Segmentation of the liver from 3D MRI data

Haidi Ibrahim
Supervisors: Dr Kevin Wells & Professor Maria Petrou

Thesis submitted to the University of Surrey for the degree of Doctor of Philosophy

Centre for Vision, Speech and Signal Processing
School of Electronic and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

May 2005

© Haidi Ibrahim 2005
Summary

Three dimensional (3D) visualisation has the potential to significantly ease the decision making in presurgical planning. The first stage of creating a 3D model for this purpose is to segment the liver from magnetic resonance (MRI) images. However, MRI images often contain data corrupted by intensity variations in field strength due to the sensitivity of the radio frequency (rf) coils used in the MRI scanner.

In this thesis, we investigate several approaches to arrive at a solution to overcome this inhomogeneity problem, and at the same time, improve the image quality. These experiments show that the use of local enhancement, followed by median filtering, and toboggan contrast enhancement, is a good solution to achieve this aim.

We then automate a segmentation technique known as intelligent scissors to segment the liver. The user only needs to select an initial slice, and the method is executed automatically. From the initial slice, the contour propagates inside the volume and segments the liver in every slice using a dynamic programming algorithm.

Key words: MRI, segmentation, liver, 3D model, presurgical planning, noise reduction, intelligent scissors.
Acknowledgements

In the name of Allah, the Most gracious, the Most merciful.

I dedicate this thesis to my parents. Their blessings and prayers helped me greatly in my life and studies. Not forgetting my other family members, especially my sisters, Rohaida Ibrahim and Hasniza Ibrahim, my brother-in-law, Edinolfi Kamalul Azmi, and my nephew, Ezri Edinolfi.

I would like to thank my sponsor, namely Universiti Sains Malaysia, for their financial support which allowed me to pursue this study at the University of Surrey.

I am indebted to both my supervisors, Dr. Kevin Wells and Professor Maria Petrou, for their sincere advice and support for my project. Without their guidance, this work would not have been possible.

I would like to thank Dr. Simon Doran who provided the raw magnetic resonance imaging (MRI) datasets, and Dr. Oystein Olsen who provided reference MRI images for validation. I would like to thank Dr. Adrian Hilton and Dr. Amr Ahmed who kindly provided me with some basic knowledge on computer generated imagery.

Thanks to Professor Sameer Singh and Dr. Farzin Mokhtarian for their constructive comments toward this thesis.

Thanks to the current and previous members of CVSSP for assisting me during my stay in the department, especially to Daniel Rodriguez, Eleni Kokkinou, Farahnaz Mohanna, John Chiverton, Manuel Segovia, Maria Faraklioti, Matthew Deighton, Mohammad Hamiruce Marhaban, Mohd Iqbal Saripan, Munir Zaman, Siti Khairunniza Bejo, Sunil Chandra, Robert King and Roberta Piroddi.

I would also like to give a special appreciation to my friends, Azrulazhar Jamaludin, Berdibek Nagmetov, Ch’ng Heok Ngce & Areeya Asimblu, Elbek Yusupov, Farrukh Kamalov, Foo Yoke Wai & Chong Poi Wan, Ithom Djalahov, Lee Wei Kian, Lee Yuen Tat, Loh Wei Kiet, Lydia Sumirah Abdul Khalil, and Tengku Shuhaidi Tengku Ibrahim, for their supportive help and concern.

Thanks to everybody.
## Contents

1 Introduction .............................................................................................................. 1  
1.1 Motivation .............................................................................................................. 1  
1.2 Achievements of this Thesis .................................................................................. 5  
1.3 Structure of the Thesis .......................................................................................... 6  

2 Literature survey on liver segmentation ..................................................................... 9  
2.1 Segmentation Methods .......................................................................................... 10  
2.1.1 Manual segmentation ..................................................................................... 10  
2.1.2 Thresholding .................................................................................................... 10  
2.1.3 Image subtraction ............................................................................................ 12  
2.1.4 Classifiers ........................................................................................................ 12  
2.1.5 Clustering ......................................................................................................... 13  
2.1.6 Seeded region growing .................................................................................... 14  
2.1.7 Fuzzy segmentation ......................................................................................... 15  
2.1.8 Mathematical morphology .............................................................................. 15  
2.1.9 Watershed transformation ................................................................................. 16  
2.1.10 Active contours .............................................................................................. 16  
2.1.11 Deformable 3D model ................................................................................... 17  
2.1.12 Intelligent scissors ......................................................................................... 18  
2.2 Discussion ................................................................................................................ 24  
2.3 Summary .................................................................................................................. 25
3 2D preprocessing of axial abdominal MRI images

3.1 Training Dataset and the Quality Measures

3.1.1 Measure of the efficiency of smoothing, \( \varepsilon \) .......................... 32

3.1.2 Measure of the strength of edges, \( \beta \) ............................ 33

3.1.3 Measure of the bias field homogeneity, \( \alpha \) ................. 34

3.1.4 Criterion used to find the optimal kernel size for smoothing ... 35

3.2 Problem Formulation ................................................................. 36

3.3 Stage 1: Bias Field Correction Techniques ................................. 36

3.3.1 Homomorphic filtering in the frequency domain ................. 38

3.3.2 Homomorphic unsharp masking (HUM) ................. 40

3.3.3 Local enhancement .............................................................. 41

3.3.4 Results of the bias field correction .................................... 41

3.4 Stage 2: Additive Noise Reduction Techniques ......................... 51

3.4.1 Median filtering ................................................................. 52

3.4.2 Gaussian smoothing ......................................................... 53

3.4.3 Results on the additive noise reduction ......................... 54

3.5 Stage 3: Enhancement of the Edges ........................................... 56

3.5.1 Results of edge enhancement ............................................ 64

3.6 Conclusions .................................................................................. 65

4 3D preprocessing of MRI dataset ................................................. 67

4.1 3D Preprocessing Chain ........................................................... 67

4.1.1 3D local enhancement ........................................................ 68

4.1.2 3D median filtering ........................................................... 68

4.1.3 3D toboggan contrast enhancement ................................... 68

4.2 Measurements ............................................................................ 70

4.2.1 Measure of the strength of edges, \( \beta \) ......................... 70

4.2.2 Measure of the bias field inhomogeneity, \( \alpha \) ......... 71

4.2.3 Criterion used to find the optimal kernel size for smoothing ... 72

4.3 Results and Discussion ............................................................. 73

4.3.1 The training dataset ......................................................... 73

4.3.2 3D local enhancement ......................................................... 73
<table>
<thead>
<tr>
<th>Contents</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.3 3D median filtering</td>
<td>75</td>
</tr>
<tr>
<td>4.3.4 3D toboggan contrast enhancement</td>
<td>75</td>
</tr>
<tr>
<td>4.3.5 Discussion</td>
<td>76</td>
</tr>
<tr>
<td>4.4 Conclusions</td>
<td>79</td>
</tr>
</tbody>
</table>

## 5 Salienation of the liver edges

5.1 Introduction to the Primary Visual Cortex (VI) Model .................. 82
5.2 Implementation of the VI Model                                      89
5.3 Results and Discussion                                              89
5.4 Conclusions                                                         91

## 6 Automatic intelligent scissors for MRI liver segmentation

6.1 Methodology                                                         93
6.2 Stage 1: Selection of the Initial Slice                              94
6.3 Stage 2: Segmentation of the Initial Slice                           95
   6.3.1 Thinning of the saliency map                                     95
   6.3.2 Estimation of the centre of the curve                           96
   6.3.3 Identification of the seed points                               97
6.4 Stage 3: Propagation of the Contour                                 98
   6.4.1 Selection of the seed points                                    98
   6.4.2 Point snap                                                       102
   6.4.3 Local computation                                               103
6.5 Stage 4: Combining the Segmentation Results                         104
   6.5.1 Stand-alone segmentation in the axial direction                 104
   6.5.2 Combination of axial-coronal and axial-sagittal results         104
   6.5.3 Axial-coronal-sagittal segmentation                             105
   6.5.4 Axial-sagittal-coronal segmentation                             105
6.6 Results and Discussion                                              106
   6.6.1 The input dataset                                               106
   6.6.2 The ground truth                                                107
   6.6.3 Modification of the cost function                               110
   6.6.4 Stage 1: Selection of the initial slice                         115
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.6.5</td>
<td>Stage 2: Segmentation of the initial slice</td>
<td>115</td>
</tr>
<tr>
<td>6.6.6</td>
<td>Stage 3: Propagation of the contour in the axial direction</td>
<td>118</td>
</tr>
<tr>
<td>6.6.7</td>
<td>Stage 4: Combining the segmentation results</td>
<td>119</td>
</tr>
<tr>
<td>6.6.8</td>
<td>Evaluation of the results</td>
<td>125</td>
</tr>
<tr>
<td>6.7</td>
<td>Conclusions</td>
<td>129</td>
</tr>
<tr>
<td>7</td>
<td>Evaluation of the algorithm</td>
<td>131</td>
</tr>
<tr>
<td>7.1</td>
<td>Input Datasets</td>
<td>133</td>
</tr>
<tr>
<td>7.2</td>
<td>Results and Discussion</td>
<td>143</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Results of the segmentation</td>
<td>143</td>
</tr>
<tr>
<td>7.2.2</td>
<td>The role of preprocessing towards segmentation</td>
<td>162</td>
</tr>
<tr>
<td>7.3</td>
<td>Conclusions</td>
<td>164</td>
</tr>
<tr>
<td>8</td>
<td>Conclusions and future work</td>
<td>165</td>
</tr>
<tr>
<td>8.1</td>
<td>Summary and Conclusions</td>
<td>165</td>
</tr>
<tr>
<td>8.2</td>
<td>Future work</td>
<td>167</td>
</tr>
<tr>
<td>A</td>
<td>Literature review on 3D visualisation techniques</td>
<td>171</td>
</tr>
<tr>
<td>A.1</td>
<td>Maximum Intensity Projection (MIP)</td>
<td>171</td>
</tr>
<tr>
<td>A.2</td>
<td>Local Maximum Intensity Projection (LMIP)</td>
<td>171</td>
</tr>
<tr>
<td>A.3</td>
<td>Volume Rendering</td>
<td>173</td>
</tr>
<tr>
<td>A.4</td>
<td>Surface Rendering</td>
<td>174</td>
</tr>
<tr>
<td>B</td>
<td>3D visualisation using marching cube algorithm</td>
<td>177</td>
</tr>
<tr>
<td>B.1</td>
<td>Methodology</td>
<td>178</td>
</tr>
<tr>
<td>B.2</td>
<td>Results and Discussion</td>
<td>180</td>
</tr>
<tr>
<td>B.2.1</td>
<td>Results by roughly estimating the surfaces</td>
<td>181</td>
</tr>
<tr>
<td>B.2.2</td>
<td>Results by using the actual location of the surfaces</td>
<td>181</td>
</tr>
<tr>
<td>B.2.3</td>
<td>The effects of cube size on the rendered model</td>
<td>182</td>
</tr>
<tr>
<td>B.3</td>
<td>Conclusions</td>
<td>184</td>
</tr>
<tr>
<td>C</td>
<td>Calculation of the 3D gradient kernels</td>
<td>185</td>
</tr>
<tr>
<td>D</td>
<td>The division of the body mask in 3 equal volumes</td>
<td>187</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Segmentation is among the most important branches of digital image processing. This process enables us to emphasise different regions in the image (which can help us to evaluate the information from a scene more easily), or to eliminate irrelevant pixels from the image scene (which can result in a significance reduction in the complexity of an automatic machine vision system) [125, 149, 157]. In this thesis, we present the work regarding the automatic segmentation of the liver surface from volumetric (3D) medical datasets, which are acquired by a magnetic resonance imaging (MRI) scanner. The results of this segmentation are aimed at being used by surgeons or doctors to evaluate the condition of their patient in a more efficient way. The motivation of this project is presented in section 1.1. Our achievements of the thesis are presented in section 1.2. The last section of this chapter, which is section 1.3, introduces each individual chapter of this thesis and describes briefly how the chapters are connected to each other.

1.1 Motivation

The liver is located in the upper right portion of the abdominal cavity as shown in Figure 1.1. This organ is the largest organ in the human body, and plays numerous vital roles in order to make the body functioning properly. This organ converts glucose
to glycogen, produces bile, synthesises urea, destroys old blood cells, and has many other functions.

Figure 1.1: Location of the liver within the body. This figure shows that the liver is in contact with other organs such as lungs, heart, stomach, and intestine. (Illustration taken from [113])
1.1. Motivation

Unfortunately, there are some deadly diseases associated with the liver, for example, cirrhosis and liver cancer. Cirrhosis is the condition of the liver where the scars caused by the infection of hepatitis C virus or alcoholic liver diseases, replace the healthy tissues of the liver, change the liver structure, and impair its performance. It is estimated that about 26,000 to 35,000 patients die because of liver cirrhosis every year [47, 61].

