $C(e_s(v))$ respectively.

- Sink flows capacity:
$$0 \leq p_t(v) \leq C_t(v), \tag{3.2}$$
where $p_t(v)$ is the abbreviation of $p(e_t(v))$ and $C_t(v)$ is shorthand for $C(e_t(v))$.

- Spatial flows capacity:
$$|p(e_n)| \leq C(e_n), \tag{3.3}$$
The flows through the spatial edges linking pixels can have two directions. If the flow in the pipe $e_n = (v_1, v_2)$ goes from v_1 to v_2, the value of the flow is positive. On the contrary, it is negative if the flow goes from the opposite direction. Thus here we use the absolute value in the inequality above.

- flows conservation:
$$\sum_{w \in N_4(v)} p((v,w)) - p_s(v) + p_t(v) = 0, \tag{3.4}$$
where $N_4(v)$ stands for the 4-connectivity neighbourhood of pixel v.

Thus the max-flow problem is to maximize the amount of flow from source s to sink t while satisfying the above constraints:
$$\max_{p_s} \sum_{v \in V \setminus \{s,t\}} p_s(v), \text{ subject to (3.1), (3.2), (3.3) and (3.4).} \tag{3.5}$$

3.3 Max-flow min-cut theorem

Theorem 3.3.1. *For any network the maximal flow value from source s to t is equal to the minimal cut capacity of all cuts separating s and t.*

This theorem [14] indicates that the cost function value of the min-cut problem is identical with the max-flow problem. Thus when our task is to calculate the minimum of an energy function by cutting some edges from some graph, we can solve this problem by maximising its corresponding network instead. From our experience, the algorithm developed under the max-flow circumstance is much more easier and faster than those founded under the min-cut circumstance.

3.4 Continuous max-flow and min-cut

We can also formulate the max-flow and min-cut problem in the continuous setting [15] which can help us to develop mathematical algorithm. Now the vertices except the source s and sink t in the domain Ω are continuous set of points. The continuous version of constraints on flows are as following:

$$\begin{aligned}
p_s(x) &\le C_s(x), && \forall x \in \Omega; && (3.6)\\
p_t(x) &\le C_t(x), && \forall x \in \Omega; && (3.7)\\
|p(x)| &\le C(x), && \forall x \in \Omega; && (3.8)\\
\text{div} p(x) - p_s(s) + p_t(x) &= 0, && \text{a.e. } x \in \Omega; && (3.9)\\
p(x) \cdot v &= 0, && \text{on } \partial\Omega; && (3.10)
\end{aligned}$$

where a.e. means "for almost every" and v is the outward normal vector to the boundary $\partial\Omega$. Constraints (3.6) and (3.7) are changed a little bit. Because p_s and p_t are directed, the positiveness of the flows are not needed. Thus the capacity $C_s(x)$ and $C_t(x)$ are not necessary to be positive. The continuous max-flow mathematical model which also is called the primal model is formulates as:

$$\sup_{p_s, p_t, p} \int_\Omega p_s(x)\, dx, \quad \text{subject to constraints (3.6) through (3.10).} \qquad (3.11)$$

Because there is an equality constraint (3.9), we can reduce the number of constraints by introducing the Lagrangian multipliers. Here the number of the equality constraints are infinite, so the Lagrangian multiplier u should be a function. Then the Lagrangian function of the primal model is :

$$\begin{aligned}
L(p_s, p_t, p, u) &= \int_\Omega p_s(x)dx + \int_\Omega u(x)(\text{div} p(x) - p_s(x) + p_t(x))dx\\
&= \int_\Omega \left[(1-u(x))p_s(x) + u(x)p_t(x) + u(x)\text{div} p(x)\right] dx
\end{aligned}$$

Now the task of the problem (3.11) has become finding the saddle point of L subject to (3.6), (3.7), (3.8) and (3.10). Because the Lagrangian function L is linear in all variables and we have convex constraints on flows, according to the minimax theorem [11], we can find the saddle point by solving the following model:

$$\begin{aligned}
&\min_u \sup_{p_s, p_t, p} \left\{ \int_\Omega [(1-u(x))p_s(x) + u(x)p_t(x) + u(x)\text{div} p(x)]dx \right\} \qquad (3.12)\\
&\text{s.t.} \quad p_s(x) \le C_s(x), \quad p_t \le C_t(x), \quad |p(x)| \le C(x) \quad \forall x \in \Omega.
\end{aligned}$$

This is called the primal-dual problem which has the same solution with the primal model. The primal-dual model (3.12) can be rearranged as

$$\begin{aligned}\min_u \Bigg\{ \sup_{\substack{|p(x)|\\ \le C(x)}} \int_\Omega u(x)\mathrm{div} p(x)\ dx + \sup_{\substack{p_s(x)\\ \le C_s(x)}} \int_\Omega p_s(x)(1-u(x))\ dx \\ + \sup_{\substack{p_t(x)\\ \le C_t(x)}} \int_\Omega p_t(x)u(x)\ dx \Bigg\}. \end{aligned} \tag{3.13}$$

First let us see the first part of (3.13). The divergence theorem gives us that

$$\begin{aligned}\int_\Omega u(x)\mathrm{div} p(x)\ dx &= \int_\Omega \mathrm{div}(p(x)u(x))\ dx - \int_\Omega p(x)\cdot\nabla u(x)\ dx \\ &= \int_{\partial\Omega} (p(x)u(x))\cdot v\ ds - \int_\Omega p(x)\cdot\nabla u(x)\ dx. \end{aligned} \tag{3.14}$$

Because of the constraint (3.10) on the boundary, we get

$$\int_\Omega u(x)\mathrm{div} p(x)\ dx = -\int_\Omega p(x)\cdot\nabla u(x)\ dx. \tag{3.15}$$

It is obvious that

$$-\int_\Omega p(x)\cdot\nabla u(x)\ dx \le \int_\Omega |p(x)||\nabla u(x)|\ dx \le \int_\Omega C(x)|\nabla u(x)|\ dx.$$

When $p(x) \to -C(x)$, the supremum is attained as

$$\sup_{\substack{|p(x)|\\ \le C(x)}} \int_\Omega u(x)\mathrm{div} p(x)\ dx = \int_\Omega C(x)|\nabla u(x)|\ dx. \tag{3.16}$$

We should note that $u(x) \in [0,1]$. if not, the energy of the last two terms is infinity. Since we have proven that the energy of the first term is be non-negative, the sum of the energy of the two terms left in (3.13) being infinite will lead the energy of (3.13) to being infinite. However this can not be the case for the reason that we have proven that there exists at least one saddle point. Now we know that $u(x)$ and $1-u(x)$ are all positive. It is easy to prove that

$$\begin{aligned}\sup_{\substack{p_s(x)\\ \le C_s(x)}} \int_\Omega p_s(x)(1-u(x))\ dx \ + \sup_{\substack{p_t(x)\\ \le C_t(x)}} \int_\Omega p_t(x)u(x)\ dx \\ = \int_\Omega (1-u(x))C_s(x)\ dx + \int_\Omega u(x)C_t(x)\ dx \end{aligned} \tag{3.17}$$

By adding (3.16) to (3.17), the primal-dual model (3.12) can be written as

$$\min_{u\in[0,1]}\left\{\int_\Omega [(1-u(x))C_s(x)+u(x)C_t(x)+C(x)|\nabla u(x)|]\ dx\right\}. \tag{3.18}$$

This model is called dual model which is equivalent to the primal model (3.11) and the primal-dual model (3.12). And the dual model can also be called the continuous min-cut model. We can simplify this model by only remaining the part depending on u as

$$\min_{u\in[0,1]}\left\{\int_\Omega [u(x)(C_t(x)-C_s(x))+C(x)|\nabla u(x)|]\ dx\right\}. \tag{3.19}$$

3.5 Supervised continuous max-flow and min-cut

Some priori information can help us with solving the problem. Here, the priori given information is that some pixels in the image have been labeled in advance, as foreground or background. We denote the priori given foreground by Ω_f while denoting the priori given background by Ω_b.

Two indicator functions [15] which characterize these two priori given regions are given as

$$u_f(x)=\begin{cases}1 & x\in\Omega_f\\ 0 & x\neq\Omega_f\end{cases},\qquad u_b(x)=\begin{cases}0 & x\in\Omega_b\\ 1 & x\neq\Omega_b\end{cases} \tag{3.20}$$

We regard the supervised max-flow model as a problem of flow cost. For the source flow $p_s(x)$, the flow costs nothing when $x\in\Omega_b$; otherwise, it costs $p_s(x)$. Since $u_b(\Omega_b)=0$ and $u_b(\Omega\backslash\Omega_b)=1$, the total cost from source flows is $\int_\Omega u_b(x)p_s(x)$. For the sink flow $p_t(x)$, as a flow passes a known foreground pixel, it is valued as $-p_t(x)$ where its negative sign means it reduces the cost; otherwise, it is valued as zero. Likewise, since $u_f(\Omega_f)=1$ and $u_f(\Omega\backslash\Omega_f)=0$, the total cost from sink flows is $-\int_\Omega u_f(x)p_t(x)$.

In view of the continuous max-flow model (3.11), the supervised continuous max-flow is formulated as

$$\sup_{p_s,p_t,p}\int_\Omega u_b(x)p_s(x)\ dx-\int_\Omega u_f(x)p_t(x)\ dx \tag{3.21}$$

subject to the flow constraints (3.6) through (3.10). If there is no priori information given in advance, we have $u_f=0$ and $u_b=1$, $\forall x\in\Omega$. Then, the problem (3.21) become the continuous max-flow model. Thus the continuous max-flow primal model is a special case of the supervised continuous max-flow primal model.

Like the continuous max-flow model, the equivalent constraints can be removed by introducing the Lagrangian multiplier $u(x)$. Then the equivalent primal-dual model of (3.21) is obtained as

$$\begin{aligned}\min_u \sup_{p_s,p_t,p} & \int_\Omega (u_b(x) - u(x))p_s(x)\ dx + \int_\Omega (u(x) - u_f(x))p_t\ dx + \\ & \int_\Omega u(x)\mathrm{div}p(x)\ dx \qquad (3.22) \\ \text{s.t.} \quad & p_s(x) \le C_s(x), \quad p_t \le C_t(x), \quad |p(x)| \le C(x) \quad \forall x \in \Omega\end{aligned}$$

which can be rearranged as

$$\begin{aligned}\min_u \Bigg\{ & \sup_{\substack{|p(x)| \\ \le C(x)}} \int_\Omega u(x)\mathrm{div}p(x)\ dx + \sup_{\substack{p_s(x) \\ \le C_s(x)}} \int_\Omega p_s(x)(u_b(x) - u(x))\ dx \\ & + \sup_{\substack{p_t(x) \\ \le C_t(x)}} \int_\Omega p_t(x)(u(x) - u_f(x))\ dx \Bigg\}. \qquad (3.23)\end{aligned}$$

Since there exists at least a saddle point for (3.23), the energy of the function (3.23) cannot be infinite. So there is a constraint hiding behind this function which is that

$$u_f(x) \le u(x) \le u_b(x), \qquad x \in \Omega. \qquad (3.24)$$

So now, $u_b(x) - u(x)$ and $u(x) - u_f(x)$ are all positive. By maximizing over all flows p_s, p_t and $p(x)$, we get the dual model

$$\begin{aligned}\min_u & \int_\Omega (u_b(x) - u(x))C_s(x)\ dx + \int_\Omega (u(x) - u_f(x))C_t(x)\ dx \\ & + \int_\Omega C(x)|\nabla u(x)|\ dx \qquad (3.25) \\ . \quad \text{s.t.} \quad & u_f(x) \le u(x) \le u_b(x) \qquad \forall x \in \Omega.\end{aligned}$$

which is equivalent to the primal supervised model (3.21) and the primal-dual supervised model (3.22). Because we have $u_f(\Omega_f) = u_b(\Omega_f) = 1$ and $u_f(\Omega_b) = u_b(\Omega_b) = 0$, the inequality constraints in (3.25) gives us that $u(\Omega_f) = 1$ and $u(\Omega_b) = 0$. So the priori given information which Ω_f is labeled as foreground and Ω_b is labeled as background still is guaranteed in (3.25).