Liver cancer can be divided into two types, i.e. primary liver cancer and secondary liver cancer. Primary liver cancer is the cancer that starts from the liver itself and has a strong relation to hepatitis B virus, hepatitis C virus, alcoholism, and alfatoxin\(^1\) [4, 8, 18]. Primary liver cancer such as hepatocellular carcinoma (HCC) is really one of the potentially life threatening problems in the liver. HCC is the fourth most common cancer in the world [4], and results in about one million deaths per year [28].

Secondary cancer, or also well known as metastatic cancer, is the cancer that begins from other parts of the body. This is because the cancer can spread up by local extension or through the blood and lymphatic system [18]. For example, the secondary liver cancer can originate from the primary cancer of the colon, breast, or pancreas [21, 68, 164]. In the United Kingdom alone, there are about 70,000 patients of secondary liver cancer per year [111].

Nevertheless, there are some treatments available for life threatening cirrhosis or liver cancer, such as surgery, chemotherapy or radiotherapy, depending on the stage, type, location and the number of the tumours [1, 21, 102]. Among these techniques, liver surgery, which is either liver resection or liver transplant, is the most popular and effective treatment, especially to treat cirrhosis or primary liver cancer [24, 36, 37].

Yet, in order to detect the abnormalities of the liver, or to plan the surgery, medical imaging is usually needed. Up to now, there are three common imaging techniques used to access the liver, i.e. ultrasonography, x-ray computed tomography and MRI. However, currently MRI is well known to have better soft tissue contrast, which makes this modality superior compared with other imaging modalities in detecting early, as well as widespread liver diseases [69, 145]. The aim of this project is to successfully

\(^1\)Alfatoxin is a group of carcinogens (cancer-causing agents) that is produced by a fungi that sometimes contaminates certain foods, such as peanuts, grains and seeds.
segment the liver surface, automatically or at least with very minimal user intervention, from 3D MRI data.

Before proceeding further, it is pertinent to describe some of the main issues which make fully automatic liver segmentation from 3D MRI data challenging:

1. The liver is a deformable organ, and move-able inside the abdominal cavity. So, the liver is preferably imaged in a single breath hold in order to eliminate respiratory misregistration. However, as the scan time decreases, the signal to noise ratio (SNR) also decreases. Thus, the quality of imaging of the liver is not as good as imaging of a rigid structure, such as the bone, or the brain, which can be immobilised.

2. There are inter- and intra-patient variations in the shape of the liver, making it difficult to use strict a-priori contour based knowledge.

3. The abdominal cavity also contains other organs, which are often in contact with each other, including the liver surface, as shown in Figure 1.1. In MRI, those organs and the liver’s parenchyma\(^2\) may be represented by very similar intensity values, depending on the setting of acquisition parameters such as repetition time (TR), echo time (TE), and the flip angle. Furthermore, MRI datasets also suffer from radio frequency (rf) field inhomogeneity. Thus, a-priori knowledge based on thresholding the intensity value cannot be used.

There are many possible advantages of segmenting the liver surface, such as to create 3D liver model, to separate the liver region from the surrounding organs (for better visual inspection), and to calculate the changes in liver volume due to the diseases. Although the liver can be segmented manually for these purposes, an automatic segmentation method is very desirable, thus our work is in line for filling this requirement.

The current common routine in liver presurgical planning requires the surgeons to inspect a series of 2D image slices, and mentally reconstruct the 3D anatomical information within the patient. As this approach cannot fully realise the subtle volumetric

---

\(^2\)Parenchyma is defined as the set of cells that constitute the functioning part of the tissue or organ.
1.2 Achievements of this Thesis

detail within these data, computerised 3D visualisation of the liver has been introduced in this presurgical planning stage. It has been suggested by several authors that 3D visualisation can help the surgeons to plan minimal invasive surgery effectively and to decide whether the patient is suitable for surgery or not [38, 45, 118, 135, 138, 149].

Yet, the first stage to produce a 3D visualisation of the liver, is to extract the liver components from the dataset. Several stages of segmentation, as shown in Figure 1.2, are needed to define the anatomy of the patient's liver; i.e. the surface of the liver, the blood vessels, and diseased areas such as lesions or tumours [52, 118, 147, 149, 156]. As these segmentations may be independent from each other [118, 149], in this project, we only consider the segmentation of the surface of the liver.

![Figure 1.2: Basic steps of creating a 3D liver model for preoperative planning. Starting from the original tomographic data, several image segmentation tasks have to be carried out. (Based on [118].) ](image)

1.2 Achievements of this Thesis

The major contribution of this thesis is the development of a novel approach to segment the liver surface from 3D MRI dataset. It is worth noting that this work is actually among the earliest works regarding segmentation of the liver from an MRI dataset. Up to now, most of the research in this field has been based on x-ray computed tomography
datasets, which are much easier to segment as most tissues can be characterised by their Hounsfield unit [156].

In order to make an automatic segmentation possible, we introduce a preprocessing stage to enhance the quality of the input dataset. We demonstrate that a preprocessing chain which consists of 3D local enhancement, followed by 3D median filtering, and 3D toboggan contrast enhancement, is very effective. Combination of these methods can reduce the rf inhomogeneity artifact, and improve the strength of the liver edges.

This work demonstrated that we can automate a hitherto 2D interactive segmentation tool to segment the liver surface from 3D MRI dataset. The human operator only needs to select one axial slice as the initial slice, and our algorithm segments the liver automatically by using the information provided by the previous segmentation result, which is either from the previous neighbouring slice, or from the segmentation in a different orientation. As five out of eight segmentation results from the testing datasets are approved by the radiologist as acceptable results, this indicates that our automatic intelligent scissors is reliable for segmenting the liver surface despite of the large variation among the input datasets.

1.3 Structure of the Thesis

This thesis is divided into eight chapters:

Chapter 2 presents a literature survey on segmentation techniques developed by other researchers to extract the anatomical structures from a 3D dataset. In this review, only the segmentation from an MRI or a CT dataset are considered. Based on this review, we decide to adapt the segmentation technique known as intelligent scissors and make it automatic.

To simplify the problem of automatic segmentation, we introduced a preprocessing step in our work. In Chapter 3, by using one training dataset, we investigate several 2D preprocessing techniques to improve the quality of our MRI input dataset. The main aim of this chapter is to reduce the rf field inhomogeneity artifact. In addition to this, we also try to reduce the level of noise, and improve the strength of the liver edges. Several
1.3. Structure of the Thesis

measurements are introduced in order to choose the optimal parameter setting, and to select the best preprocessing protocol. It is shown that the preprocessing chain which consists of local enhancement, median filtering and toboggan contrast enhancement fulfills our requirement.

In Chapter 4, the 2D preprocessing protocol adopted is extended into 3D. It is shown that 3D preprocessing produces a better quality output compared with its 2D counterpart, and thus it is decided to use 3D preprocessing for the rest of the thesis.

Chapter 5 describes our implementation of the primary visual cortex model (i.e. V1 model) proposed by Li [92] to salicinate the edge of the liver. The salienciation maps produced from this model are used as one of the features in the new cost function we propose for the intelligent scissors algorithm. The saliency map constructed in this chapter is also used in order to find the seed points in the initial slice of the segmentation process described in Chapter 6.

Chapter 6 explains the approach of automating the segmentation technique known as intelligent scissors [109], starting from the selection of the initial slice, propagating the segmentation process to the remaining slices in the dataset, and refining the result. In this chapter, we are still using our training dataset. In this chapter we also show that the combination of the segmentation results from orthogonal directions reduces significantly the segmentation error.

Evaluation of this technique, using eight different testing datasets, is presented in Chapter 7. The preprocessing protocol and the segmentation was run with all their parameter values fixed for all datasets. The output segmentations were shown to a radiologist and judged as acceptable. The datasets for which the algorithm failed were those that violated the basic assumption on which the presented work lies, namely that the slices are dense enough so the shape of the liver does not change significantly from one slice to the next.

Chapter 8 consists of general conclusions from this project. Some suggestions for future work are also presented.
Chapter 2

Literature survey on liver segmentation

The human abdominal cavity contains several organs in close proximity. As a result, when imaging the liver using magnetic resonance imaging (MRI) or x-ray computed tomography (CT), other organs are also included in these 2D axial slices. Thus, a process known as segmentation is required to define the liver.

Segmentation is very useful for 3D visualisation of objects from complex 3D datasets. For example, visualisation using surface rendering can use the results from this segmentation to locate the surface of the object. In volume rendering, output from the segmentation process can be used to assign the opacity value to the voxels based on the particular regions.

Section 2.1 presents a review on segmentation methods commonly used to extract anatomical information, such as the brain and liver, from 3D MRI and CT data. A brief discussion on this literature survey can be found in section 2.2. However, because medical data are normally corrupted by artifacts, this section also includes an introductory discussion on appropriate preprocessing for MRI data. Then, a summary of this chapter is given in section 2.3.

A 3D dataset is usually represented by a stack of 2D contiguous axial slices.

We present a brief review on some common methods for displaying medical images in 3D, in Appendix A. Some simple experiments with marching cube algorithm, for displaying the liver as a surface, can be found in Appendix B.
2.1 Segmentation Methods

In this section, a number of segmentation approaches commonly applied to x-ray computed tomography (CT) or magnetic resonance imaging (MRI) datasets are described, starting below with a simple method and proceeding to more advanced methods.

2.1.1 Manual segmentation

The most obvious approach to segmentation involves manual tracing of the MRI or CT image of patient’s anatomical information. The method requires an experienced radiologist to trace manually the boundary of important regions on the 2D input slices by using a pointing device.

This method has been used to create a 3D model for planning of liver cryosurgery in Brigham and Women’s Hospital, Massachusetts, USA [66]. Robb et al. [138] also use a manual tracing technique to create 3D models in some of their presurgery planning cases in Mayo Clinic. As the user has overall control of the output, Bae et al. [5] and Matsushita et al. [105], for example, use the manually segmented liver as their ground truth (gold standard) to validate their segmentation technique. Lamecker et al. [83, 84] on the other hand, segment the liver manually from 30 CT training datasets, to create a statistical shape for their model based segmentation method.

Up to now, manual tracing is the most useful technique to segment the patient’s anatomical structure for abdominal imaging applications [138]. However, this technique is slow, tedious, and prone to errors, being highly dependent on the skill of the radiologist [5, 65, 146].

2.1.2 Thresholding

Thresholding is based on the assumption that every object in the image scene can be represented by a unique range of intensity values. Thus, a specific object can be extracted from the image by using the following equation:

\[
g(x, y, z) = \begin{cases} 
1 & : T_L \leq f(x, y, z) \leq T_U \\
0 & : \text{otherwise}
\end{cases}
\]  \hspace{1cm} (2.1)
where \( g(x, y, z) \) is the output image, \( f(x, y, z) \) is the input image, and \( T_U \) and \( T_L \) are the upper and lower threshold values respectively [55, 125, 141, 150, 157, 162].

These threshold values can be set manually based on the visual inspection of the segmentation result [59, 81, 98], or automatically by using \textit{a-priori} knowledge [5, 114, 161]. Another possibility is by calculating the threshold value using Otsu’s method [95, 168], which estimates the valley between two overlapped intensity distribution [121].

Bae et al. [5] employ a thresholding technique to segment the liver automatically from a 3D CT dataset. They use one of the middle axial slices, where the liver fills almost the entire right half of the image, as the reference image. In this slice, a region of interest (ROI) is placed automatically into the area of the liver using \textit{a-priori} knowledge of the liver location. Based on the intensity distribution in this ROI, the algorithm automatically estimates the intensity range of the liver, and thresholds the 3D dataset. Because other pixels may also be misclassified as the liver parenchyma (see Figure 2.1(c)), they use \textit{a-priori} knowledge of the compactness (i.e. compactness = area/perimeter\(^2\)) to select the actual liver area in every slice. This area is further validated by comparing it with the location of the detected area in the previous image.

![Figure 2.1](image-url): An example of segmentation using a thresholding method. Image (a) shows an axial abdominal CT slice, while image (b) is its corresponding histogram. The histogram shows several peaks, which correspond to the objects in the image. We assume that the liver values are in the range 165 to 185. Image (c) shows the thresholded version of image (a) using this range of intensity. As shown in (c), further processing is needed to refine the result. (Image (a), which is a greyscale image, is taken from http://www.pruenergang.de/projekte/waveletbeispiele.html.)

A thresholding technique, however, is only suitable for uniform, high contrast objects. The greyscale data characteristic of abdominal MRI, in contrast to CT, include many...
objects containing similar distributions of greyscale intensity. Thus, this potentially makes a thresholding approach unsuitable for MRI hepatic segmentation.

### 2.1.3 Image subtraction

If contrast agent is administered into the patient’s body during the acquisition process, a dynamic dataset which consists of images with and without contrast can be acquired. By first registering these images (i.e., the image containing contrast agent with the image containing no contrast agent), the areas which are enhanced by the contrast agent can be easily segmented by taking the different of these two images. Currently, this method is widely used to segment blood vessels [63, 77, 116, 170] and tumours [20, 32, 71, 103, 175, 177]. However, it is of limited use for hepatic applications as contrast agent is not always used, and in any case would tend to enhance the hepatic vasculature rather than the parenchyma.

### 2.1.4 Classifiers

![Figure 2.2: An example of 2D feature space of a multispectral dataset. In this example, the pixels correspond to this plot, can be divided into three distinguish groups. The segmentation result can be obtained by projecting back this labelled feature into the spatial domain.](image)

The classifier approach is a method used in pattern recognition field to classify regions in the image scene by assuming that regions of interest can be identified by unique, quantifiable features, as shown in the Figure 2.2. These features, such as the intensity value, outputs from different filtering processes, texture measures, gradient magnitude
and gradient direction, can be derived from a single spectral dataset, or from a multi-spectral dataset. The number of features needed is highly dependent on the complexity of the image scene [25, 70, 128, 173].

The human operator, first, segments the regions of interests from the training datasets, manually, and searches the features that can isolate these regions from each other. This step is compulsory in order to automatically segment the new input dataset. This is because the features measured from the input dataset will be classified according to the a-priori knowledge derived from the training dataset [128]. Various assumptions about the statistical nature of the objects are required, along with a suitable classification scheme. The latter may include a simple Bayes Minimum Error decision boundary [128] through the data or may include combining features (e.g. by using principle components analysis (PCA)), before a decision boundary is applied.

It has been reported that this technique successfully classifies brain tissues [26, 27, 31], and segments the cartilage of the knee [172], all from 3D MRI datasets.

2.1.5 Clustering

Clustering methods have been used to classify brain tissues [35, 89, 126, 127], and to distinguish between cancer tissue with normal tissue [82, 183]. Clustering follows the same concept of the classifiers (see section 2.1.4) except that clustering methods do not need initial training data. This method divides the feature space iteratively, where at each iterations, the members and the properties of each clusters are updated [128].

One of common clustering methods is the K-means algorithm, which divides the feature space into K different classes. The algorithm begins by placing K group centroids located far away from each other in the feature space. These group centroids produces K regions based on the nearest neighbour procedure. Then, the location of the group centroids are updated, and the algorithm re-segments the feature space based on the new centroids. The process is repeated, and the clustering of the feature space is only accepted when the location of the group centroids are unchanged [106, 119].

Another clustering method is based on the expectation-maximisation (EM) algorithm. This method assumes that the distribution of the pixel intensities in the image is a
combination of Gaussian distributions with different coefficients. Thus, the EM method tries to segment the image by finding the coefficients of these Gaussian distributions [128].

2.1.6 Seeded region growing

Seeded region growing requires the user to select a point, known as the seed, in the region of interest and set a threshold interval. Neighbouring pixels around the seed are examined. Then pixels are appended the region if they are within the threshold interval. If so, then each added pixel becomes a new seed [2, 49, 55, 166].

There are many works that use seeded region growing to extract anatomical information from medical data. Works by Petrick et al. [124], Guliato et al. [57], and Lee, Park and Park [88], for example, use this technique to segment the breast tumours from mammography. Work by Justice and Stokely [72] segments the brain from MRI datasets. Others use seeded region growing to segment 3D blood vessels [9, 16, 33, 148, 178, 182], bone [136, 137], liver [62, 132, 133], and lung [67].

Pohle and Toennies [132, 133] implement a seeded region growing technique with updated threshold values, in order to reduce the influence of the partial volume effect to the segmented result. Their thresholds are based on the average intensity value of the current segmented region, and two standard deviation values, which are computed respectively from the voxels that have the intensity greater, or lower than the mean of the population.

To be successful, seeded region growing requires strong edge or discontinuities in order to successfully terminate the growing region within the object of interest [2, 25]. However, in MRI liver data, there are large parts with poor or indiscernible edges, resulting in a region “leaking” into adjacent objects when such seeded region growing procedures are employed.
2.1.7 Fuzzy segmentation

Matsushita et al. [105] use a fuzzy segmentation frame to extract the liver component from a 3D MRI dataset. They use 36 fuzzy interference rules, which are influenced by the intensity of the voxel, the gradient magnitude value, the distance between the edgels, and also the segmented result from the previous slice. They claim that this approach can produce a liver volume with a segmentation error of about 11%.

Kobashi et al. [78, 79], try to imitate the radiologist’s decision making process in segmenting the liver from dynamic MRI data, by implementing fuzzy rules in their region growing approach. Their fuzzy rules are based on the characteristics of the liver and corresponding major blood vessels in response to the injection of a contrast agent over time.

2.1.8 Mathematical morphology

The researchers in the Toyohashi University of Technology, Japan [48, 56, 73], use morphological operations to segment all abdominal organs from a 3D CT scan, in a single pass. By manipulating the 3D greyscale morphological operations, with structuring element of different sizes, they first detect the region that contains the abdominal organs. Then, by using a recursive 3D binary erosion of the detected region, various seed points are created. The corresponding organs for seed points (i.e. the liver, left kidney, spleen, and stomach) can be recognised by using a-priori knowledge of the location. As these organs are connected to each other, the authors iteratively dilate these seed points, using 3D binary dilation, and estimate the border of the organs based on the location where these dilated regions meet.

Recently, this group uses multi-phase CT datasets in their application [142]. First, they segment the liver blood vessels by using a thresholding technique, as the blood vessels in the dataset are contrast enhanced. Then, by using a spherical structuring element of radius nine voxels, they dilate these blood vessels to estimate the region of the liver.

Other research groups, such as Selle et al. [148] and Soler et al. [156], in their work, use morphological operations to improve the result of the liver blood vessel segmentation.
These operations make their analysis of the blood vessels possible.

### 2.1.9 Watershed transformation

The basic idea of the watershed transformation is to consider the greyscale image as a topological relief, which will be flooded by water. The first step of this transformation is to mark the seeds, which are the points where the pixel flooding will begin. As the water becomes deeper, the regions around the seeds become flooded, and the regions expand. The waters are stopped when they are about to flood any regions from other seeds, and the edges are identified as the locations where the waters from different seeds meet (i.e. the location of the watershed) [12, 99, 151].

Lapeer, Tan and Aldridge [85], roughly estimate the contour of the liver by using interactive watershed segmentation. This contour then is refined by using an active contour. Selle et al. [149] use a 3D watershed transformation to extract tumours from the liver images. In their implementation, the human operator is required to mark two points, interactively, one inside and one outside the lesion.

### 2.1.10 Active contours

Examples of implementation of active contours in liver segmentation from a CT dataset can be found in [6],[39],[85],[101],[134] and [181]. Active contours, also known as snakes, require an initial approximate contour which is drawn manually by a radiologist, or automatically, based on prior knowledge. Image forces will push the contour to the edge of the object in the image.

In order to achieve this, an energy function is associated with the snake, which has to be minimised. The snake is represented by a curve, \( v(s) = (x(s), y(s)) \), where \( x(s) \) and \( y(s) \) are the \( x \) and the \( y \) co-ordinates of points along the contour, parameterised by \( s \). The snake's energy is given by:

\[
E_{\text{snakes}} = \int_0^1 E_{\text{int}}(v(s)) + E_{\text{image}}(v(s)) + E_{\text{con}}(v(s)) \, ds
\]

(2.2)

where \( E_{\text{int}} \) is the internal energy due to bending or discontinuities, \( E_{\text{image}} \) is the image
force, and $E_{con}$ denotes the external constraint forces. For each iteration, the snake will deform towards its minimum energy [76, 107].

However, snakes have some disadvantages. Snakes are sensitive to numerous parameters, and can produce undesirable deformation effects, such as shrinking and vertex clustering. In addition, snakes need multiple iterations, and thus require a relatively large computational time [7, 74, 146].

2.1.11 Deformable 3D model

The extension of snakes to 3D are known as deformable surfaces, otherwise referred to as balloons or shrink-wrap surfaces. Although deformable surfaces are often considered less robust and less practical compared to snakes, this technique is more flexible and powerful in estimating missing parts of volumetric boundaries [51, 146].

Soler et al. [156] successfully segment the liver from a CT dataset using a 3D deformable model. First, they threshold the dataset based on the Hounsfield units, in order to estimate the location and the shape of the liver, and also to take out unrelated organs, such as the kidneys, bone, and spleen, from the dataset. Then, when they locate the 3D liver template into the dataset, they deform the model in order to minimise the internal and external forces of the surface.

Lamecker et al. [83, 84] use a statistical 3D liver model in their application. In every iteration, the coefficient of this model changes. The deformation stops when the model matches the shape of the liver in the input CT data.

Another way to deform the 3D model is by using probability theory. Work by Boes, Weymouth and Meyer [13], for example, use a Bayesian formulation in order to register a 3D liver model with a CT dataset. They suggest that their approach works better when the 3D kidney model is introduced and deformed simultaneously with the liver in the dataset.

Similar to the active contour, segmentation methods based on a 3D deformable model are also sensitive to their numerous parameters, and computationally intensive [7].
2.1.12 Intelligent scissors

The idea of a user-steered semi-automatic segmentation, which is known as intelligent scissors, was originally proposed by two independent research groups, namely Mortensen and Barrett [109, 110], and Falcao et al.\(^3\) [42, 41]. However, this review only relates to the Mortensen and Barrett's framework, as their work is easier to understand, less heuristic, and has been implemented in several medical applications, including the segmentation of the liver surface for presurgical planning [146, 147].

![Figure 2.3](image)

**Figure 2.3:** An example of segmenting the liver using intelligent scissors. Figure (a) shows the input image and (b) shows an initial user-defined starting point on part of the desired edge contour. While the user move the cursor on the image, the potential edge segment from the current position of the cursor to the starting point, is automatically calculated and displayed, as shown in (c) and (d). Then, when the user satisfied with the suggested edge contour, the user defines the termination point. The edge segment freeze and the termination point now become a new starting point, as shown in (e). The same procedure is applied until a closed loop, as in (f), is defined.

Intelligent scissors can be considered as a combination of edge-based and dynamic programming segmentation. As shown in Figure 2.3, this technique requires the user to determine the starting and the termination points for the edge segments, interactively. The resultant edge segment, which connects two defined points, is created based on the

\(^3\)Falcao et al. name their method as live wire, or live lane.
lowest cost path according to Dijkstra's dynamic programming algorithm\(^4\)[11].

\[ l(p, q) = \omega_z f_z(q) + \omega_s f_s(q) + \omega_D f_D(p, q) \]  

(2.3)

where \( f_z \) is the feature of Laplacian zero crossing, \( f_s \) is the feature of gradient magnitude, \( f_D \) is the feature of gradient direction, while \( \omega \) represents the weight of the corresponding cost function. As intelligent scissors uses dynamic programming to find

\(^4\)Dynamic programming is broadly applied in data networking field. This process which is a graph theoretic algorithm, is used to determine the minimum cost path joining two nodes, i.e. the start point and the end point, in the graph.
the edges, $f_Z$ and $f_G$ are designed to have lower values at the edges by taking the reciprocal of these features.

The zero crossing feature, $f_Z$, is calculated by convolving the input image with the Laplacian convolution kernel. The definition of the Laplacian is given by:

$$\nabla^2(x, y) = \frac{\partial^2 g(x, y)}{\partial x^2} + \frac{\partial^2 g(x, y)}{\partial y^2} \tag{2.4}$$

where $g(x, y)$ is the input image for the convolution. The location where the zero crossings were detected is assigned 0 while the other locations were assigned 1. This is because the edge features, such as zero crossings need to be assigned to minimal cost.

If $\delta_x$ and $\delta_y$ are the first derivative components of the input image $g(x, y)$ in $x$ and $y$ direction, respectively, the gradient magnitude feature, $f_G$, is defined as:

$$f_G = 1.0 - \frac{\delta_M - \min(\delta_M)}{\max(\delta_M) - \min(\delta_M)} \tag{2.5}$$

where $\delta_M$ is the gradient magnitude, i.e. $\delta_M = \delta_x^2 + \delta_y^2$. Thus, $f_G$ is inversely proportional to the actual gradient magnitude value.