When no priori information is given, $u_f(x) = 0$ and $u_b(x) = 1$, the supervised dual model (3.25) coincides with the non-supervised dual model (3.18).

Thus the non-supervised continuous max-flow dual model is a special case of supervised continuous max-flow dual model.

Since $u_b(x)$ and $u_f(x)$ are given, after removing those parts which does not depend on u in (3.25), The dual model can be shortened as

$$\begin{aligned} \min_u & \int_\Omega u(x)(C_t(x) - C_s(x))\ dx + \int_\Omega C(x)|\nabla u(x)|\ dx \\ \text{s.t.} & \quad u_f(x) \leq u(x) \leq u_b(x) \qquad \forall x \in \Omega. \end{aligned} \tag{3.26}$$

Comparing (3.19) with (3.26), we can see that the only difference between the non-supervised continuous max-flow dual model and the supervised continuous max-flow dual model is the constraint on $u(x)$. In fact this difference comes from the given information which forces $u(x)$ in foreground to be 1 and $u(x)$ in background to be 0.

Chapter 4

Segmentation models

While many image segmentation models rely heavily on edge detection, one popular and well-known model inspired by the Mumford-shah model[16], which is named Chan-Vese model [4], ignores edges completely. This model is based on intensity homogeneity of images, however, the intensity inhomogeneities exists in many real word images especially in medical images. In order to segment this kind of images, a local region-based Chan–Vese model[5] is proposed . But the sensitivity to the initialization input limits its application. Thus Xing Hui and Peng Yali present a regional adaptive active contour model [6] which is the combination of the original Chan-Vese model and the local region-based chan-vese model. In this chapter, first we introduce the Chan-Vese model. Then the local-region based chan-vese model is presented before deriving the regional adaptive active contour model.

4.1 Chan-Vese segmentation model

In this section, we introduce the Chan-Vese segmentation model and the supervised Chan-Vese model which involves in the priori given information.

4.1.1 Chan-Vese segmentation model without priori gi-
ven information

The Chan-Vese model is used to divide the digital image into two parts; background and foreground, where the foreground contains the object of interest in the image. Here we only give the formulation of this model for grey scale images as in [9] [17]. The Chan-Vese segmentation model is in fact minimising the following functional:

$$E_{CV}(u,c) = J(u) + \frac{\lambda}{2}\left\{\langle 1-u,(I-c_0)^2\rangle + \langle u,(I-c_1)^2\rangle\right\} \tag{4.1}$$

with respect to u and c. The variable u is the characteristic function of the region Σ representing the foreground while c_1 is the average grey value inside Σ and c_0 is the average grey value outside Σ. The weight λ is a positive number and $J(u) = \int_\Omega |\nabla u(x)|$ is the total variation of u.

We can explain why this model works by a simple case where the image includes two approximately constant parts Ω_0 and Ω_1 with different grey values. The energy of the Chan-Vese model is dominated by the last two parts which is called fitting term. Minimizing the fitting term can give us a region Σ which equals the region Ω_1 representing the object of interest. Because $u(x) = 1$ for $x \in \Sigma$ and $u(x) = 0$ for $x \in \Omega\backslash\Sigma$, for the simplicity of the following discussion, we rewrite the the fitting terms as

$$T_0 + T_1 = \int_{\Omega\backslash\Sigma}(I-c_0)^2\ dx + \int_\Sigma (I-c_1)^2\ dx \tag{4.2}$$

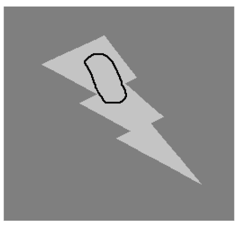
Then, when Σ is inside Ω_1 as in figure 4.1(a), c_1 is equal to the grey value inside Ω_1 and c_0 is not equal to the grey value inside Ω_1 or to the grey value in Ω_0. So $T_1 \approx 0$ and $T_0 > 0$. If $\Omega_1 \in \Sigma$ as in figure 4.1(b), c_1 lies between the grey value in Ω_0 and in Ω_1 and c_o is equal to the grey value in Ω_0. Thus $T_0 \approx 0$ and $T_1 > 0$. In the third case as shown in figure 4.1(c), the region Σ has connection with both Ω_0 and Ω_1. Then c_0 and c_1 are both between the grey value inside Ω_0 and Ω_1 which results in both terms T_1 and T_0 being positive. Finally, when contour line C of the region Σ matches with the boundary of the region Ω_1 as shown in the figure 4.1(d), c_1 and c_0 will be equal to the the grey value inside Ω_1 and the grey value inside Ω_0 respectively. Then both terms T_0 and T_1 are close to 0 and the minimum of the energy function is obtained.

Total variation

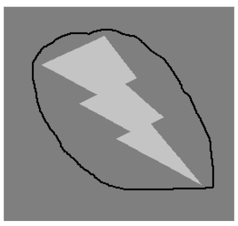
The total variation of $u \in L^1(\Omega)$ in [18] and [19] is defined as following:

$$J(u) = \sup\left\{\int_\Omega u\ \mathrm{div}(\xi)\ dx \mid \xi \in C_c^1(\Omega,\mathbb{R}^n),\ ||\xi||_\infty \le 1\right\}, \tag{4.3}$$

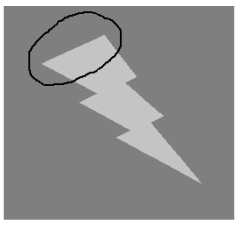
where $\Omega \in \mathbb{R}^n$ is a bounded open domain, the function $u : \Omega \to \mathbb{R}$ is absolutely integrable and ξ is one time continuously differentiable function with compact support. A function u is said to be of bounded variation, if $J(u) < \infty$.



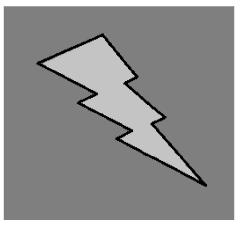
(a) $T_0 > 0$, $T_1 \approx 0$



(b) $T_0 \approx 0$, $T_1 > 0$



(c) $T_0 > 0$, $T_1 > 0$



(d) $T_0 \approx 0$, $T_1 \approx 0$

Figure 4.1: Figure showing four different cases with the respect to the position of the region Σ (region inside the black line) relative to Ω_1 (the lightning-shaped region).

The product rule for differentiation gives us

$$\int_\Omega u\mathrm{div}(\xi)\ dx = \int_\Omega \mathrm{div}(\xi u)\ dx - \int_\Omega \xi \cdot \nabla u\ dx \tag{4.4}$$

According to the divergence theorem, we get the following

$$\int_\Omega u\mathrm{div}(\xi)\ dx = \int_{\partial\Omega} (\xi u) \cdot v\ dS - \int_\Omega \xi \cdot \nabla u\ dx \tag{4.5}$$

where v is the outward unit normal vector along $\partial\Omega$. Since ξ is a function with support, the first term on the right hand of (4.5) becomes *zero*. Then we have

$$\int_\Omega u\mathrm{div}(\xi)\ dx = -\int_\Omega \xi \cdot \nabla u\ dx \le \int_\Omega |\xi \cdot \nabla u|\ dx \le \int_\Omega |\xi||\nabla u|\ dx \tag{4.6}$$

where the last step comes from Cauchy-Schwarz inequality. From the definition of the total variation, we know that $||\xi||_\infty \le 1$. Then we get

$$\int_\Omega u\mathrm{div}(\xi)\ dx \le \int_\Omega |\nabla u|\ dx \tag{4.7}$$

The supremum $J(u) = \int_\Omega |\nabla u|\ dx$ can be attained if we let $\xi \to \frac{-\nabla u}{|\nabla u|}$.

In Chan-Vese model, u represents the characteristic function of $\Sigma \subset \Omega$, then we have

$$\int_{\Omega} u\mathrm{div}(\xi)\ dx = \int_{\Sigma} \mathrm{div}(\xi)\ dx = \int_{\partial\Sigma} \xi \cdot v\ dS \leq \int_{\partial\Sigma} |\xi \cdot v|\ dS \tag{4.8}$$

By using the Cauchy-Schwarz inequality, we get the following

$$\int_{\Omega} u\mathrm{div}(\xi)\ dx \leq \int_{\partial\Sigma} |\xi||v|\ dS \leq |\partial\Sigma|. \tag{4.9}$$

when we set $\xi = v$ on $\partial\Sigma$, $\int_{\Omega} u\mathrm{div}(\xi)\ dx = \int_{\partial\Sigma} \xi \cdot v\ dS = \int_{\partial\Sigma} |\xi||v|\ dS = |\partial\Sigma|$ and $J(u) = |\partial\Sigma|$ can be obtained. Thus, the first term is used to control the length of the border of the region Σ.

The truncation lemma

However, the function u in the model (4.1) is required to be a binary function. This fact indicates that the model (4.1) is a non-convex optimisation problem which is hard to be solved by the theory introduced in chapter 2. Nevertheless, Chan, Esedoglu and Nikolova [20] have proven that the problem with fixed c can be solved globally by relaxing the constraint on u to allow $u \in K$ where

$$K = \{u \in BV(\Omega) \mid 0 \leq u(x) \leq 1\ \forall x \in \Omega\}. \tag{4.10}$$

The energy function E_{CV} can be rewritten as

$$J(u) + \frac{\lambda}{2}\langle 1, (I - c_0^2)\rangle + \frac{\lambda}{2}\{\langle -u, (I - c_0^2)\rangle + \langle u, (I - c_1^2)\rangle\}. \tag{4.11}$$

When c is fixed, minimizing E_{CV} with respect to u is equivalent to minimisation of the following functional

$$\widetilde{E}_{CV}(u) = J(u) + \frac{\lambda}{2}\{\langle -u, (I - c_0^2)\rangle + \langle u, (I - c_1^2)\rangle\} = J(u) + \frac{\lambda}{2}\langle g, u\rangle, \tag{4.12}$$

where $g = (I - c_1)^2 - (I - c_0)^2$. Next we state the lemma and prove it as done in [17]:

Lemma 4.1.1 (The Truncation Lemma I). *If* $u_* = \underset{u \in K}{\mathrm{arginf}} \widetilde{E}_{CV}(u)$, *then* $u_*^t = \underset{u \in \{0,1\}}{\mathrm{arginf}} \widetilde{E}_{CV}(u)$ *for almost all* $t \in [0, 1]$ *where* $u_*^t(x)$ *is defined as*

$$u_*^t(x) = \begin{cases} 1 & \text{if } u_*(x) > t, \\ 0 & \text{otherwise} \end{cases} \qquad \forall x \in \Omega.$$

Proof. Because of the coarea formula, $J(u_*) = \int_0^1 J(u_*^t)\, dt$ and the layer cake representation, $\langle g, u_* \rangle = \int_0^1 \langle g, u_*^t \rangle$, we have $\widetilde{E}_{CV}(u_*) = \int_0^1 \widetilde{E}_{CV}(u_*^t)\, dt$. After putting $\widetilde{E}_{CV}(u_*)$ and $\widetilde{E}_{CV}(u_*^t)$ on one side, we get

$$\int_0^1 (\widetilde{E}_{CV}(u_*^t) - \widetilde{E}_{CV}(u_*))\, dt = 0 \tag{4.13}$$

Since u_* is the minimum value, $\widetilde{E}_{CV}(u_*^t) \geq \widetilde{E}_{CV}(u_*)$. It is easy to observe that $\widetilde{E}_{CV}(u_*) = \widetilde{E}_{CV}(u_*^t)$ for almost all t. If not, the formulation (4.13) can not hold. Thus $u_*^t = \underset{u \in K}{\operatorname{arginf}} \widetilde{E}_{CV}(u)$ and $u_*^t \in \{0, 1\}$ is the minimum of $\underset{u \in \{0,1\}}{\widetilde{\mathrm{E}}_{\mathrm{CV}}(\mathrm{u})}$. □

4.1.2 Chan-Vese segmentation model with priori given information

If regions Ω_f and Ω_b are labeled as foreground and background respectively in advance, the Chan-vese model become

$$\begin{aligned} E_{CV}(u, c) &= J(u) + \frac{\lambda}{2} \left\{ \langle 1 - u, (I - c_0)^2 \rangle + \langle u, (I - c_1)^2 \rangle \right\} \\ \text{s.t.} \quad & u_f(x) \leq u \leq u_b \end{aligned} \tag{4.14}$$

where the definition of $u_b(x)$ and $u_f(x)$ can be found in (3.20). The characteristic function u belongs to the following space

$$B = \{u \in \{0, 1\} \mid u_f(x) \leq u(x) \leq u_b(x)\ \forall x \in \Omega\}. \tag{4.15}$$

which makes the problem (4.14) become a non-convex problem. Next we prove that when c is fixed, this problem can be solved by relaxing the constraint on u to allow $u \in G$ where

$$G = \{u \in BV(\Omega) \mid u_f(x) \leq u(x) \leq u_b(x)\ \forall x \in \Omega\}. \tag{4.16}$$

After loosing the constraint on u, the problem becomes a convex problem which is easy to be solved. Like the case in non-supervised Chan-vese model, the minimisation of E_{CV} over G is equivalent to the minimisation of $\widetilde{E}_{CV}$ over G.

Lemma 4.1.2 (The Truncation Lemma II). *If* $u_* = \underset{u \in G}{\operatorname{arginf}} \widetilde{E}_{CV}(u)$ *then* $u_*^t = \underset{u \in B}{\operatorname{arginf}} \widetilde{E}_{CV}(u)$ *for almost all* $t \in [0, 1]$ *where* $u_*^t(x)$ *is defined as*

$$u_*^t(x) = \begin{cases} 1 & \textit{if } u_*(x) > t, \\ 0 & \textit{otherwise} \end{cases} \qquad \forall x \in \Omega.$$

Proof. We can use the same way as the Lemma 4.1.1 to prove that $\widetilde{E}_{CV}(u_*) = \widetilde{E}_{CV}(u_*^t)$ for almost all t. Thus we have that $u_*^t = \underset{u \in G}{\operatorname{arginf}} \widetilde{E}_{CV}(u)$. After the thresholding, the given background and the given foreground still remain what they are set in advance, thus $u_*^t \in B$ and it solves $\underset{u \in B}{\operatorname{arginf}} \widetilde{E}_{CV}(u)$. □

4.2 Local region-based Chan-Vese model

In this section, we present the Local region-based Chan-Vese model [5] which considers the image local characteristics. Before introducing the LRCV model, we define the local mean intensity $d = [d_0, d_1]$ as

$$\begin{cases} d_0(x) = \dfrac{\int_\Omega g_k(x-y)I(y)(1-u(x))\ dy}{\int_\Omega g_k(x-y)(1-u(x))\ dy} \\ d_1(x) = \dfrac{\int_\Omega g_k(x-y)I(y)u(x)\ dy}{\int_\Omega g_k(x-y)u(x)\ dy} \end{cases} \tag{4.17}$$

where g_k is a Gaussian kernel function with size k, I is the image which we are dealing with and $u(x)$ is the characteristic function of region Σ.

In fact, the CV model segments an image by adding pixels closer to foreground to foreground and pixels closer to background to background. The difference between the intensity of one pixel and the global mean intensity measure the distance between the pixel and the foreground and the distance between the pixel and the background . However, in an image with inhomogeneous intensity, the further one region is from one pixel, the less relationship the region has with the pixel. Thus, using the difference between one pixel and the local mean intensity to measure the distance between the pixel and foreground and between the pixel and the background is much more accurate than using the global mean intensity. Then we should replace the global mean intensity $c = [c_0, c_1]$ in CV model by the local mean intensity $d = [d_0, d_1]$ and the energy functional of the local region-based Chan-Vese model is obtained as following

$$E_{LRCV}(u, c) = J(u) + \frac{\mu}{2}\left\{\langle 1-u, (I-d_0)^2\rangle + \langle u, (I-d_1)^2\rangle\right\}, \tag{4.18}$$

where $J(u)$ plays the same role here as in the CV model to control the length of the region Σ and μ is the wight of the fitting term like the weight λ in CV model. Because of the localization property of the kernel function g_k, the influence of the value of $I(y)$ decreases and approaches zero as the point y goes away from point x. The value of $d = [d_0, d_1]$ is determined by the intensity values in the small neighborhood around point x. If we choose the Gaussian kernel function as an averaging filter with infinity window size, $[d_0, d_1]$ becomes the global mean intensity $c = [c_0, c_1]$ in the Chan-Vese model. Then the LRCV model degenerates to the CV model.

4.3 Regional adaptive active contour model

However, the local region based Chan-Vese model is sensitive with the initialization for the reason that this model only considers the local intensity information. In order to overcome this practical disadvantage, Xing Hui, Peng Yali [6], proposed a new model including the regularization term, the local energy term and the global energy term:

$$\begin{aligned} E_{RAAC}(c,d,u) = J(u) &+ \frac{\lambda(k)}{2}\left\{\langle 1-u,(I-c_0)^2\rangle + \langle u,(I-c_1)^2\rangle\right\} \\ &+ \frac{\mu(k)}{2}\left\{\langle 1-u,(I-d_0)^2\rangle + \langle u,(I-d_1)^2\rangle\right\} \end{aligned} \tag{4.19}$$

where $c = [c_0, c_1]$ and $d = [d_0, d_1]$ are the same as in the Chan-Vese model and the Local region-based Chan-Vese model respectively and k means the iteration number. Here, the weight function $\lambda(k)$ is a non-increasing function while the weight function $\mu(k)$ is a non-decrasing function.

At the beginning of the evolution, the global term much bigger than the local term yields a good initial segmentation of the image quickly by utilizing the advantages of the Chan-Vese mode: the fast convergence and non-sensitivity to the initialization. In the later stage of the process, the local energy term overwhelming the global term can locate the object of interest precisely.

Chapter 5

The proposed method

In this chapter, the supervised continuous max-flow method is proposed to solve the regional adaptive active contour model numerically. We largely follow the discussion in [9]. For the sake of notation simplicity we only construct the mathematical algorithm for the grey scale images, but the algorithm can be extended to colour images easily.

Because of the given priori information which is that we have labeled one region Ω_f as foreground and one region Ω_b as background, the regional adaptive active contour model should be revised as

$$\begin{aligned} E_{RAAC}(c,d,u) = J(u) + &\frac{\lambda(k)}{2}\left\{\langle 1-u,(I-c_0)^2\rangle + \langle u,(I-c_1)^2\rangle\right\} \\ &+\frac{\mu(k)}{2}\left\{\langle 1-u,(I-d_0)^2\rangle + \langle u,(I-d_1)^2\rangle\right\} \\ \text{s.t.} \quad u_f(x) \le u(x) \le u_b(x) & \qquad\qquad \forall x \in \Omega. \end{aligned}$$

The idea of minimising the revised RAAC model is that we minimise the energy function with respect to one variable at one time after fixing the

(a) (b)

Figure 5.1: a is the original figure. The region inside the green line in b is the region Ω_f while the region Ω_b is the part inside the red line.

others. In the first part, we minimise the energy function with the respect to the characteristic function u while keeping c and d fixed. Then after fixing u and d, we minimize with the respect to c. At the last step, we just update d by its definition to minimise the cost function in the revised RAAC model.

5.1 Minimising w.r.t. u

We rearrange the regional adaptive contour model as the following

$$\begin{aligned} E_{RAAC}(c,d,u) &= J(u) + \frac{\lambda(k)}{2}\left\{\langle 1-u,(I-c_0)^2\rangle + \langle u,(I-c_1)^2\rangle\right\} \\ &+ \frac{\mu(k)}{2}\left\{\langle 1-u,(I-d_0)^2\rangle + \langle u,(I-d_1)^2\rangle\right\} \\ &= J(u) + \left\langle u, \frac{\lambda(k)}{2}(I-c_1)^2 + \frac{\mu(k)}{2}(I-d_1)^2\right\rangle \\ &+ \left\langle 1-u, \frac{\lambda(k)}{2}(I-c_0)^2 + \frac{\mu(k)}{2}(I-d_0)^2\right\rangle \\ &\text{s.t.} \quad u_f(x) \le u(x) \le u_b(x) \qquad \forall x \in \Omega. \end{aligned} \tag{5.1}$$

The difference between (5.1) and (4.14) is only the weight of the fitting term and the quadratic terms. However, the quadratic terms in and (4.14) and (5.1) are only numbers when keeping $c = [c_0, c_1]$ and $d = [d_0, d_1]$ fixed. Thus these two problems with fixing c and d can be regarded as one kind of problems. In chapter 4, we have proven that this kind of problems can be solved by relaxing the constraint on u from $u(x) \in B$ to $u \in G$ and truncating the solution by almost every number t between 0 and 1. By deleting those terms in (5.1) which does not depend on u, we get

$$\begin{aligned} \widetilde{E}_{RAAC}(c,d,u) = J(u) &+ \left\langle u, \frac{\lambda(k)}{2}(I-c_1)^2 + \frac{\mu(k)}{2}(I-d_1)^2\right\rangle \\ &- \left\langle u, \frac{\lambda(k)}{2}(I-c_0)^2 + \frac{\mu(k)}{2}(I-d_0)^2\right\rangle \end{aligned}$$

Now the formula for updating u is following

$$u^{k+1} = \underset{u\in G}{\operatorname{argmin}} \widetilde{E}_{RAAC}(u, c^k, d^k). \tag{5.2}$$

If we put $C(x) = 1$, $C_s(x) = \frac{\lambda(k)}{2}(I-c_0)^2 + \frac{\mu(k)}{2}(I-d_0)^2$ and $C_t(x) = \frac{\lambda(k)}{2}(I-c_1)^2 + \frac{\mu(k)}{2}(I-d_1)^2$, minimizing $\widetilde{E}_{RAAC}$ over set G become the min-cut problem (3.26). The constant $C(x)$, which we call α from now on, measures the importance of the region having a short boundary. From the expressions of C_s and C_t, we can see that C_s shows how big the difference between the pixel and the background is and C_t is the other way around. Here, when measuring the difference between the pixel and background or foreground, we should take both the global intensity information and local intensity information into account. When C_s is larger than C_t, it means that this pixel is closer to the foreground than the background, then we should label this pixel as foreground. On the contrary, the pixel with a smaller C_s should be labeled as the background.