By taking $\vec{E}(p) = (\delta_y, -\delta_x)$ as a vector of the edge direction at a point $p$ (i.e. gradient direction rotated $90^\circ$ clockwise), the gradient direction feature, $f_D$, at point $p$ with respect to the neighbouring point $q$, can be calculated by using these following formulas:

$$f_D(p, q) = \frac{2}{3\pi} (\arccos(d_p(p, q)) + \arccos(d_q(p, q))) \tag{2.6}$$

where

$$d_p(p, q) = \vec{E}(p) \cdot \vec{L}(p, q) \tag{2.7}$$
$$d_q(p, q) = \vec{E}q \cdot \vec{L}(p, q) \tag{2.8}$$

and

$$\vec{L}(p, q) = \frac{1}{||p - q||} \begin{cases} q - p &: \vec{E}(p) \cdot (q - p) \geq 0 \\ p - q &: \text{otherwise} \end{cases} \tag{2.9}$$

Mortensen and Barrett [109] use the concept of multiple size kernels to calculate $f_Z$, $f_G$, and $f_D$. In order to calculate $f_Z$, they first convolve the input image with Gaussian
2.1. Segmentation Methods

kernels of size $3 \times 3$ and $7 \times 7$, and produce two different versions of smoothed images. For each smoothed image version, they then convolve it with the Laplacian convolution kernel of size $3 \times 3$. These results estimate the second derivative of Gaussian. Next, the locations of zero crossing on each result are detected and assigned zero. This produces two versions of zero crossing features, which is $f_{Z_3}$ (that derived from smoothed image by Gaussian with kernel size $3 \times 3$), and $f_{Z_7}$ (that derived from smoothed image by Gaussian with kernel size $7 \times 7$). Then, these two versions are combined together as a single zero crossing feature, $f_z$, using the following equation:

$$f_z = 0.45f_{Z_3} + 0.55f_{Z_7}$$

In order to calculate $f_G$ and $f_D$, Mortensen and Barrett [109] first convolve the input image with Gaussian kernels of size $3 \times 3$ to $13 \times 13$, and produce six different versions of smoothed images. Then, they convolve each version of the smoothed image with the kernel shown in Figure 5.3(a) to estimate the first derivative of Gaussian in the $x$ direction. For each pixel location, the largest value of the gradient is selected to represent $\delta_x$ at that point. Then, each version of the smoothed image is convolved with the kernel shown in Figure 5.3(b) to estimate the first derivative of Gaussian in the $y$ direction. Similarly, for each pixel location, the largest value of gradient is selected to represent $\delta_y$ at that position. Then, using these $\delta_x$ and $\delta_y$ values, $f_D$ and $f_G$ are calculated using equation (2.5) to equation (2.9).

The benefit of using the multiple size kernels technique, as shown in Figure 2.5, is that it combines the advantage of small kernels, which are suitable in finding step edges, with the advantage of the larger kernels, which can perform well in noisy images [46, 109]. However, this approach increases processing time.

In intelligent scissors, dynamic programming is initiated immediately after the operator defines the starting point of the desired edge contour (by clicking the mouse). The search algorithm uses dynamic programming, following Dijkstra's method. The output from this processing is a 2D array, showing the linking of edges to the starting point (see Figure 2.6).
Figure 2.5: An example demonstrating the advantage of using the concept of multiple size kernel in order to improve edge detection. (a) Input image: a circle half of which is noiseless and half is immersed in a noisy background. (b) Gradient magnitude produced by 5 x 5 kernel cannot easily detect the edge of the circle in the noisy area. (c) Gradient magnitude produced by 15 x 15 kernel can detect the edge of the circle in the noisy area, but the localisation of the edge reduces (indicated by thick edges). (d) The combination of (b) and (c) by selecting the biggest \( S \times S \) at every pixel position improves the detection of the circle in noisy area and the edge localisation is better than that in (c).

Figure 2.6: An example of dynamic programming using Dijkstra’s algorithm. In this example, only the gradient magnitude feature, \( f_C \), is taken into consideration (i.e. \( \omega_z = 0.0, \omega_C = 1.0, \) and \( \omega_D = 0.0 \)). The starting point is indicated by the circle. (a) Initial local cost matrix. (b) Final cumulative cost and path matrix showing directional links between pixels. (c) Output of dynamic programming; pointers or links from every location back to the starting point. Note, however, that \( f_C \) must be divided by \( \sqrt{2} \) if \( q \) is a horizontal or vertical neighbour of \( p \) in order to maintain a normalised Euclidean distance. (Based on [109, 110].)

The user must move the cursor or pointer along the approximate vicinity of the desired edge contour. The algorithm, using the edge link pointer as shown in Figure 2.6(c), displays the edge which corresponds to the user’s current pointer position. When the user is satisfied with the displayed edge, then a termination point is selected (with a further mouse click). The selected edge is then saved into the output array, and the termination point becomes the new starting point. A further cycle of dynamic programming is automatically executed from this new location. The process is repeated until a closed loop is formed. In general, intelligent scissors can be represented by the
2.1. Segmentation Methods

following steps:

Step 1: Convolve the input image with Gaussian kernels of size $3 \times 3$ to $7 \times 7$.

Step 2: Calculate $f_s$ and $f_o$.

Step 3: Reset the output array.

Step 4: The user defines the starting point of the desired edge contour by using the mouse.

Step 5: Calculate the lowest cost path from every pixel in the image relative to the starting point
using dynamic programming based on Dijkstra's algorithm. The cost function for this dynamic
programming is defined by equation (2.3). Save the link's pointer as a new array.

Step 6: While the user moves the cursor, by using the previously saved link's pointer from Step 4,
display the appropriate edge (i.e. the lowest cost path from the current cursor position relative
to the starting point).

Step 7: The user defines a termination point, again by clicking the mouse, when satisfied with the
displayed edge. Save the current selected edge into the output array.

Step 8: If the edges in the output array defined a closed loop, segmentation is complete, and the
algorithm end. Otherwise, the termination point from Step 7 becomes a new starting point.
Go to Step 5.

In order to reduce human intervention, Schenk et al. [146] use a combination of 2D in-
telligent scissors and a shape-based interpolation technique to extract the liver surface
from 3D datasets. In this approach, the user only needs to use the intelligent scissors
tool in certain slices, and the remaining slices will be automatically segmented by the
shape-based interpolation. The shape-based interpolation technique they use first gen-
erates the binary edges of the contours that have been segmented using an interactive
intelligent scissors. Next, 2D distance maps are created to represent the distance relative
to the detected boundary. The distance of the pixels inside the boundary are signed
as negative, while the distance of the pixels outside the boundary are signed as positive.
Then, they interpolate these distance maps for the remaining slices in the dataset. The
edges in between two slices, which have been segmented using the intelligent scissors,
are then detected as the location of zero crossings.

Stalling and Hege [159], in their implementation of intelligent scissors for medical image
segmentation, use $f_G$ as their only cost feature. They claim that, for medical images,
$f_D$ does not play any significant role in finding suitable edge, and $f_S$, in their opinion,
is very sensitive to noise.
2.2 Discussion

We start this section with a description about the input datasets used in this work. In this project, we are provided with nine complete 3D abdominal MRI datasets, where one randomly selected dataset will be used as the training dataset, while the other eight datasets will be used as the testing datasets. These datasets are single spectral datasets, without contrast agent, and each of them are not compulsory with same acquisition parameter settings.

Based on this fact, it is clear that image substraction, classifier, clustering, and fuzzy segmentation methods are not suitable for our work. Image substraction method requires the input datasets to be contrast enhanced [32], but the datasets used are not. Similarly, classifier and fuzzy segmentation are too dependent on the training dataset [128]. As our input datasets appear visually different from each other, these approaches are unlikely to yield acceptable results. It has also been suggested that classifier and clustering techniques are more suitable for multispectral datasets [126, 127].

Having reviewed the literature in this area, it would appear that up to now, only nine techniques have successfully extracted the liver surface from 3D datasets; manual tracing, thresholding, seeded region growing, fuzzy segmentation, mathematical morphology, watershed transformation, active contour, deformable 3D model, and intelligent scissors. All of these are applicable to x-ray computed tomography (CT) datasets. However, segmentation of CT data is relatively easier compared to the segmentation of MRI data because most of the tissues in CT can be identified based on a repeatable set of Hounsfield units [156]. Of the nine approaches listed above, only manual tracing, fuzzy segmentation, and intelligent scissors are reported to have been used for segmenting the liver from MRI data.

Intelligent scissors has been used in segmenting liver surface from MRI datasets, and this work has been implemented in real life presurgical planning [146, 147]. Thus, in this work, we decided to implement intelligent scissors. However, as this technique requires human interaction, this technique is still too slow and reliant on expert guidance for large 3D datasets. Thus, the contribution of this thesis in this area will be the development of a methodology to segment the liver with minimal user interaction.
2.3 Summary

From the literature, we found that some researchers first preprocessed their data before taking these to the segmentation stage. The main concern for this, is that of MRI datasets often suffer from bias field inhomogeneity, where the recorded signal intensity of an homogenous region changes slowly over the image, produces a shading effect [128, 152]. There are many factors that contribute to this inhomogeneity, including the variation of the rf coils strength in transmitting and receiving signal, where the intensity of the tissues reduces as the distance from the rf coils increases [60, 152]. Bias field inhomogeneity can therefore degrade segmentation performance significantly, particularly where there is little expert intervention, especially with those the segmentation techniques which depend strongly on intensity value [25].

Other than bias field inhomogeneity, the majority of preprocessing techniques in MRI and CT address noise suppression. Techniques such as low pass filtering, or Gaussian filtering, have been reported for reducing the noise level in MRI data [25, 108, 179].

Following in this line, the preprocessing developed for this application will be mainly to address the bias field inhomogeneity problem, and also to reduce the significance of additive noise.

2.3 Summary

Based on the literature, it was decided to further investigate and develop an approach based on intelligent scissors as this technique has been used in real clinical presurgery planning. A method of automating this algorithm is described in Chapter 6.

However, as there are many artifacts present in MRI data, a decision was made to preprocess the input data before the segmentation process. The aim in this preprocessing is to reduce the bias field inhomogeneity, reduce the level of noise, and improving the strength of the liver edges. This should assist in any machine-based decision-making methodology in terms of determining the actual liver boundary from background clutter, as the quality of the liver boundary is highly variable, as described previously in Chapter 1. Further descriptions and experimental results on the training dataset can be found in Chapter 3, and Chapter 4.
2D preprocessing of axial abdominal MRI images

Data acquired by magnetic resonance imaging (MRI) is often corrupted by intensity variations in field strength due to the sensitivity of the radio frequency (rf) coils used in the MRI scanner [60, 152]. In addition to the field bias problem, soft tissue exhibits strong intra-organ variation, making organ segmentation based on edge detection complicated [147]. The problem is aggravated, especially for abdominal MRI, when the target organ touches adjacent organs, which makes the edges barely discernible. Also, if the MRI slice is thick, the segmentation becomes more cumbersome as the partial volume effects make the edges fuzzy [14, 34, 167]. Thus, we decide to preprocess the data before considering any segmentation strategy.

By using a training dataset, we investigate several 2D approaches to arrive at a quick simple solution to these problems. Based on the facts stated above, a preprocessing chain, which consists of three stages, is proposed. Each stage of this preprocessing chain has its own objective. The first stage deals with the reduction of rf bias field inhomogeneity. The second stage lowers the level of the additive noise in the data, while the last stage of preprocessing improves the contrast of the liver delineation.

This chapter is divided into six sections. Section 3.1 presents the training dataset and the quality measure used in this chapter to evaluate the correction results in each
Chapter 3. 2D preprocessing of axial abdominal MRI images

3.1 Training Dataset and the Quality Measures

In this experiment, a complete three dimensional MRI dataset of the abdominal region is used as the training dataset. This dataset was acquired at the CRC Clinical Magnetic Resonance Centre, Institute of Cancer Research, London. It is a gradient echo dataset, with parameter TR = 74.7 ms, TE = 4.0ms, and flip angle = 80°. The dimensions of each voxel are 1 mm × 1 mm × 8 mm.

This dataset consists of eighteen 256 × 256 axial images. Each slice of this dataset is shown in Figure 3.1. As we can see clearly, this dataset suffers from field intensity inhomogeneity, manifest in the bright spots seen at the front and back areas of the patient, and to a lesser extent at the middle of this anatomical region.

Figure 3.1: Original slices of a 3D dataset used in this experiment. Slice on the top left is the top most slice in this dataset.
We divide the preprocessing into three consequence stages, which are:

1. Bias field inhomogeneity correction
2. Reduction of additive noise
3. Enhancement of the edges

All techniques that will be used in each stage will be described in section 3.3, section 3.4, and section 3.5, respectively, together with their experiment results.