Now the problem (3.26) is equivalent to another problem (3.21). Thus we can construct an algorithm for (3.21) to minimise $\widetilde{E}_{RAAC}$ with respect to u. The problem (3.21) is a constraint optimization problem with equality constraints and inequality constraints. Thus we can use the augmented Lagrangian method to solve this problem. The augmented Lagrangian function is following

$$L_a(p_s, p_t, p, u) = \int_\Omega u_b(x)p_s(x) - u_f(x)p_t(x)\ dx + \int_\Omega u(\mathrm{div}p - p_s + p_t)\ dx - \frac{\gamma}{2}\left\|\mathrm{div}p - p_s + p_t\right\|^2 \tag{5.3}$$

where γ is a positive number. When updating p_s, p_t and p, we also need to fulfill the inequality constraints (3.6), (3.7) and (3.8). Here we have four variables: p, p_s, p_t and u. Thus we need to use the so-called alternating directions method of multipliers [15] which is that we optimise the function with one variable at a time while keeping the others fixed and repeat steps until convergence:

1. Optimising p by fixing the others:

$$p^{k+1} = \underset{\|p\|_\infty \le \alpha}{\mathrm{argmax}}\ L_a(p_s^k, p_t^k, p, u^k). \tag{5.4}$$

where α representing $C(x)$ is a constant. We can ignore the first integral of (5.3) which is independent of p. The last two terms can be rewritten as following:

$$\begin{aligned}
&u^k(\mathrm{div}p - p_s^k + p_t^k)\ dx - \frac{\gamma}{2}(\mathrm{div}p - p_s^k + p_t^k)^2 \\
&= -\frac{\gamma}{2}\left((\mathrm{div}p - p_s^k + p_t^k)^2 - \frac{2u^k}{\gamma}(\mathrm{div}p - p_s^k + p_t^k)\right) \\
&= -\frac{\gamma}{2}\left((\mathrm{div}p - p_s^k + p_t^k - \frac{u^k}{\gamma})^2 - \left(\frac{u^k}{\gamma}\right)^2\right)
\end{aligned}$$

where the last step comes from completing the square. The term $\left(\frac{u^k}{\gamma}\right)^2$ does not depend on p ,thus can be removed. Then the problem (5.4) is equivalent to the following

$$\begin{aligned} p^{k+1} &= \underset{\|p\|_\infty \leq \alpha}{\text{argmax}} \; -\frac{\gamma}{2}\left\|\text{div}p - M^k\right\|^2 \\ &= \underset{\|p\|_\infty \leq \alpha}{\text{argmin}} \; \frac{\gamma}{2}\left\|\text{div}p - M^k\right\|^2 \end{aligned} \tag{5.5}$$

where $M^k = p_s^k - p_t^k + \frac{u^k}{\gamma}$. The exact solution to this problem can not be found because of the divergence term. However we can use the gradient descent method to find the numerical solution. Because p in $G(p) = \int_\Omega \frac{\gamma}{2}(\text{div}p - M^k)^2\, dx)$ is a function, the Gateaux derivative w.r.t. p should be found. According to the definition of Gateaux derivative, we have

$$\begin{aligned} G'(p;d) &= \lim_{h\to 0} \frac{G(p+hd) - G(p)}{h} \\ &= \lim_{h\to 0} \frac{\gamma}{2h}\int_\Omega \left[(\text{div}p + h\text{div}d - M^k)^2 - (\text{div}p - M^k)^2\right]\, dx \end{aligned}$$

Because of the fact that $a^2 - b^2 = (a-b)(a+b)$, the equation above becomes the following

$$\begin{aligned} G'(p;d) &= \lim_{h\to 0} \frac{\gamma}{2h}\int_\Omega h\text{div}d(2\text{div}p + h\text{div}d - 2M^k)\, dx \\ &= \lim_{h\to 0} \frac{\gamma}{2}\int_\Omega \text{div}d(2\text{div}p + h\text{div}d - 2M^k)\, dx \\ &= \gamma\int_\Omega \text{div}d(\text{div}p - M^k)\, dx. \end{aligned}$$

The product rule for differentiation gives us

$$\text{div}d(\text{div}p - M^k) = \text{div}(d(\text{div}p - M^k)) - d\cdot\nabla(\text{div}p - M^k),$$

hence

$$G'(p;d) = \gamma\int_\Omega \text{div}(d(\text{div}p - M^k))\, dx - \gamma\int_\Omega d\cdot\nabla(\text{div}p - M^k)\, dx$$

After using the divergence theorem to the first term on the right hand, we get

$$G'(p;d) = \gamma\int_{\partial\Omega} d(\text{div}p - M^k)\cdot v\, dS - \gamma\int_\Omega d\cdot\nabla(\text{div}p - M^k)\, dx \tag{5.6}$$

where v is the outward unit normal vector along $\partial\Omega$. The first term on the right hand in (5.6) becomes zero after we impose $(\text{div}p - M^k)\cdot v = 0$ on $\partial\Omega$. Because $G'(p;d) = \langle G'(p), d\rangle$, the Gateaux derivative $G'(p)$ is obtained as $-\gamma\nabla(\text{div}p - M^k)$. The gradient method with step size ϵ gives us the approximate solution as following

$$p^{k+1} = p^k - \epsilon G'(p^k) = p^k + \widetilde{\gamma}\nabla(\text{div}p^k - M^k) \tag{5.7}$$

where $\widetilde{\gamma} = \epsilon\gamma$. Besides, a procedure of projection on p^{k+1} to a specified convex set is required in order to satisfy the inequality constraint (3.8). Then the final formula for updating p is following

$$p^{k+1} = \text{proj}_\alpha(p^k + \widetilde{\gamma}\nabla(\text{div}p^k - M^k)) \tag{5.8}$$

where proj_α is the projection onto the convex set $S_a = \{q \mid \|q\|_\infty \leq \alpha\}$.

2. Optimising p_s by fixing other variables:

$$p_s^{k+1} = \underset{p_s(x)\leq C_s(x)}{\text{argmax}} \; L_a(p_s, p_t^k, p^{k+1}, u^k). \tag{5.9}$$

From E.q. (5.3), we can see that flows $p_s(x)$ are independent of each other, thus we can maximise (5.3) by maximising at each point. Because the constraint on p_s is concave, the exact maximum can be found by just setting the derivative of $l_{a.s} = u_b p_s + u(\text{div}p^{k+1} - p_s + p_t^k) - \frac{\gamma}{2}(\text{div}p^{k+1} - p_s + p_t^k)^2$ w.r.t p_s to be zero:

$$\begin{aligned} p_s^*(x) &= \underset{p_s(x)}{\text{argmax}} \; l_{a.s} \\ \leftrightarrow \;\; & u_b - u^k + \gamma(\text{div}p^{k+1} - p_s^* + p_t^k) = 0 \\ \leftrightarrow \;\; & p_s^* = \frac{u_b}{\gamma} + \text{div}p^{k+1} + p_t^k - \frac{u^k}{\gamma}. \end{aligned}$$

Besides we also need to fulfill the inequality constraint on $p_s(x)$, so we choose the smaller value between $\text{div}p^{k+1} + p_t^k - \frac{u^k}{\gamma}$ and C_s as the optimal solution. In order to make the notation short, we define $N^k = \text{div}p^{k+1} + p_t^k - \frac{u^k}{\gamma}$. Then the formula for updating p_s is given by

$$p_s^{k+1} = \min\left\{\frac{u_b}{\gamma} + N^k, C_s\right\}. \tag{5.10}$$

3. Optimising p_t by fixing other variables:

$$p_t^{k+1} = \underset{p_t(x)\leq C_t(x)}{\text{argmax}} \; L_a(p_s^{k+1}, p_t, p^{k+1}, u^k). \tag{5.11}$$

Like p_s, p_t does not depend on each other and the integrand is concave w.r.t p_t. We can use the same approach as p_s to find the maximum:

$$\begin{aligned}
p_t^*(x) &= \underset{p_t(x)}{\text{argmax}}\ l_{a.t} \\
&\leftrightarrow -u_f + u^k - \gamma(\text{div}p^{k+1} - p_s^{k+1} + p_t^*) = 0 \\
&\leftrightarrow p_t^* = -\text{div}p^{k+1} + p_s^{k+1} + \frac{u^k - u_f}{\gamma}
\end{aligned}$$

where $l_{a.t} = -u_f p_s^{k+1} + u(\text{div}p^{k+1} - p_s^{k+1} + p_t) - \frac{\gamma}{2}(\text{div}p^{k+1} - p_s^{k+1} + p_t)^2$. Here we define $O^k = -\text{div}p^{k+1} + p_s^{k+1} + \frac{u^k - u_f}{\gamma}$. After imposing O^k in the concave set $p_t(x) \leq C_t(x)$, we get the updating formula for p_t as

$$p_t^{k+1} = \min\left\{O^k, C_t\right\} \tag{5.12}$$

4. According to the augmented Lagrangian method, the Lagrange multiplier u is updated by one step of gradient descent:

$$u^{k+1} = u^k - \gamma(\text{div}p^{k+1} - p_s^{k+1} + p_t^{k+1}). \tag{5.13}$$

   where γ is the time step which we choose to keep fixed.

Before moving forward to next step, several loops are needed to make sure that u converges.