To avoid the influence of the background noise, such as the ghosting artifact due to the heart beat, in this preprocessing stage, for each slice in this dataset, we generate the binary mask, (i.e. the body mask), which defines the area of the patient’s body, by using a combination of region growing and morphological operations. An example of these masks is shown in Figure 3.2. Only input pixels corresponding to the “white” region of the body mask are considered in the processing.
Chapter 3. 2D preprocessing of axial abdominal MRI images

Figure 3.2: An example of the body mask, which is actually the binary mask that define the area of the patient body in the image. In this example, the body mask shown in (b) corresponds to the input slice shown in (a).

Before proceeding to define the measures of quality we shall use, it is necessary to define two more binary masks, which are the liver mask and the edge mask. The liver mask is a mask that represents the area of the liver, which we assume would tend to become uniform in the corrected output image. We create this mask manually with the help of the thresholded gradient magnitude image (see Figure 3.3).

Figure 3.3: An example showing the creation of the liver mask. From the input image (a), we calculate the gradient magnitude of the image by using Sobel operators, as described in section 3.5. Then, by a trial-and-error process, we threshold the gradient magnitude image, so only the pixels with low gradient value are retained, as shown in (b). Next, by using the paint brush tool (which is commonly available in any photo editor package), we refine this binary mask manually to keep only the liver areas. This final image, shown in (c), represents the liver mask.

The edge mask, on the other hand, is the binary mask that defines the areas that we think contain the edges around the liver. The edge mask is created from the liver mask, as shown in Figure 3.4.
3.1. Training Dataset and the Quality Measures

Figure 3.4: An example shows how the edge mask been created from the corresponding liver mask (a). First of all, we dilate image (a) four times with $3 \times 3$ structuring element to produce image (b). Then, we make an approximation of the liver mask (which includes the areas of the blood vessels) by erode image (b) four times with $3 \times 3$ structuring element, and we get image (c). The edge mask (shown in (d)) is equivalent to image (b) minus image (c).

We will use three measurements, which are designed specially for our problems, to calculate the improvement done by each method in this preprocessing stage. We introduce the measurement of the bias field inhomogeneity ($\alpha$), the measurement of the efficiency of smoothing ($\epsilon$), and the measurement of the strength of the edges ($\beta$).

However, because we have to explore many different choices of parameter values for each technique we use, instead of evaluating the results on a slice-by-slice basis, we estimate the improvements based on a dataset-by-dataset basis. For each output dataset produced by each set of parameters, we evaluate the result based on the average value (i.e. $\bar{\alpha}$, $\bar{\epsilon}$, $\bar{\beta}$). Then, the improvement on each stage of the preprocessing chain is indicated as below:

1. Bias field inhomogeneity correction: high value of $\bar{\alpha}$

2. Reduction of additive noise: high value of $\bar{\epsilon}$

3. Enhancement of the edges: high value of $\bar{\beta}$

Although an individual slice may have different improvement pattern from other slices in the dataset, evaluation based on dataset-by-dataset is more favorable as this represents the improvement of the majority of slices. Besides, this is also more practical because in our case, we are searching for the global optimal parameter settings which
can be applied to any testing dataset. On the other hand, evaluation based on slice-by-slice basis finds the optimal parameters for each individual slice in the training dataset. However, these parameters are only optimal for the training data, and not necessarily optimal for the testing datasets, which may be different in the number of slices, and the anatomical details they contain.

Because $\alpha$, $\beta$, and $\epsilon$ are somehow related to each other, in this section, we will first introduce the measure of $\epsilon$ (section 3.1.1). Then, this will be followed by the measure of $\beta$ (section 3.1.2), and the measure of $\alpha$ (section 3.1.3). Section 3.1.4, which is related to the second stage of the preprocessing chain, describes the technique we use to choose the optimal smoothing kernel size for each slice in order to lower the level of additive noise in the dataset.

3.1.1 Measure of the efficiency of smoothing, $\epsilon$

The measure of $\epsilon$ relates to the second stage of our preprocessing chain, which is the reduction of the additive noise in the image. Researchers in additive noise reduction normally evaluate the smoothing performance based on the root mean square error (RMSE) [22, 23, 154], mean absolute error (MAE) [19, 22, 154], signal to noise ratio (SNR) [22, 131, 176], and peak signal to noise ratio (PSNR) [19, 22, 155]. These evaluation methods work by comparing the output of smoothing with the ground truth (that is represented by a noise-free image).

In our case, we measure the efficiency of smoothing by using the fact that smoothing reduces the intra-organ contrast in the image. One possible technique to measure the intra-organ contrast is by inspecting the gradient magnitude value. Regions with low intra-organ contrast should have low gradient magnitude because the majority of the pixels in these regions have a very similar intensity values with their neighbouring pixels [55, 125].

First, we calculate the gradient magnitude of the image by using Sobel operators (see section 3.5). Next, we define $\delta_L$ as the average gradient magnitude inside the region defined by the liver mask (an example of the liver mask is shown in Figure 3.3). Then,

$$\epsilon = \delta_L - \delta_L$$

(3.1)
where $\delta_{E_0}$ is the average gradient magnitude of the image before smoothing. It is expected as a result of smoothing, that this measure should give a value greater than zero.

### 3.1.2 Measure of the strength of edges, $\beta$

Singh and Bovis [153] employ several contrast measures to evaluate some contrast enhancement techniques used in their work regarding mammography images. Their measures involve the measurement of the mean intensity of the object of interest, and the mean intensity of the background. These measures however consider these regions represented by only two intensity distributions. But, in the case of MRI liver, this condition is not always true. The liver is normally surrounded by different types of tissue, and for example, the liver on the top slice of our training dataset, as shown in Figure 3.1, touches the lung and the heart. This makes the background region having a variety of intensity values that are both higher and lower than the intensity value inside the liver region. Thus, measuring contrast by mean intensity is in fact not acceptable in this case.

The measure $\beta$ we use is highly relates to the third stage of our preprocessing chain, which is the enhancement of the edges. Thus, it is more applicable to measure the improvement of the liver edges by inspecting its gradient magnitude value. We first calculate the gradient magnitude of the image by using Sobel operators (see section 3.5). Then we define $\delta_E$ as the mean gradient magnitude value inside the edge mask (an example of the edge mask is shown in Figure 3.4). As we want to see how much the edge of the liver improves compared with the homogenous region of the liver, we define $\beta$ as:

$$\beta = \delta_E - \delta_{E_0} + \epsilon$$

(3.2)

where $\delta_{E_0}$ is the mean gradient value inside the edge mask of the image before the enhancement, and $\epsilon$ is defined in section 3.1.1. It is expected as a result of edge enhancement, that the value of $\beta$ should be positive.
3.1.3 Measure of the bias field homogeneity, $\alpha$

We have reviewed some evaluation techniques used by other researchers regarding the bias field inhomogeneity correction, but all of them are not applicable to our application. Works in [17, 43, 53] use the root mean square error (RMSE) to assess their results, but this measure is only in use for evaluating synthetic MRI datasets. Likar et al. [96, 97], Thacker et al. [163], and Viola [169] use a standard entropy measure to find the set of parameters which can optimise their correction. Thacker et al. assume that the optimal correction produces maximum entropy. On the other hand, Likar et al. and Viola assume that the optimal correction produces the minimum entropy. As there are controversial opinions on this method, we consider that this measure is not a very reliable to be used. Work by Gelber et al. [50], simply ask the radiologist to evaluate their output based on visual inspection. But this technique is too subjective and it cannot provide a quantitative measure.

We measure the bias field homogeneity based on the assumption that the brightness of the image is distributed evenly in the corrected image. This assumption agrees with the work by Brinkmann et al. [17] who state that the local mean of the corrected image should fit its global mean intensity. To calculate how much the inhomogeneity of the field was reduced in the output test image, we consider for this type of dataset, an equal pixel division of the body mask region into three subsections in the vertical direction (see Figure 3.5). We divide the body mask into three subsections because we want to breach the bright areas from the dark area in our training dataset (see Figure 3.1).

![Figure 3.5: An example showing the body mask and its corresponding subsections ($i = 1, 2, 3$). Each subsection has the same number of pixels.](image-url)
3.1. Training Dataset and the Quality Measures

For each subsection, \( i \), we find the mean grey value, \( \mu_i \). Next, we define \( m \) as the range of these local average values, i.e.

\[
m = \max_{i=1,2,3} \{ \mu_i \} - \min_{i=1,2,3} \{ \mu_i \}
\]

(3.3)

It is expected as a result of a good bias field correction, \( m \) to be near zero. Then, the measure of homogeneity, \( \alpha \), is given as:

\[
\alpha = \begin{cases} 
 m_0 - m & : \beta \geq 0 \\
 0 & : \text{otherwise}
\end{cases}
\]

(3.4)

where \( m_0 \) is the value of \( m \) before the bias field correction, and \( \beta \) is as defined in section 3.1.2. In this equation, we put the condition \((\beta \geq 0)\) because we want the correction to at least retain the strength of the liver edges of the input image. A good bias field correction technique should give high positive \( \alpha \) values.

3.1.4 Criterion used to find the optimal kernel size for smoothing

The optimal kernel size is found by finding the optimally smoothed output. The optimally smoothed output image can be selected according to its average contrast [40]. We convolve the input image with Gaussian kernels or median kernels of several sizes to produce several versions of the smoothed image.

To calculate the average contrast, 10000 pairs of pixels are selected randomly from each smoothed image. If the smoothed image was produced by using a smoothing kernel with size \((2N+1) \times (2N+1)\), the pixels that represent the pairs are \(0.6590/7\) positions apart from each other. The contrast can be represented by the summation of the absolute intensity difference between the pixels in the pairs.

According to Fairfield [40], the most optimally smoothed image can be considered to be that which has the first local minimum average contrast value. This process is shown schematically in Figure 3.6. In the case of insufficient smoothing, the average contrast is high because the existence of noise makes the intensity of the pixels representing the same region vary over a wide range. As the distance between the pixel pairs is proportional to the size of the smoothing kernel we use, when the image is over-smoothed, the comparison of the intensity values is between different regions, and thus
the average contrast is also expected to be high. However, when the image is extremely over-smoothed, the entire scene in the image is represented by almost the same intensity value, thus the contrast is low.

![Image contrast versus smooth level characteristics.](image)

**Figure 3.6:** Image contrast versus smooth level characteristics.

### 3.2 Problem Formulation

We can model the detected 2D MRI signal, \( f(x, y) \), as follows [43, 53, 160, 180]:

\[
    f(x, y) = b(x, y)s(x, y) + n(x, y)
\]  

(3.5)

where \( b(x, y) \) is the bias field, \( s(x, y) \) is the transmitted signal by the tissues, and \( n(x, y) \) is additive noise from the scanner. In the preprocessing stage, we hope to find a way to minimise \( n(x, y) \) and the spatial variation of \( b(x, y) \), so that \( f(x, y) \) is more like \( s(x, y) \).

### 3.3 Stage 1: Bias Field Correction Techniques

The earliest bias field correction technique requires the scan of a phantom, after each scan of the patient, to estimate the correction matrix [43, 53]. As the bias field is assumed as a multiplicative interference, the correction is done by dividing the acquired
image of the patient with this correction matrix [29, 54, 86, 174]. However, because
the scan of the patient and the scan of the phantom are not acquired at the same time,
an accurate image registration is needed to maximise the correction result. Besides, as
this technique requires an extra acquisition process, this increases the cost of each scan
[43].
Other researchers assume the correction matrix of the bias field to be the sum of some
basis functions. For example, Dawant et al. [30] uses as basis functions the thin-
plate splines, and Styner et al. [160] use orthogonal Legendre polynomials. They
approximate the correction matrix by finding the coefficient of these basis functions
which can minimise the defined fitting functions. However, the correction by this
technique is highly dependent to the basis functions used [43].
We consider that a filtering approach is more applicable to MRI abdominal. In order
to do this, first we assume that the additive noise from the scanner, \( n(x, y) \) in equation
(3.5), is negligible (i.e. \( n(x, y) = 0 \)). Thus, we get the following equation:
\[
    f(x, y) = b(x, y)s(x, y)
\]
(3.6)
The bias field, \( b(x, y) \) can be assumed to change slowly and thus it can be considered as
a low frequency multiplicative interference. On the other hand, the transmitted signal
by the tissues, \( s(x, y) \), represents the imaged scene with all its details, thus contains
more high frequency components than \( b(x, y) \). So, by reducing the value of the low
frequency components, the variation of \( b(x, y) \) becomes less significant [43].
In this section, we will evaluate three filtering techniques, namely homomorphic filtering
in the frequency domain [3, 60, 125], homomorphic unsharp masking (HUM) [17], and
local enhancement [10, 55, 125]. Homomorphic filtering in the frequency domain and
HUM are among the common techniques to tackle the bias field inhomogeneity problem
in MRI [43]. But, as far as we are concerned, our work is the first attempt to use a
local enhancement technique to correct the bias field inhomogeneity in MRI.
This section is divided into four subsections. Section 3.3.1 presents a review of the
use of homomorphic filtering in the frequency domain. Section 3.3.2 reviews HUM,
and section 3.3.3 reviews the local enhancement technique. The experimental results
of these three methods and their evaluations can be found in section 3.3.4.
3.3.1 Homomorphic filtering in the frequency domain

The implementation of homomorphic filtering in the frequency domain is shown in Figure 3.7. First of all, we create the logarithm image, \( f'(x, y) \), from the input image, \( f(x, y) \). This is because it is well known that multiplicative interference can be converted into additive interference by taking the logarithm of the observed field:

\[
f'(x, y) = \ln(f(x, y)) = \ln(b(x, y)) + \ln(s(x, y))
\]  

\( (3.7) \)

\[
G(u,v) = H(u,v)F(u,v)
\]

\( (3.8) \)

\[
g'(x,y) = \exp(g'(x,y))
\]

\( (3.9) \)

Then, \( f'(x, y) \) needs to be transformed into the frequency domain. An image can be transformed from the spatial domain, \((m, n)\), into the frequency domain, \((k, l)\), by taking its discrete Fourier transform (DFT). The DFT for a two dimensional image, \( f(m, n) \), of dimensions \( M \times N \), is given by:

\[
F(k,l) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) \exp \left( -j2\pi \left[ \frac{km}{M} + \frac{ln}{N} \right] \right)
\]

\( (3.8) \)

Because of the linear property of DFT, we have:

\[
F(u,v) = B(u,v) + S(u,v)
\]

\( (3.9) \)

where \( F(u,v), B(u,v), \) and \( S(u,v) \) are the DFTs of \( \ln(f(x, y)), \ln(b(x, y)), \) and \( \ln(s(x, y)) \), respectively.

The idea of homomorphic filtering is to improve the homogeneity of the image by altering its low frequency components. Reducing the low frequency values makes the
variation of the bias field less significant. Petrou and Bosdogianni [125] suggest that the DFT of the image is multiplied by the transfer function \( H(u,v) \):

\[
H(u,v) = \frac{1}{1 + e^{-\pi(u^2 + v^2) - \omega_0}} + A \quad (3.10)
\]

The parameters of this filter (i.e. \( A, s, \) and \( \omega_0 \)) are related to the upper limit (\( \gamma_H \)), and the lower limit (\( \gamma_L \)) of the filter function shown in Figure 3.8. These parameters can be expressed in terms of parameters \( s, \omega_0, \) and \( A \) by the following equations:

\[
\gamma_H = 1 + A \quad (3.11)
\]
\[
\gamma_L = (1 + e^{s\omega_0})^{-1} + A \quad (3.12)
\]

![Figure 3.8: A cross section of a homomorphic filter, \( H(r) \), as a function of polar frequency, \( r = \sqrt{u^2 + v^2} \). (Taken from [125].)](image)

After this filtering, the image, (i.e. \( G(u,v) = H(u,v)F(u,v) \)), needs to be transformed back into the spatial domain by using the inverse DFT:

\[
g'(m,n) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} G(k,l) \exp(\frac{j2\pi}{M} [km + ln]) \quad (3.13)
\]

One has then to take the anti-logarithm of \( g'(x,y) \), and round the values to the nearest integer to produce image \( g(x,y) \).

However, since the bias field components have been reduced, \( g(x,y) \) is darker than \( f(x,y) \) in most cases (see Figure 3.9). Thus, we need to normalise \( g(x,y) \) with respect to \( f(x,y) \). We achieve this by performing the following correction:

\[
g_n(x,y) = \frac{M_f}{M_g} g(x,y) \quad (3.14)
\]
where \( g_n(x, y) \) is the normalised output value at position \((x, y)\), \( M_f \) is the global mean value of the input image \( f(x, y) \), and \( M_g \) is the global mean value of \( g(x, y) \). This normalisation ensures that homomorphic filtering does not reduce the overall brightness of the image, but only redistributes the brightness from the low frequency components to the higher frequency components.

\[
\frac{g_n(x, y)}{M_g} = \frac{g(x, y)}{M_f}
\]

This normalisation ensures that homomorphic filtering redistributes the brightness inside \( f(x, y) \), while keeping the overall brightness the same.

### 3.3.2 Homomorphic unsharp masking (HUM)

Brinkmann et al [17] noted that homomorphic filtering reduces the significance of the low frequency components. Thus, in the spatial domain, the same effect can be obtained by using an unsharp mask, where the output image is equal to the original image, minus the smoothed version of the image. With this, the high frequency components become relatively stronger.

The authors used the homomorphic unsharp mask (HUM), which does not involve smoothing, and the logarithm transform. By using a binary mask, which defines the area of interest, they calculated the global mean value, \( M \). Then, by using a sliding window, they calculated the local mean value, \( m(x, y) \). Their idea was that, without the presence of the bias field in the image, the local mean value must be equal to the global mean value. Thus, the intensity of the output pixel was calculated as:

\[
g(x, y) = \frac{M}{m(x, y)} f(x, y)
\]
where \( g(x, y) \) was the intensity of the output image, and \( f(x, y) \) was the intensity of the input image.

### 3.3.3 Local enhancement

Another way to enhance the high frequencies at the expense of low frequencies in the image is to apply local image enhancement which is similar to homomorphic unsharp masking (HUM). The formula used to do such an enhancement is [112]:

\[
g(x, y) = \frac{kM}{\sigma(x, y)} [f(x, y) - m(x, y)] + m(x, y)
\]

(3.16)

where \( g(x, y) \) is the output pixel value, \( f(x, y) \) is its corresponding input value, \( k \) is a constant, \( M \) is the global mean value of the input image, while \( m(x, y) \) and \( \sigma(x, y) \) are the local mean value and the local standard deviation, respectively, inside the sliding window at position \((x, y)\).

### 3.3.4 Results of the bias field correction

The training dataset was processed by three different techniques, in order to overcome the bias field inhomogeneity problem. These techniques are, homomorphic filtering in the frequency domain, homomorphic unsharp masking (HUM), and local enhancement.

For each correction technique, we use the hill climbing method, to identify the set of parameters that produces the optimal result. The optimal parameter set is the one that maximised the value of \( \bar{\sigma} \). (Our training dataset has the value of \( \bar{\delta}_{\text{Lo}} = 10.3345, \bar{\delta}_{\text{Es}} = 35.2152 \) and \( \bar{m}_o = 28.2948 \).)

#### Homomorphic filtering in the frequency domain

In this experiment, homomorphic filtering in the frequency domain is undertaken by using the transfer function, \( H(u, v) \), of equation (3.10). In this equation, there are three parameters that need to be adjusted (i.e. \( \omega_0 \), \( s \), and \( A \)). Figure 3.10 shows an example to illustrate the role of each parameter in this equation.
Chapter 3. 2D preprocessing of axial abdominal MRI images

Parameter $\omega_0$ corresponds to the position where the value of $H(u, v)$ is equal to $(0.5 + A)$. However, in analogy to high pass filtering, we can assume that $\omega_0$ corresponds to the cutoff frequency. This parameter controls the number of low frequency components that we consider represent the bias field in the image.

Parameter $s$ relates to the ratio of the upper limit, $\gamma_H$, to the lower limit, $\gamma_L$, of the transfer function. The ratio increases when $s$ increases. Also from the shape of $H(u, v)$, we can deduce that $s$ is proportional to the steepness of the filter.
Positive $A$ increases the value of $H(u,v)$. On the other hand, negative values of $A$ (i.e. $\gamma_L < A < 0$) reduce the value of $H(u,v)$. Thus, we conclude that parameter $A$ is inversely proportional to the ratio of $\gamma_H : \gamma_L$.

As we do not know which parameter is the most critical parameter in equation (3.10), we tried all possible combinations of ordering the parameters in the hill climbing method. This equation has three parameters, thus we tried all six possible arrangement of parameter orders.

First, we consider parameter $\omega_0$ as the most critical parameter, followed by $s$ and $A$. By using the hill climbing method, we found that this parameter order gives maximum $\alpha$ when $\omega_0 = 40$, $s = 0.04$ and $A = 0.00$ (see Figure 3.11). With this parameter setting, $\alpha$ is equal to $17.0915$.

![Figure 3.11: The results by assuming parameter $\omega_0$ as the most critical parameter, followed by $s$ and $A$.](image)

Then, we consider parameter $\omega_0$ as the most critical parameter, followed by $A$ and $s$. We found that this parameter order gives maximum $\alpha$ when $\omega_0 = 40$, $s = 0.05$ and $A = 0.02$ (see Figure 3.12). With this parameter setting, $\alpha$ is equal to $17.0004$.
Chapter 3. 2D preprocessing of axial abdominal MRI images

Figure 3.12: The results by assuming parameter \( w_0 \) as the most critical parameter, followed by \( A \) and \( s \). (a) First, we give both parameters \( s \) and \( A \) random constant values (i.e. \( s = 0.05 \) and \( A = 0.00 \)), and vary the value of \( w_0 \) until we found maximum \( \alpha \) (i.e. \( w_0 = 40 \)). (b) Then, by keeping the value of \( s \) as before (i.e. \( s = 0.05 \)) and setting the value of \( w_0 = 40 \), we vary the value of \( A \). We found maximum \( \alpha \) when \( A = 0.02 \). (c) Finally, we keep \( w_0 = 40 \) and \( A = 0.02 \), and vary the value of \( s \). This arrangement of parameter order produces the maximum \( \bar{\alpha} \) (i.e. \( \bar{\alpha} = 17.0994 \)) when \( w_0 = 40, s = 0.05, \) and \( A = 0.02 \).

The next combination is when we consider parameter \( A \) as the most critical parameter, followed by \( s \) and \( w_0 \). We found that this parameter order gives maximum \( \bar{\alpha} \) when \( w_0 = 45, s = 0.02 \) and \( A = 0.05 \) (see Figure 3.13). With this parameter setting, \( \bar{\alpha} \) is equal to 17.0994.

Figure 3.13: The results by assuming parameter \( A \) as the most critical parameter, followed by \( s \) and \( w_0 \). (a) First, we give both parameters \( w_0 \) and \( s \) constant values (i.e. \( w_0 = 40 \) and \( s = 0.05 \)), and vary the value of \( A \) until we found maximum \( \alpha \) (i.e. \( A = 0.02 \)). (b) Then, by keeping the value of \( w_0 \) as before (i.e. \( w_0 = 40 \)) and setting the value of \( A = 0.02 \), we vary the value of \( s \). We found maximum \( \alpha \) when \( s = 0.05 \). (c) Finally, we keep \( s = 0.05 \) and \( A = 0.02 \), and vary the value of \( w_0 \). This arrangement of parameter order produces the maximum \( \bar{\alpha} \) (i.e. \( \bar{\alpha} = 17.0994 \)) when \( w_0 = 45, s = 0.05, \) and \( A = 0.02 \).
The fourth combination is when we consider parameter $A$ as the most critical parameter, followed by $\omega_0$ and $s$. We found that this parameter order gives maximum $\bar{\alpha}$ when $\omega_0 = 45$, $s = 0.04$ and $A = 0.02$ (see Figure 3.14). With this parameter setting, $\bar{\alpha}$ is equal to 17.2441.

Figure 3.14: The results by assuming parameter $A$ as the most critical parameter, followed by $\omega_0$ and $s$. (a) First, we give both parameters $\omega_0$ and $s$ constant values (i.e. $\omega_0 = 40$ and $s = 0.05$), and vary the value of $A$ until we found maximum $\alpha$ (i.e. $A = 0.02$). (b) Then, by keeping the value of $s$ as before (i.e. $s = 0.05$) and setting the value of $A = 0.02$, we vary the value of $\omega_0$. We found maximum $\alpha$ when $\omega_0 = 45$. (c) Finally, we keep $\omega_0 = 45$ and $A = 0.02$, and vary the value of $s$. This arrangement of parameter order produces the maximum $\bar{\alpha}$ (i.e. $\bar{\alpha} = 17.2441$) when $\omega_0 = 45$, $s = 0.04$, and $A = 0.02$.