5.2 Minimising w.r.t. c

Now, after getting the new characteristic function u, we minimise with respect to c:

$$c^{k+1} = \underset{c \in \mathbb{R}^2}{\text{argmin}} E_{RAAC}(u^{k+1}, c, d^k). \tag{5.14}$$

Because problem (5.14) is a quadratic optimisation problem, the solution can be found by setting the gradient of E_{RAAC} w.r.t. c equal to zero. First we set the gradient of E_{RAAC} w.r.t. c_0 equal to zero:

$$\begin{aligned}
&\frac{\partial E_{RAAC}}{\partial c_0}(u^{k+1}, c_0^{k+1}, c_1^k, d^k) = 0 \\
&\leftrightarrow \frac{\lambda(k)}{2}\left\langle(1 - u^{k+1}), -(I - c_0^{k+1})\right\rangle = 0 \\
&\leftrightarrow \left\langle 1 - u^{k+1}, I\right\rangle = \left\langle 1 - u^{k+1}, c_0^{k+1}\right\rangle \\
&\leftrightarrow c_0^{k+1} = \frac{\left\langle 1 - u^{k+1}, I\right\rangle}{\left\langle 1 - u^{k+1}, 1\right\rangle}
\end{aligned} \tag{5.15}$$

The eq. (5.15) means that c_0^{k+1} is the mean intensity value outside the region Σ. Similarly we get

$$\begin{aligned}
&\frac{\partial E_{RAAC}}{\partial c_1}(u^{k+1}, c_0^{k+1}, c_1^{k+1}, d^k) = 0 \\
&\leftrightarrow \frac{\lambda(k)}{2}\left\langle u^{k+1}, -(I - c_1^{k+1})\right\rangle = 0 \\
&\leftrightarrow \left\langle u^{k+1}, I\right\rangle = \left\langle u^{k+1}, c_0^{k+1}\right\rangle \\
&\leftrightarrow c_0^{k+1} = \frac{\left\langle u^{k+1}, I\right\rangle}{\langle u^{k+1}, 1\rangle}
\end{aligned} \tag{5.16}$$

From (5.16) we can see that c_1^{k+1} is the mean intensity value inside the region Σ. Then (5.15) and (5.16) can be summarized as

$$\begin{cases} c_0^{k+1}(x) = \dfrac{\int_\Omega I(y)(1 - u^{k+1}(x))\; dy}{\int_\Omega (1 - u^{k+1}(x))\; dy} \\ c_1^{k+1}(x) = \dfrac{\int_\Omega I(y)u^{k+1}(x)\; dy}{\int_\Omega u^{k+1}(x)\; dy} \end{cases} \tag{5.17}$$

5.3 Updating d

After keeping the new u and c fixed, we use the definition of the local mean intensity to update d:

$$\begin{cases} d_0^{k+1}(x) = \dfrac{\int_\Omega g_k(x-y)I(y)(1 - u^{k+1}(x))\; dy}{\int_\Omega g_k(x-y)(1 - u^{k+1}(x))\; dy} \\ d_1^{k+1}(x) = \dfrac{\int_\Omega g_k(x-y)I(y)u^{k+1}(x)\; dy}{\int_\Omega g_k(x-y)u^{k+1}(x)\; dy} \end{cases} \tag{5.18}$$

After updating u, in fact, not all pixels' local mean intensity will be changed. If the characteristic function u changes in the Gaussian mask of one pixel, d of this pixel will change, if not, d of this pixel remains the same as the last iteration. Thus we can conclude that d only varies near the boundary of the region Σ during the process.

5.4 Updating λ and μ

According to the regional adaptive active contour model [6], the weight of the global fitting term should decrease with the increasing of the iteration number. By fixing the decreasing step t_λ, we get the following mathematical formula:

$$\lambda(k+1) = \lambda(k) - t_\lambda, \qquad k = 0, 1, \dots, \tag{5.19}$$

where $\lambda(0) = T$ and $\lambda(k)$ can not be smaller than L which is a non-negative number.

On the contrary, μ as the weight of the local fitting term should be increased and can not be larger than a threshold. Here we also keep the increasing step t_μ fixed:

$$\mu(k+1) = \mu(k) + t_\mu, \qquad k = 0, 1, \dots, \tag{5.20}$$

where $\mu(0) = 0$ and all μ should be less than a predefined threshold U.

5.5 The main steps of the algorithm

The main steps can be summarised as following:

1. Let the average intensity in Ω_f be c_1 and the average intensity in Ω_b be c_0. Initialize the characteristic function by the following rule:

$$D_b = (I - c_0)^2 \text{ and } D_f = (I - c_1)^2$$
$$u(x) = \begin{cases} 1 & if\ D_b(x) \geq D_f(x) \\ 0 & \text{otherwise} \end{cases} \qquad \forall x \in \Omega.$$

.

2. Computing $c(x)$ and $d(x)$ by their definition. Let $C_s(x) = \frac{\lambda}{2}(I - c_0)^2 + \frac{\mu}{2}(I - d_0)^2$ and $C_t(x) = \frac{\lambda}{2}(I - c_1)^2 + \frac{\mu}{2}(I - d_1)^2$. Set the initial spatial flow $p(x) = 0$, the initial source flow $p_s(x) = min\{C_s(x), C_t(x)\}$ and the initial sink flow $p_t(x) = p_s(x)$.

3. Update $p(x), p_s(x), p_t$ and $u(x)$ according to (5.8), (5.10), (5.12) and (5.13) respectively.

4. Check whether $u(x)$ converges (For fixed $c(x)$ and $d(x)$ fixed). If converge, go to the next step, otherwise return to step 3.

5. Check whether $u(x)$ converges (For changing $c(x)$ and $d(x)$). If converge, stop, otherwise move forward to next step.

6. Update $c(x)$ and $d(x)$ according to (5.17) and (5.18) respectively. Use (5.19) to decrease $\lambda(x)$ and (5.20) to increase $\mu(x)$. Let $C_s(x) = C_s(x) = \frac{\lambda}{2}(I - c_0)^2 + \frac{\mu}{2}(I - d_0)^2$ and $C_t(x) = \frac{\lambda}{2}(I - c_1)^2 + \frac{\mu}{2}(I - d_1)^2$. Return to step 3.

5.6 Implementation issues

Before implementing the algorithm constructed under continuous circumstance in MATLAB, We need to discretize the operators applied in the algorithm. There is no need to discuss the basic mathematical operators such as addition, subtraction, multiplication and division which are done element-wise. However, we need to specify how to discretize the derivative, the divergence and the integral to make sure that they can function correctly in MATLAB code.

5.6.1 Approximating the derivative

Before discretising the derivative, it is logical to look at the formal definition of the derivative:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}, \tag{5.21}$$

When h is positive, this definition leads us to the formula:

$$f'(x_k) \approx f'_+(x_k) = \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}, \tag{5.22}$$

which is also called the forward-difference approximation [21]. when h in (5.21) is negative, we get the backward-difference approximation:

$$f'(x_k) \approx f'_-(x_k) = \frac{f(x_{k-1}) - f(x_k)}{x_{k-1} - x_k} = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}. \tag{5.23}$$

Even though there are other forms of derivative approximation such as central differences, here only these two ways are used. In digital images the discrete point x_k is the image pixel and there are two directions of the derivative denoting by x_1 and x_2. Then, by setting the distance between two pixels as one, the partial derivatives are given as

$$I^{x_1}_+(i,j) = I(i+1,j) - I(i,j) \quad \text{and} \quad I^{x_2}_+(i,j) = I(i,j+1) - I(i,j)$$

or

$$I^{x_1}_-(i,j) = I(i,j) - I(i-1,j) \quad \text{and} \quad I^{x_2}_-(i,j) = I(i,j) - I(i,j-1).$$

Observe that, we can not avoid making use of the pixels outside the image when calculating the derivative for all image pixels no matter which kind of difference is used. For forward difference, we can define $I(m+1,j) = I(m,j), I(i,n+1) = I(i,n)$ or $I(m+1,n) = 0, I(i,n+1) = 0$. For backward differences some of the choices are $I(0,j) = I(1,j), I(i,0) = I(i,1)$ or $I(0,j) = 0, I(i,0) = 0$. In real case, you can use any of the choices listed above or other ways to define those pixels which are needed. No matter what kind of way you are using, the final result will not be changed too much.

5.6.2 Approximating the divergence

The generalized definition of the divergence for a vector $F(F_1, F_2, \ldots, F_n)$ is

$$\operatorname{div}(F) = \nabla \cdot F = \frac{\partial F_1}{\partial x_1} + \frac{\partial F_2}{\partial x_2} + \cdots + \frac{\partial F_n}{\partial x_n}. \tag{5.24}$$

In the digital image case, the flow p only have two dimensions which means that $n = 2$. So the divergence of the flow in image $P(P_1, P_2)$ should be defined as

$$\operatorname{div}(P) = \nabla \cdot P = \frac{\partial P_1}{\partial x_1} + \frac{\partial P_2}{\partial x_2} \tag{5.25}$$

where P_1 and P_2 denotes the flow along the direction x_1 and the direction x_2 respectively. From (5.25), we can see that the divergence of the flow through one pixel in image I is the sum of the gradients in two directions. Because we have described how to approximate the gradient in section 5.6.1, it is easy to get the approximation of divergence. If the forward-difference approximation is used, the divergence of the pixel $I(i, j)$ is following

$$\operatorname{div}(P(i,j)) = P_1(i+1,j) + P_2(i,j+1) - P_1(i,j) - P_2(i,j).$$

where $P(i, j)$ is the two-dimension flow through pixel $I(i, j)$. Similarly, we can use the backward-difference to derive the approximation of divergence as

$$\operatorname{div}(P(i,j)) = P_1(i,j) + P_2(i,j) - P_1(i-1,j) - P_2(i,j-1).$$

From the formulation of the divergence calculated by the backward-difference approximation, we can see that the practical meaning of the divergence of a flow is the amount of water flowing outside the corresponding pixel.

5.6.3 Approximating the integral

In this section, we spend a little bit time on computing the values of integral. One most popular method is trapezoidal rule [22] in which we break up the interval [a,b] into n subintervals of width h and set $x_0 = a,\ x_n = b$:

$$h = \frac{b-a}{n}.$$

Then on each subinterval, we approximate the function $f(x)$ with a straight line which has the same value as the original function at the two endpoints. The reason why we call this method as trapezoidal method is that each object on the subinterval $[x_k, x_{k+1}]$ is trapezoidal. The area of the trapezoidal in interval $[x_k, x_{k+1}]$ is following

$$A_k = \frac{h}{2}(f(x_k) + f(x_{k+1})).$$

Now, we use the sum of all the area of all trapezoidal to estimate the integral:

$$\int_a^b f(x)\ dx \approx \sum_{k=0}^{n-1} \frac{h}{2}(f_k + f_{k+1}) = \frac{h}{2}(f_0 + 2f_1 + \cdots + 2f_{n-1} + f_n),$$

where f_k is short for $f(x_k)$. In our case, the interval length is set as one and the nodes x_k are image pixels. Because the number of pixels is quite large, multiplying f_0 and f_n with 2 does not change the result too much. For simplification, the general trapezoidal rule can be changed as

$$\int_a^b f(x)\ dx \approx h(f_0 + f_1 + \cdots + f_{n-1} + f_n). \tag{5.26}$$

Similarly, we can approximate the two-dimensional integral by the following

$$\int_\Omega I(x_1, x_2)\ dx \approx \sum_{i=1}^{m}\sum_{j=1}^{n} I(i, j). \tag{5.27}$$

5.6.4 Smoothing the two indicator functions

We understand these two indicator functions (3.20) from the perspective of probability. Here, u_f represents the probability of one region to be foreground and $1 - u_b$ represents the probability of the region to be background. Since Ω_f is foreground and Ω_b is background, $u_f = 1$ in Ω_f and $1 - u_b = 1$ in Ω_b.

Since each region in an image is continuous, when one region, Ω_f, is regarded as foreground, then the surrounding area has certain probability to be foreground. The probability of the surrounding area to be foreground will decrease with the distance away from the region, Ω_f. This also applies to Ω_b, and it is logical to smooth these two indicator functions. In this thesis, u_f is smoothed out by using a convolution with a Gaussian function $G(x)$:

$$u_f^s(x) = G(x) \star u_f(x).$$

Likewise, the function, $1 - u_b$, is handled in the same fashion:

$$u_b^s(x) = 1 - G(x) \star (1 - u_b(x)) = G(x) \star u_b(x).$$

Chapter 6

Experimental results

We have applied our method on several different kinds of images and will dispaly some of the results in this section. The weights function for updating λ and μ are chosen differently for different images while $\gamma = 0.3$, $\widetilde{\gamma} = 0.16$, $\epsilon = 1$ and the Gaussian filter used for getting the local information and the Gaussian filter for smoothing those two indicator functions are fixed with $n = 21, \sigma = 5$ and $n = 7, \sigma = 1$ respectively where n is the window size and σ is the standard deviation. The value of λ controls the influence of the global information while the value of μ controls influence of the local information. As λ is decreasing and μ is increasing, the influence from the local information dominates the whole segmentation process. Moreover, the importance of the shortness of the boundary of the region Σ is inversely proportional to the sum of λ and μ. When $\lambda + \mu$ is large, $J(u)$ will not contribute much to the energy function and thus allow the length of the boundary of Σ to be relatively large. On the contrary, if $\lambda + \mu$ is relatively small, we will get the region Σ with a relatively short boundary. In order to control the length of the contour of the foreground Σ, the parameter μ which does not decrease during the process should be less than a threshold.

In the algorithm, there are two loops: outer loop and inner loop. The inner loop uses the max-flow method to minimize the energy function with fixed c and d w.r.t. u. So we divide this chapter into two parts: the first section talks about the choice of the max-flow iteration number; the second part is our test results of the Region adaptive active contour model.

6.1 Number of max-flow iterations

As Kvile points out in [9], there is no point in actually minimising w.r.t. u. Experiments on different images show that the result will be almost the same

n_{inner}	Time	n_{outer}
1	6.2	111
2	3.2	56
3	3.41	55
5	2.7	42
15	2.9	32
25	3.5	31
50	5.1	33
80	7.2	32
100	8.2	32

Table 6.1: Computing time in seconds and number of outer iterations needed for different number of maximum inner iterations on supervised continuous max-flow.

regardless of the maximum number of iterations. However, the algorithm with different max-flow iterations number needs different computing time. From now on, we represent the maximum inner iteration number and the outer loop number by n_{inner} and n_{outer}.

One of the experiments was done on the image with intensity inhomogeneities in figure 5.1(a). Here we make the n_{inner} change and add a stopping criteria in case of convergence of u. From table 6.1, we see that n_{inner} being 5 gave the shortest computation time. As n_{inner} is decreased, n_{outer} will increase. On the contrary, as we increase the n_{inner}, n_{outer} will decrease.

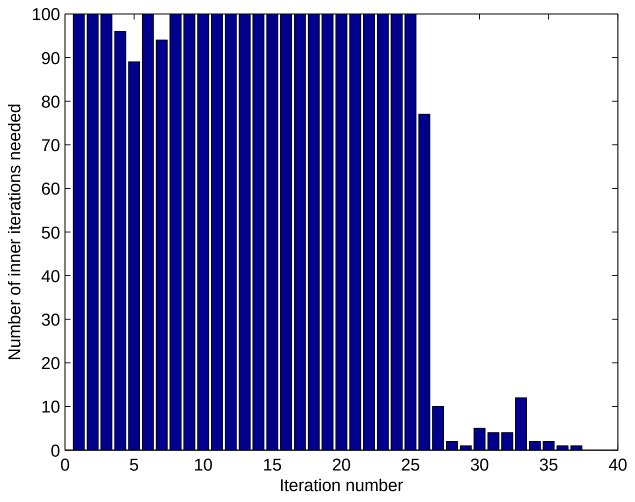


Figure 6.1: Number of iterations until convergence of u for each outer iteration

Since there are two processes in the outer loop involving convolution which is very time-consuming, increasing the outer loop number n_{outer} will up the time of the whole process substantially. Thus the inner iteration number n_{inner} should not be too small. On the other hand, n_{inner} should not be too large. From figure (6.1), we see that it is in the beginning and the mid-term that a large number of iterations are needed to minimise w.r.t. u, but at this point it is not possible to get a good segmentation anyhow, because the weights λ and μ are still changing and the capacities on flows which control the inner loop are entirely wrong. Therefore, using a large n_{inner} to make u converge is pointless. At the later stage when λ and μ are stable, only few iterations are needed to minimise w.r.t. u. Like the case in figure (5.1(a)), experiments on other images show that choosing 5 for the maximum number of inner loop iterations can give us good results within a relatively short time.

6.2 Trials on some images

At first, we state that in the original image the region inside the green line is the given foreground and the given background is inside the red line. Figure 6.2 illustrates how the maximum value of μ affects the segmentation result. In the RAAC model, the decease step t_λ and the increase step t_μ are chosen as 1 and 2 separately and L is set as 1. The result in 6.2(b) is obviously wrong, because the CV model only considering the global intensity information can just divide the initial images into two parts: one is brighter and the other one is darker. The accurate position of the object of interest cannot be found. Even though the result in figure 6.2(c) is not good enough, the contour line C is moving towards the boundary of the right part because the local force is working. With the increasing of U, the result is getting better and better. When U reaches 50, the contour line of u matches the boundary of the object of interest perfectly.

The largest value of the local weight μ should be large enough, otherwise u will converge too quickly because of the influence from the global force. This fact can be observed in figure 6.3. In RAAC model, we set $t_\lambda = 1$, $t_\mu = 2$ and $L = 1$. From figure 6.3(c) and 6.3(d), we can see that there is no big difference between the results from the RAAC model and the CV model in 6.3(b), the reason is that the global force is so strong that the local force does not affect the whole process too much. However, there are some details changed because of the local term. The contour lines in results from RAAC model stick with the boundary of the vessel more tightly especially at the end of the right part and the part below. Like the last case, as we increase U, the contour line will approach the true boundary of the object of interest. Thus

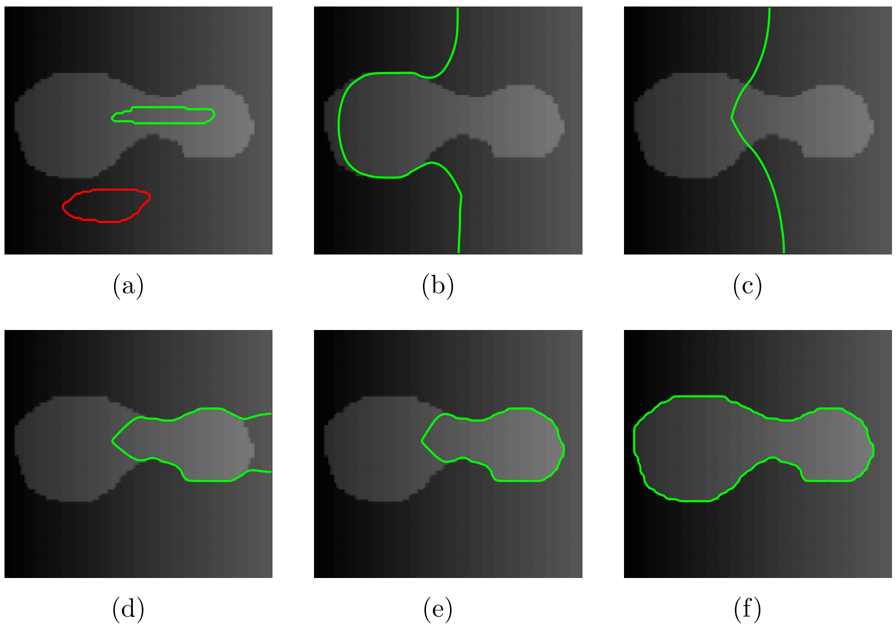


Figure 6.2: First row: Initial image,the result from CV model and the result from RAAC model with $T = 10, U = 10$. Second row: Results from RAAC model with $T = 10, U = 20$,$T = 10, U = 30$ and $T = 10, U = 50$

the choice of U should be made carefully, otherwise the contour line will stop at the wrong place. Because the intensity of one pixel is much closer to their local information than their global information, $\langle 1-u, (I-d_0)^2\rangle + \langle u, (I-d_1)^2\rangle$ is much smaller than $\langle 1-u, (I-c_0)^2\rangle + \langle u, (I-c_1)^2\rangle$. Thus the final weight of the local force U should be much greater than the initial weight of the global force.

The highest value of μ should not be too big, otherwise, some small regions will be included in the contour line. There are two reasons behind this: one is that strong local force can detect weak edges; the other one is that when the weight of the fitting term is large, the length of the contour line C is allowed to be relatively large. As 6.4(b) shows, some small regions with weak edges are detected by RAAC model. When the value of U is decreased, the RAAC model can find the whole vessel without including those vague vessels from the background. Other parameters used in RAAC model are $T = 20, t_\lambda = 1, t_\mu = 8$ and $L = 0$.

The value of the increase step t_μ needs to be chosen carefully. If t_μ is too small, the global term which dominating the whole energy function will make u converge too early. The contour lines in 6.5(b) and 6.5(c) only cover one part of the vessel. After increasing t_μ to be 10, the contour line

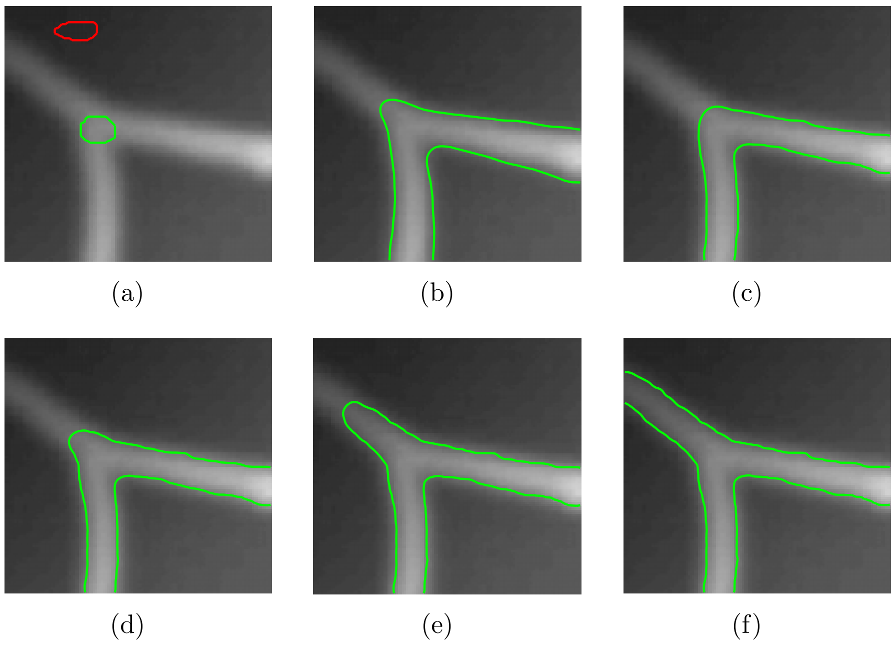


Figure 6.3: First row: Initial image, the result from CV model and the result from RAAC model with $T = 10, U = 50$. Second row: Result from RAAC model with $T = 10, U = 100$, $T = 10, U = 150$ and $T = 10, U = 200$.