Figure 3.15: The results by assuming parameter $s$ as the most critical parameter, followed by $A$ and $\omega_0$. (a) First, we give both parameters $\omega_0$ and $A$ constant values (i.e. $\omega_0 = 40$ and $A = 0.0$), and vary the value of $s$ until we found maximum $\alpha$ (i.e. $s = 0.04$). (b) Then, by keeping the value of $\omega_0$ as before (i.e. $\omega_0 = 40$) and setting the value of $s = 0.04$, we vary the value of $A$. We found maximum $\alpha$ when $A = 0.0$. (c) Finally, we keep $s = 0.04$ and $A = 0.0$, and vary the value of $\omega_0$. This arrangement of parameter order produces the maximum $\bar{\alpha}$ (i.e. $\bar{\alpha} = 17.5444$) when $\omega_0 = 55$, $s = 0.04$, and $A = 0.0$. 
Next, we consider parameter $s$ as the most critical parameter, followed by $A$ and $\omega_0$. We found that this parameter order gives maximum $\alpha$ when $\omega_0 = 55$, $s = 0.04$ and $A = 0.0$ (see Figure 3.15). With this parameter setting, $\alpha$ is equal to 17.5444.

The last arrangement of the parameter order is by considering parameter $A$ as the most critical parameter, followed by $\omega_0$ and $s$. We found that this parameter order gives maximum $\alpha$ when $\omega_0 = 55$, $s = 0.03$ and $A = 0.00$ (see Figure 3.16). With this parameter setting, $\alpha$ is equal to 17.7100.

![Figure 3.16](image)

(a) First, we give both parameters $\omega_0$ and $A$ constant values (i.e. $\omega_0 = 40$ and $A = 0.0$), and vary the value of $s$ until we found maximum $\alpha$ (i.e. $s = 0.04$). (b) Then, by keeping the value of $A$ as before (i.e. $A = 0.0$) and setting the value of $s = 0.04$, we vary the value of $\omega_0$. We found maximum $\alpha$ when $\omega_0 = 55$. (c) Finally, we keep $\omega_0 = 55$ and $A = 0.00$ and vary the value of $s$. This arrangement of parameter order produces the maximum $\alpha$ (i.e. $\alpha = 17.7100$) when $\omega_0 = 55$, $s = 0.03$, and $A = 0.0$.

<table>
<thead>
<tr>
<th>Order of importance</th>
<th>Parameter values contribute to maximum $\alpha$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega_0 \rightarrow s \rightarrow A$</td>
<td>$\omega_0 = 40$, $s = 0.04$, $A = 0.00$</td>
<td>17.0915</td>
</tr>
<tr>
<td>$\omega_0 \rightarrow A \rightarrow s$</td>
<td>$\omega_0 = 40$, $s = 0.05$, $A = 0.02$</td>
<td>17.0004</td>
</tr>
<tr>
<td>$A \rightarrow s \rightarrow \omega_0$</td>
<td>$\omega_0 = 45$, $s = 0.05$, $A = 0.02$</td>
<td>17.0994</td>
</tr>
<tr>
<td>$A \rightarrow \omega_0 \rightarrow s$</td>
<td>$\omega_0 = 45$, $s = 0.04$, $A = 0.02$</td>
<td>17.2441</td>
</tr>
<tr>
<td>$s \rightarrow A \rightarrow \omega_0$</td>
<td>$\omega_0 = 55$, $s = 0.04$, $A = 0.00$</td>
<td>17.5444</td>
</tr>
<tr>
<td>$s \rightarrow \omega_0 \rightarrow A$</td>
<td>$\omega_0 = 55$, $s = 0.03$, $A = 0.00$</td>
<td>17.7100</td>
</tr>
</tbody>
</table>

Table 3.1: The values of maximum $\alpha$ obtained from all possible parameter orders (in term of their assumed relative significance) for homomorphic filtering in the frequency domain.

Table 3.1 compares the maximum $\alpha$ produced by each combination of parameter orders. From this table, it is shown that $\alpha$ is at its highest when $\omega_0 = 55$, $s = 0.04$ and
$A = 0.00$. Thus, we consider this parameter setting as the optimal parameter setting for homomorphic filtering in the frequency domain for this training dataset. This table also shows that the best order of parameters is in the order of $s$ followed by $\omega_0$, and followed by $A$.

**Homomorphic unsharp masking (HUM)**

In the implementation of the homomorphic unsharp masking (HUM), only one parameter needs to be tuned, which is the size of the sliding window. In this work, we ranging the size of the sliding window from size $3 \times 3$ to size $193 \times 193$. However, all the results we obtained produce $\tilde{\alpha} = 0$ due to the strength of the liver edges, $\tilde{\beta}$, less than zero (see equation (4.5)).

![Figure 3.17: The bottom most slice from several outputs of HUM. (a) The output of HUM with size $3 \times 3$. (b) The output of HUM with size $19 \times 19$. (c) The output of HUM with size $51 \times 51$. (d) The output of HUM with size $193 \times 193$.](image-url)
As the measurements of $\alpha$ cannot help us to decide the best output from this processing, we judge the output mainly based on the visual inspection. From this inspection, we can conclude that the smaller kernel of HUM tends to reduce the inter-organ contrast, while the bigger kernel tends to retain the bias field inhomogeneity problem. An example is shown in Figure 3.17. From this figure, we assume that the kernel size of $19 \times 19$ produces the best result of HUM, as the output seems to be uniform, and the shape of the organs are still visible.

**Local enhancement**

In the implementation of the local enhancement, two parameters need to be tuned, namely the size of the kernel and $k$ (see equation (3.16)). We first consider that the size of the kernel is a more critical parameter than $k$. By using the hill climbing method, we found that this parameter order gives maximum $\alpha$ when the size of the kernel is equal to $13 \times 13$ and $k = 10000$ (see Figure 3.18). With this parameter setting, $\alpha$ is equal to 27.1267.

![Diagram](a)

![Diagram](b)

Figure 3.18: The results by assuming the size of the filter is more critical than parameter $k$. (a) First, we optimise the choice of the kernel size by keeping parameter $k$ constant (i.e. $k = 10$). From this, we found maximum $\bar{\alpha}$ when the size of kernel is equal to $13 \times 13$. (b) Then, by keeping the size of the kernel equal to $13 \times 13$, we vary the value of $k$. The graph shows that the value of $\bar{\alpha}$ increases when $k$ increases. Thus, we conclude that a size of $13 \times 13$ and $k = 10000$ are adequate for representing the good parameter setting for this order set.

Then, we consider $k$ to be more important than the size of the kernel. We found that this parameter order also gives the maximum $\bar{\alpha}$ when the size of the kernel is equal
3.3. Stage 1: Bias Field Correction Techniques

to $13 \times 13$ and $k = 10000$ (see Figure 3.19). As both parameter orders lead to the same parameter setting, the significance of the size of the kernel and parameter $k$ are interchangable.

![Figure 3.19](image)

**Figure 3.19:** The results by assuming parameter $k$ is more critical than the size of the filter. (a) First, keep the size of the kernel equal to $9 \times 9$, and vary the value of $k$. The graph shows that the value of $\hat{\sigma}$ increases when $k$ increases. Thus, in the range of 0 to 10000, $k = 10000$ gives the maximum $\hat{\sigma}$. (b) We then set the value of $k$ equal to 10000, and change the size of the kernel. We obtained the maximum $\hat{\sigma}$ (i.e. $\hat{\sigma} = 27.1267$) when the size of kernel is equal to $13 \times 13$.

**Comparison between bias field correction methods**

In this section, we compare the best result from each technique with the original testing dataset. Table 3.2 shows the measurements of $\hat{\sigma}$ of the training dataset and the best output from each technique. This table shows that the output of the local enhancement has the biggest value of $\hat{\sigma}$, thus this technique is the best bias field correction technique among the methods tested in our work.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Parameter setting</th>
<th>$\hat{\sigma}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original input</td>
<td>-</td>
<td>0.0000</td>
</tr>
<tr>
<td>Homomorphic filtering in the frequency domain</td>
<td>$\omega_0 = 55, A = 0.0, s = 0.03$</td>
<td>17.7190</td>
</tr>
<tr>
<td>Homomorphic unsharp masking (HUM)</td>
<td>kernel size = $19 \times 19$</td>
<td>0.0000</td>
</tr>
<tr>
<td>Local enhancement</td>
<td>kernel size = $13 \times 13$, $k = 10000$</td>
<td>27.1267</td>
</tr>
</tbody>
</table>

**Table 3.2:** The bias field inhomogeneity measurement, $\hat{\sigma}$, and the strength of the edges, $\hat{\beta}$ of the input dataset and the optimal output from each bias field correction technique used in this work.
This argument is also supported by visual inspection. We show the bottom most slice, in Figure 3.20, as an example. Although the local enhancement improves the intra-organ homogeneity, the edges of the liver can be perceived easily in this image compared to the others. The brightness of the image also can be considered distributed evenly.

Figure 3.20: The bottom most slice from (a) the input image, (b) the best output of homomorphic filtering in the frequency domain, (c) the best output from homomorphic unsharp masking (HUM), and (d) the best output from local enhancement technique. The liver is the largest organ which can be seen on the left side of the image. This demonstrates that all three methods effectively reduce the bias field inhomogeneity to negligible levels. However, the HUM produced an unacceptable loss of contrast.

Figure 3.21 shows all the slices in the output of local enhancement technique (using kernel size 13 × 13 and \( k = 10000 \)). If these processed data are compared with the input data (Figure 3.1) then we can observe significant hidden detail in the original image has been clearly resolved in the processed image.
3.4 Stage 2: Additive Noise Reduction Techniques

In this section, we consider that the problem of the bias field inhomogeneity is already solved, so we assume $b(x, y)$ in equation (3.5) is equal to 1. Thus we get this equation:

$$f(x, y) = s(x, y) + n(x, y)$$

(3.17)
Petrou and Bosdogianni [125] point out that the spectrum of \( n(x, y) \) dominates the spectrum of \( s(x, y) \) for high frequency components. However, in order to correct the bias field previously, we reduced the significance of the low frequency components, thus at the same time, we increased the strength of the additive noise.

The above statement is in agreement with the bias field correction result we obtained in section 3.3. The output of section 3.3, which is shown in Figure 3.21, has greater intra-organ variation compared with its corresponding input dataset, shown in Figure 3.1. This condition still makes the segmentation algorithm which relies on the edge information, such as the intelligent scissors, difficult, because the algorithm may get confused by the abundance of strong gradient information in the image.

In this section, our aim is to reduce the intra-organ variation inside the liver of Figure 3.21. We tried two common approaches for reducing the level of noise, namely, median filtering and Gaussian smoothing. These methods will be evaluated based on the measure of \( \bar{e} \), which was defined in section 3.1.1. The best method among these two is the one with the biggest value of \( \bar{e} \).

This section is divided into three subsections. Section 3.4.1 presents a review of median filtering. Section 3.4.2 presents a review of Gaussian smoothing, and section 3.4.3 presents the experimental results of these two techniques.

### 3.4.1 Median filtering

Median filtering is the most commonly used technique to reduce the additive impulse noise in digital images [10, 125]. The algorithm also uses a sliding window. It arranges the intensities within this window in an increasing order, finds the median value of this population, and assigns this value to the corresponding output pixel.

As an example, let us consider that we use a sliding window of size 3 x 3. Assume that at position \((x, y)\) in the input, within this sliding window, we get these intensity values: 110, 130, 110, 160, 100, 199, 101, 180, 123. After the arrangement, we get this order: 100, 101, 110, 110, 123, 130, 160, 180, 199. The output at pixel \((x, y)\) then takes value 123, which is the value of the middle position in the ordered set (i.e. the median value).
3.4.2 Gaussian smoothing

Gaussian smoothing is one of the most popular techniques used to eliminate Gaussian noise from the input image.

![Gaussian filter cross-section](image)

Figure 3.22: The cross-section of a one dimensional Gaussian filter. The filter is symmetric about its centre, and the width of the filter is equal to $2N+1$.

The kernel size of a Gaussian filter is normally defined as $2N+1$ where $N$ is an integer value (see Figure 3.22). That is why Gaussian kernels usually have odd size. A two dimensional Gaussian, $G(j, k)$, can be defined as:

$$G(j, k) = e^{-\frac{j^2+k^2}{2\sigma^2}}$$  \hspace{1cm} (3.18)

where $j$ and $k$ are the coordinates relative to the centre of the kernel, and $\sigma$ is the standard deviation used.