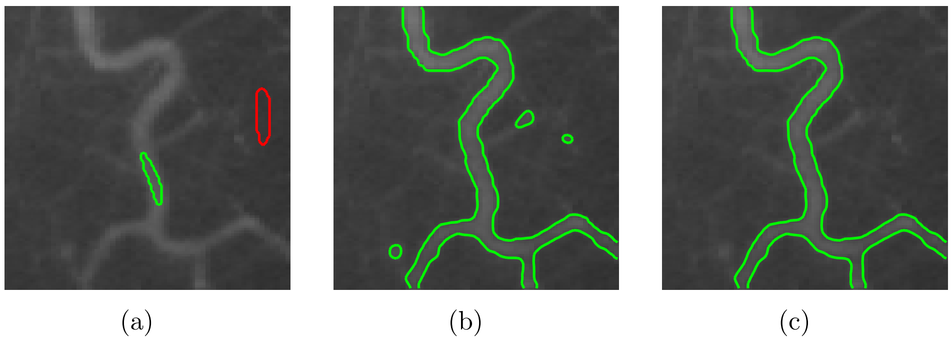


Figure 6.4: From left to right: The initial image, the result from RAAC with $U = 460$ and the result from RAAC model with $U = 360$.

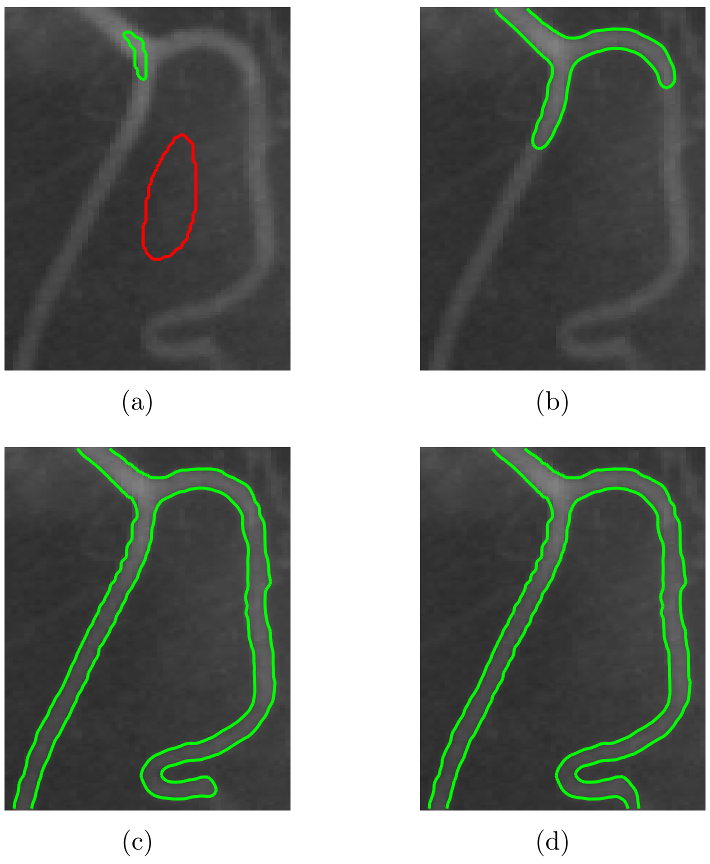


Figure 6.5: From left to right: The initial image, the result from the RAAC model with $t_\mu = 0.5, 5, 10$.

matches with the boundary of the vessel. Other parameters are following: $T = 20, t_\lambda = 1, L = 2$ and $U = 330$.

Next, we use an example to show the advantage of our RAAC model and point out the disadvantage of CV model and LRCV model. As we can see from figure 6.6(b), many edges can not be detected because CV model only considers the global information. The result in figure 6.6(c) is much better because of the local property of the LRCV model. However, this model only considering the local information is sensitive to the initial position of u. It is very likely that the contour line of u converges at a local minimum. The new model named RAAC inherits the advantages of the CV model which are that the speed of convergence is quick and the sensitivity to initialization is not high and the advantage of the LRCV model which is that the local information can localize the accurate location of the object of interest. In figure 6.6(d) and figure 6.6(e), both results are much better than the results

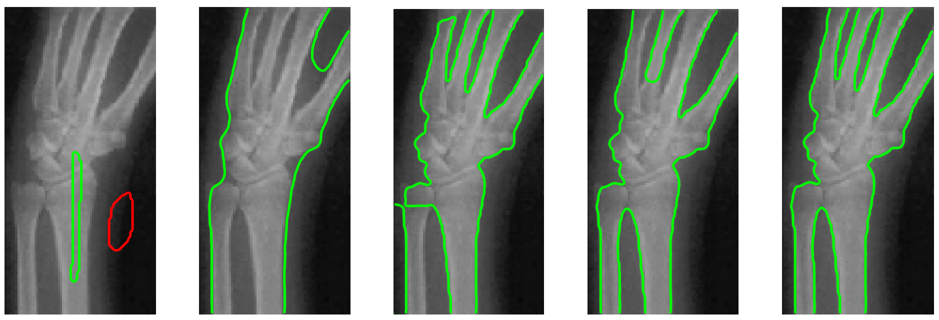
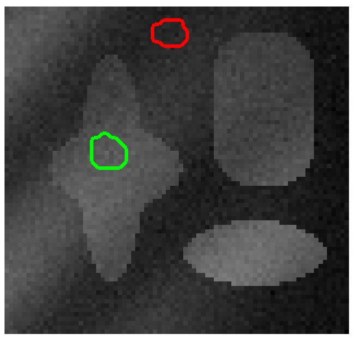
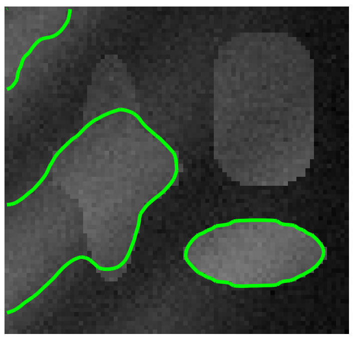


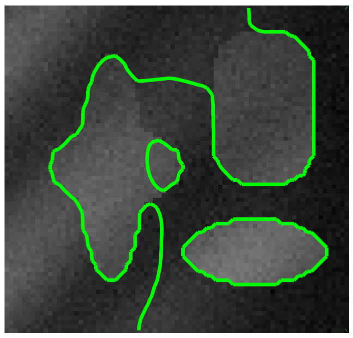
Figure 6.6: From left to right: The initial image, result from the CV model with $\lambda = 20$ and result from the LRCV model with $\mu = 120$. The second row: The result from the RAAC model with $L = 0$ and the result from the RAAC model with $L = 2$.

from CV model and LRCV model. Here the parameters are $T = 20$, $t_\lambda = 1$, $t_\mu = 2$. The only difference between figure 6.6(d) and figure 6.6(e) is the value of L. Sometimes we need to set L as a positive number to keep certain global property. If $L = 0$, when the weight of the global term become L, the RAAC model without global property is very likely to converge at a wrong position.

The object of interest in all the images which we have dealt with is one connected region. Now we pay our attention to a special case in which the object of interest includes three separated parts. Figure 6.7(b) shows the result from CV model. Notice that the CV model only divided the original image into two parts: one is brighter and the other one is darker. Hence the result is not segmented rightly. In figure 6.7(c), the LRCV model detects almost all the edges including some weak edges in the image. However, because of its sensitivity to the initialization, the contour line gets stuck at a local minimum during the process. Besides, no matter how adjusting the inputs in CV model and LRCV model, we can not still get a good result. After we chose the right parameters for our RAAC model, it is easy to get a perfect result shown in figure 6.7(d). Thus the number of parts which the object of interest includes in the image does not influence the performance of our RAAC model. What is important here is that we need to find the right parameters used in RAAC model. We use this example to show how the value of T can affect the result of the RAAC model. When T is too big, at the early stage of the process the contour line will stop at the position shown in figure 6.8 because of the large influence from the global term. After decreasing T to 25, we get the right result in 6.7(d).


(a)


(b)


(c)

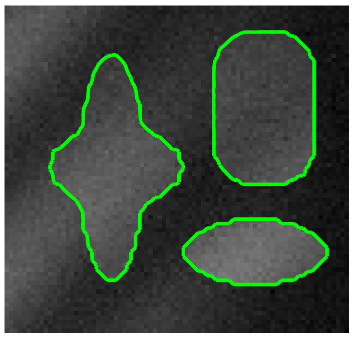

(d)

Figure 6.7: The first row: The initial image and the result from CV model with $\lambda = 20$. The second row: the result from LRCV model with $\mu = 120$ and the result from RAAC model with $L = 0, t_\lambda = 1, t_\mu = 3$ and $T = 20, U = 120$.

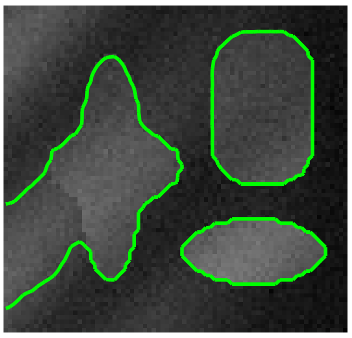


Figure 6.8: The result from the RAAC model with $T = 50$

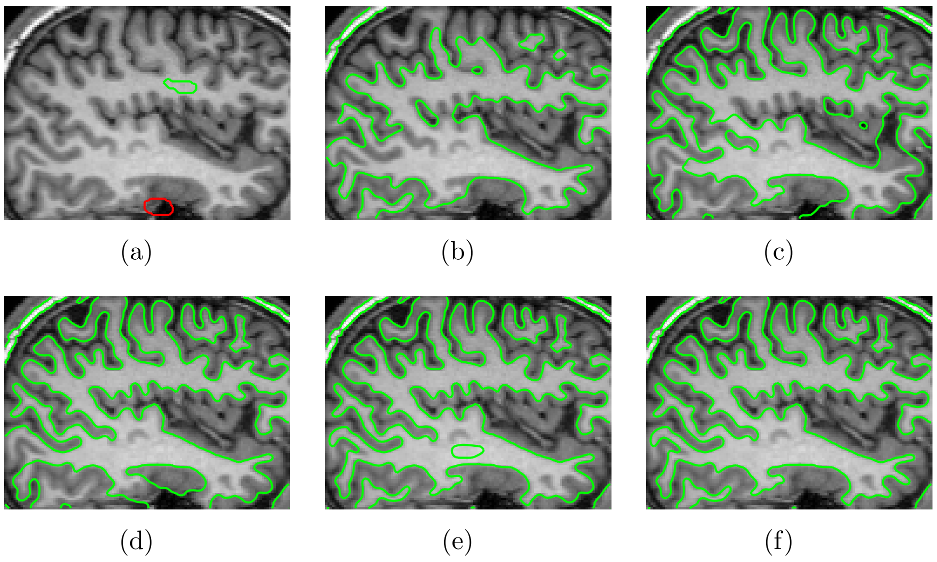
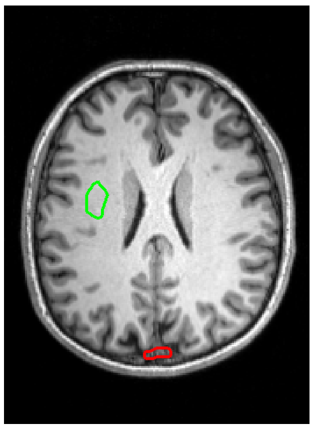


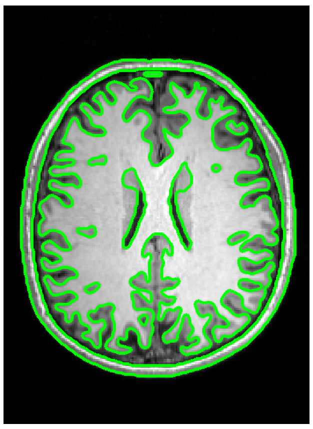
Figure 6.9: The first row: Initial image, the result from the CV model and the result from the LRCV model. The second row: From left to right: the result from RAAC model with $t_\lambda = 7, 5, 1$.

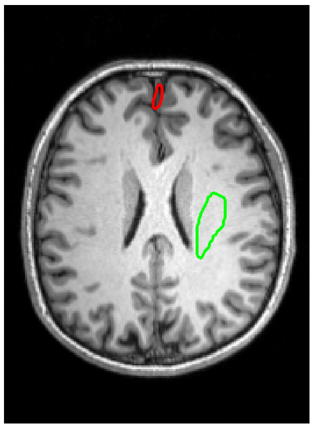
The decrease step t_λ should not be too big, because if λ decreases too quickly, the energy function dominated by the local force will act like LRCV model. In 6.9(d) and 6.9(e), the contour lines have some small circles which are caused from the local property of the local term. After we set a proper t_λ in order to take full advantage of the global property from the global term, the RAAC model gives a better result shown in 6.9(f). The weight in CV and LRCV model is 20 and 42 respectively. Other parameters in RAAC model are $T = 25, t_\mu = 3, L = 0$ and $U = 42$.

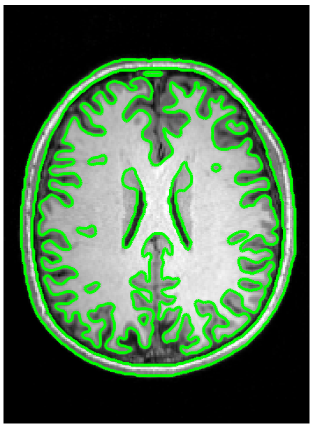
The next example is provided to show that the proposed model is stable with respect to priori given information. The parameters used in this example are $L = 0, t_\lambda = 1, T = 25, t_\mu = 3, U = 120$. The image which we want to segment here is a brain image. Our goal is to segment the white matter from the image. As we can see from figure 6.10, the segmentation results from quite different given information are almost the same. At the beginning of the segmentation process, the global force which is not sensitive to the initialization from the RAAC model dominates the evolution of the contour line, so the difference from the different given information will be removed in the beginning of the process. In our experience, as long as we label given foreground and given background rightly, the result of the RAAC model is quite good.

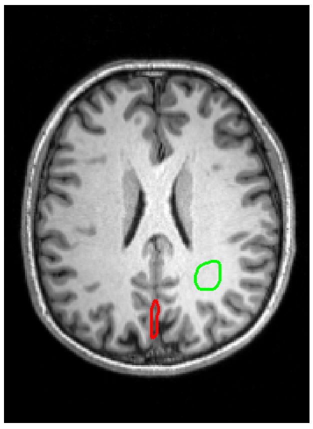
At last, we apply our proposed method on one brain image and one finger


(a)


(b)


(c)


(d)


(e)

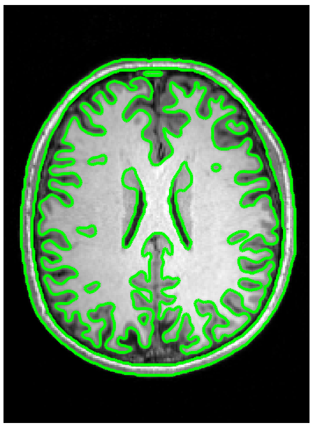

(f)

Figure 6.10: The first column are the images with different priori given information, the second column are the corresponding segmentation results.

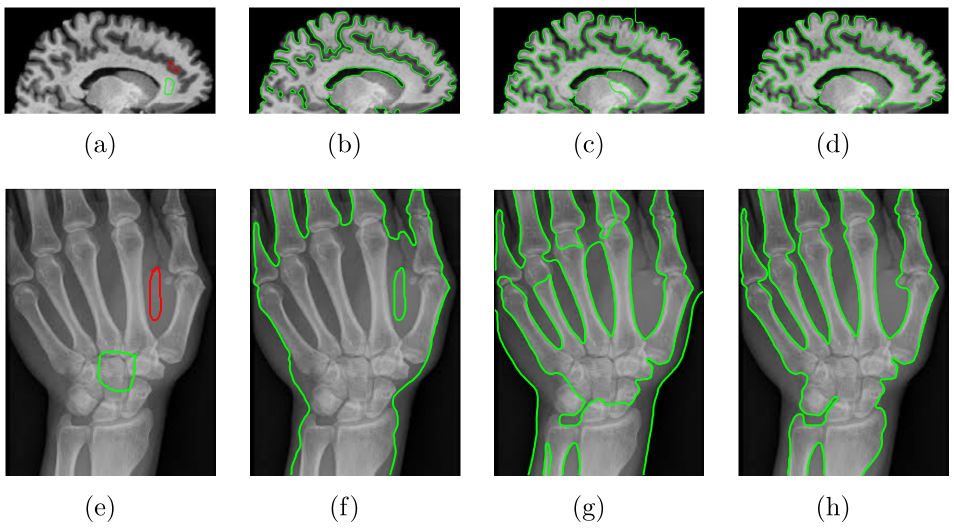


Figure 6.11: First column: The initial images with given priori given information. Second column: Corresponding segmentation results of CV. Third column: Corresponding segmentation results of LRCV. Fourth column: Corresponding segmentation results of RAAC.

image to show the advantages of this method. For the brain image, the weight of the fitting term in CV model and in LRCV model is 17 and 52 respectively, parameters in RAAC model are $L = 0, t_\lambda = 1, T = 25, t_\mu = 3, U = 52$. For the finger image, we use $25, 50$ as the weight of the fitting term in CV and LRCV respectively and chose $L = 3, t_\lambda = 1, T = 25, t_\mu = 3, U = 85$ for the RAAC model. As shown in the second column of figure 6.11, the results are too rough without detecting some weak edges. In the third column of figure 6.11, even though LRCV model can find some weak edges, the contour line is stuck in a wrong postion which results in lots of false edges. We can see from the fourth column of figure 6.11 that the RAAC model completes the segmentation mission perfectly.

Chapter 7

Summary and conclusion

In this work a new model called regional adaptive active contour model is proposed for segmentation of images with intensity inhomogeneities. This model is a combination of Chan-Vese model and Local region-based Chan-Vese model. At the beginning of the segmentation process, the global force gives us a rough but good initial segmentation quickly. In the later phase of the evolution, the local force can localize the exact position of the object. This model success in inheriting the advantage of the CV model: non-sensitivity to the initialization and the advantage of the LRCV model: the ability to locate the edges in images with intensity inhomogeneities.

This model is reformulated as RAAC model with priori given information which is marking one region as foreground and one region as background manually. Then the energy function was minimized with one variable at a time. Minimization of the characteristic function of the foreground u is achieved by regarding this problem as a min-cut problem which can be solved by the supervised continuous max-flow method. At the later stage of the process, only several iterations were needed for convergence of u. For minimization with respect to the global average intensity information c and the local average intensity information, we can just use their definitions to update them after getting new u in each iteration.

We have tested our method on several images with intensity inhomogeneities and showed that the RAAC model gives better results than CV model and LRCV model. This model proved to be not sensitive to the locations of the given foreground and the given background. The difficult part of our method is finding the right functions for controlling the updates of the global weight and the local weight. In this thesis, we just choose two linear functions with function values in non-negative intervals as those two weight functions. We have stated several rules which are used to choose the slopes and the intervals of weight functions, however, there is no mathematical proof supporting those rules, all of them come from the practical experience. The future work could be to find a more effective way of updating the weights instead of renewing these two weights according to two pre-defined functions during the iterations.

Bibliography

[1] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[2] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[3] Rémi Ronfard. Region-based strategies for active contour models. *International journal of computer vision*, 13(2):229–251, 1994.

[4] Tony F Chan and Luminita A Vese. Active contours without edges. *Image processing, IEEE transactions on*, 10(2):266–277, 2001.

[5] Shigang Liu and Yali Peng. A local region-based chan–vese model for image segmentation. *Pattern Recognition*, 45(7):2769–2779, 2012.

[6] LIU Shigang et al XING Hui, PENG Yali. Regional adaptive active contour model for image segmentation. *Computer Engineering and Applications*, 51(9), 2015.

[7] Stanley Osher Ronald Fedkiw and S Osher. Level set methods and dynamic implicit surfaces. *Surfaces*, 44:77, 2002.

[8] RC Gonzalez and RE Woods. Digital image processing: Pearson prentice hall. *Upper Saddle River, NJ*, 2008.

[9] Mari Aurlien Kvile. Continuous max-flow for image segmentation with shape priors. 2014.

[10] Robert Alexander Adams. *Calculus: Single Variable*. Pearson Addison Wesley, 2006.

[11] Ivar Ekeland and Roger Témam. *Convex analysis and Variational problems.* Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, 1999.

[12] Dimitri P Bertsekas. Nonlinear programming. 1999.

[13] Jorge Nocedal and Stephen Wright. *Numerical optimization.* Springer Science & Business Media, 2006.

[14] LR Ford and DR Fulkerson. Flows in networks, a rand corporation research study, 1962.

[15] Jing Yuan, Egil Bae, and Xue-Cheng Tai. A study on continuous max-flow and min-cut approaches. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2217–2224. IEEE, 2010.

[16] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics*, 42(5):577–685, 1989.

[17] Niels Chr Overgaard, Ketut Fundana, and Anders Heyden. Pose invariant shape prior segmentation using continuous cuts and gradient descent on lie groups. In *Scale Space and Variational Methods in Computer Vision*, pages 684–695. Springer, 2009.

[18] Tony F Chan and Jianhong Jackie Shen. *Image processing and analysis: variational, PDE, wavelet, and stochastic methods.* Siam, 2005.

[19] E Giusti. *Minimal surfaces and functions of bounded variation.* Number 80. Springer Science & Business Media, 1984.

[20] Tony F Chan, Selim Esedoglu, and Mila Nikolova. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM journal on applied mathematics*, 66(5):1632–1648, 2006.

[21] Arieh Iserles. *A first course in the numerical analysis of differential equations.* Number 44. Cambridge University Press, 2009.

[22] Kendall E Atkinson. *An introduction to numerical analysis.* John Wiley & Sons, 2008.