For efficient smoothing, the ratio of the Gaussian value at its edges relative to the value at the centre, must be small. To fulfill this requirement, we must select an appropriate size for the filter depending on its standard deviation, $\sigma$. Let us consider the value of $G(0, N)$, which represents the ratio of the filter function at the edge of the kernel to its value at the centre. Thus we get a relationship between the size of the kernel, and its standard deviation.

$$G(0, N) = e^{-\frac{N^2}{2\sigma^2}} \Rightarrow N = \sigma \sqrt{-2 \ln G(0, N)}$$  \hspace{1cm} (3.19)

Because the output of a uniform intensity region must remain unchanged, the sum of...
the weights of the Gaussian kernel must be equal to 1.0:

$$G_n(j, k) = \frac{G(j, k)}{\sum_{j=-N}^{N} \sum_{k=-N}^{N} G(j, k)}$$

(3.20)

where $G_n(j, k)$ is the desired normalised Gaussian kernel, while $G(j, k)$ is the Gaussian kernel produced using equation (3.18). An example of this calculation is shown in Figure 3.23.

![Figure 3.23](image)

Figure 3.23: An example of computation of a Gaussian kernel. In this case, the kernel size is $3 \times 3$ ($N = 1$) and its standard deviation, $\sigma$, is equal to 1/3.

### 3.4.3 Results on the additive noise reduction

From section 3.3.4, we conclude that the bias field inhomogeneity problem is now resolved by using the local enhancement technique, with kernel size $13 \times 13$ and $k$ equal to 10000. In this section, we try to reduce the additive noise from this locally enhanced dataset (i.e. Figure 3.21), by implementing median filtering and Gaussian smoothing. A good noise reduction technique should give a high positive value of $\bar{\varepsilon}$.

For both techniques, we convolve the slices shown in Figure 3.21 on a slice-by-slice basis, with square kernels, with odd dimensions, ranging from size $3 \times 3$ to $11 \times 11$. By using the technique previously described in section 3.1.4, we find the optimal smoothing level of each slice, automatically. Then, we calculate $\bar{\varepsilon}$ of the result.

#### Median filtering

We found that all slices of Figure 3.21 have the optimal size of median kernel equal to $5 \times 5$. The median filtered version of Figure 3.21 has the value of $\bar{\varepsilon}$ equal to 35.26.
3.4. Stage 2: Additive Noise Reduction Techniques

Gaussian smoothing

We found that all slices of Figure 3.21 have the optimal size of Gaussian kernel equal to $5 \times 5$. The Gaussian filtered version of Figure 3.21 has the value of $\bar{e}$ equal to 16.89.

Comparison between noise reduction techniques

From these experiments, we found that the median filtered version of Figure 3.21 has higher $\bar{e}$ compared with the Gaussianly smoothed version (i.e. 35.26 compared with 16.89). This indicates that the median filtered version is more homogenous (i.e. has lower intra-organ contrast). Thus, compared with Gaussian smoothing, we conclude that the median filtering approach is more applicable to our dataset. Each slice of the median filtered version is shown in Figure 3.24. However, we can see from this figure that the edges of the liver were also degraded due to smoothing. Thus we will try to re-enhance the edges by using toboggan contrast enhancement, and the work regarding this can be found in section 3.5.

Figure 3.24: The median filtered version of Figure 3.21.
3.5 Stage 3: Enhancement of the Edges

Due to smoothing, the output from the second stage of preprocessing (see section 3.4) has blurred liver edges. Thus, in this section, we take the image shown in Figure 3.24 as input, and try to re-enhance the edges by using a technique known as tobbogan contrast enhancement.

![Figure 3.25: Block diagram showing the implementation of tobbogan contrast enhancement.](image)

The tobbogan contrast enhancement technique has been introduced by Fairfield [40] to enhance the edges in an already smoothed image. The edges are improved by using the so-called inheritance algorithm, which changes the ramp edges into step edges. Figure 3.25 gives an overview on how tobbogan contrast enhancement can be implemented. Both smoothed image and gradient magnitude are used in the inheritance algorithm,
3.5. Stage 3: Enhancement of the Edges

which produces the toboggan image.

In this experiment, the gradient is calculated by using the Sobel operator as, after smoothing, noise is considered to be minimal. The kernels used for this purpose are shown in Figure 3.26. The gradient magnitude at pixel \( p \), \( G_m(p) \), is calculated by using formula:

\[
G_m(p) = \sqrt{[\delta_x(p)]^2 + [\delta_y(p)]^2}
\]  (3.21)

where \( \delta_x(p) \) is the component of the gradient vector along the \( x \) direction, and \( \delta_y(p) \) is the component of the gradient vector along the \( y \) direction.

![Figure 3.26: Kernels used in this experiment to calculate the gradient. (a) Sobel mask used to compute the gradient component in the \( x \) direction, \( \delta_x \). (b) Sobel mask used to compute the gradient component in the \( y \) direction, \( \delta_y \).](image)

The purpose of the inheritance algorithm is to change the ramp edges in the smoothed image into step edges. We refer to the resultant image as the toboggan image. The inputs to this algorithm are two images, representing the gradient magnitude of the smoothed image, and the smoothed image itself.

Using the information of the gradient magnitude, the algorithm slides the pixel values towards the lowest gradient pixel. The pixels can then be grouped together based on this sliding route. All pixels in the group are assigned the same intensity value, which is the intensity of the smoothed image at the position where the gradient is the lowest in that group. As a result, this eliminates the variation of intensity around the ramp edges, and produces sharp edges. An example is shown in the Figure 3.27.
Figure 3.27: The idea of the inheritance algorithm.

The implementation of inheritance algorithm is following these steps:

Step 1: Initialize all pixels in the output image as 'undefined'. Empty the 'stack'. Assign the location of the first pixel (i.e. the pixel at position (0, 0)) to variable 'Now'.
Step 2: Check whether the pixel on the output image at position 'Now' is 'undefined' or not. If still 'undefined', go to Step 3. Otherwise, go to Step 9.

Step 3: Assign variable 'Current' with the value of 'Now'.

Step 4: Define the neighbouring pixel of 'Current' as 'Neighbour'. Then calculate the different of gradient value between pixel at 'Current' and pixel at 'Neighbour' by using this formula:

\[ G_{\text{different}}(Current, Neighbour) = w(G_m(Current) - G_m(Neighbour)) \] (3.22)

where \( G_m \) represents gradient magnitude, and \( w \) represents weight value. If pixel 'Neighbour' is the horizontal or the vertical neighbour of pixel 'Current', \( w \) is equal to 1, and if pixel 'Neighbour' is the corner neighbour, \( w \) is equal to \( 1/\sqrt{2} \). If all \( G_{\text{different}} \) are negative, go to Step 6. Otherwise, assign the 'Next' variable with the position of 'Neighbour', which gives the biggest positive value of \( G_{\text{different}} \).

Step 5: Save (i.e. push) 'Current' into 'stack'. Check whether the pixel on the output image at position 'Next' is 'undefined' or not. If already 'defined', go to Step 7. Otherwise, assign the value of 'Next' to 'Current'. Repeat Step 4 and Step 5.

Step 6: Assign variable 'InheritedValue' with the intensity value of the smoothed image at pixel 'Current'. Save (i.e. push) 'Current' into 'stack'. Go to Step 8.

Step 7: Assign variable 'InheritedValue' with the value of pixel at position 'Next' on the output image.

Step 8: Read again (i.e. pop) the position stored in the 'stack'. For each position read from 'stack', assign the pixel on the output image at this value with value 'InheritedValue', and mark it as 'defined'.

Step 9: Update 'Now' value with the next value position (e.g. if the current 'Now' value is (0,0), 'Now' will be updated to (0,1)). Repeat Step 2 to Step 9 until all pixels in output image are 'defined'.

These steps can be represented by a flowchart as shown in Figure 3.28. An example on how the inheritance algorithm is implemented in a 5 × 5 image is shown in Figure 3.29.
Figure 3.28: A flowchart to represent the implementation of inheritance algorithm.
3.5. Stage 3: Enhancement of the Edges

<table>
<thead>
<tr>
<th>Gradient magnitude of smoothed image</th>
<th>Smoothed image</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>323 299 261 214 169</td>
<td>105 121 131 138 143</td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>214 159 132 54 20</td>
<td>145 152 155 154 152</td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>42 9 44 67 77</td>
<td>161 160 157 151 146</td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>79 98 96 86 72</td>
<td>155 150 145 140 135</td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>94 82 65 48 38</td>
<td>141 136 133 131 129</td>
<td>● ● ● ● ●</td>
</tr>
</tbody>
</table>

(a) At the beginning, all pixels in the output array are ‘undefined’.

<table>
<thead>
<tr>
<th>Gradient magnitude of smoothed image</th>
<th>Smoothed image</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>323 299 261 214 169</td>
<td>105 121 131 138 143</td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>214 159 132 54 20</td>
<td>145 152 155 154 152</td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>42 9 44 67 77</td>
<td>161 160 157 151 145</td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>79 98 96 86 72</td>
<td>155 150 145 140 135</td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>94 82 65 48 38</td>
<td>141 136 133 131 129</td>
<td>● ● ● ● ●</td>
</tr>
</tbody>
</table>

(b) The algorithm starts by inspecting pixel (0,0). Pixels are grouped together as the algorithm flows to find the lowest gradient value. This group of pixels is given the intensity value of the smoothed image at position (1,2), which is the position of the lowest gradient value in this group. The corresponding pixels in the output now have value 160.

<table>
<thead>
<tr>
<th>Gradient magnitude of smoothed image</th>
<th>Smoothed image</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>323 299 261 214 169</td>
<td>105 121 131 138 143</td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>214 159 132 54 20</td>
<td>145 152 155 154 152</td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>42 9 44 67 77</td>
<td>161 160 157 151 145</td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>79 98 96 86 72</td>
<td>155 150 145 140 135</td>
<td>● ● ● ● ●</td>
</tr>
<tr>
<td>94 82 65 48 38</td>
<td>141 136 133 131 129</td>
<td>● ● ● ● ●</td>
</tr>
</tbody>
</table>

(c) Next, the gradient flow starts at position (1,0). The flow is towards point (1,2) again. However, because the flow is via the point which is already ‘defined’ in the output, to save computational time, the gradient flow will stop at pixel (1,1). The output, i.e. pixel (1,0), inherits the output value at position (1,1), i.e. 160.

Figure 3.29: An example of the inheritance algorithm applied to a 5 x 5 image. The pixels are grouped together based on the slice route towards the lowest gradient magnitude.
Chapter 3. 2D preprocessing of axial abdominal MRI images

Gradient magnitude of smoothed image | Smoothed image | Output
---|---|---
323 299 261 214 169 | 109 121 133 138 143 | 160 160 160 * *
214 159 103 54 20 | 145 152 155 154 152 | * 160 160 * *
42 9 44 67 77 | 161 159 157 151 145 | * * * *
79 95 96 86 72 | 155 150 145 140 135 | * * * *
94 82 65 48 38 | 141 136 133 131 129 | * * * *

(d) Then, the flow begins at position (2, 0). The flow is also pointing to pixel (1, 2), thus pixels (2, 0) and (2, 1) in the output take value of 160.

Gradient magnitude of smoothed image | Smoothed image | Output
---|---|---
323 299 261 214 169 | 109 121 133 138 143 | 160 160 160 152 *
214 159 103 54 20 | 145 152 155 154 152 | * 160 160 152 *
42 9 44 67 77 | 161 159 157 151 145 | * * * *
79 95 96 86 72 | 155 150 145 140 135 | * * * *
94 82 65 48 38 | 141 136 133 131 129 | * * * *

(e) Following the same procedure, the flow begins at point (3, 0) and stops at position (4, 0). The output at pixels (3, 0), (3, 1), and (4, 1) take value 152.

Gradient magnitude of smoothed image | Smoothed image | Output
---|---|---
323 299 261 214 169 | 109 121 133 138 143 | 160 160 160 152 *
214 159 103 54 20 | 145 152 155 154 152 | * 160 160 152 *
42 9 44 67 77 | 161 159 157 151 145 | * * * *
79 95 96 86 72 | 155 150 145 140 135 | * * * *
94 82 65 48 38 | 141 136 133 131 129 | * * * *

(f) The flow begins at position (4, 0) and ends at position (4, 1). The output at pixel (4, 0) takes value 152.

Figure 3.29: Continued.
3.5. Stage 3: Enhancement of the Edges

The algorithm continues with the next line, i.e., at position (0,1). The flow stops at position (2,1). Pixels (0,1) and (0,2) in the output take value 160.

Pixel (1,1) is being inspected. As the output pixel at this position is already 'defined', no operation is being carried out.

The algorithm is repeated for every pixel in the image from left to right, and top to bottom. At the end, all pixels in the output image have values assigned to them. As shown in the output, the edges become sharper, and the image can be divided into three regions more easily compared with the original smoothed image.

Figure 3.29: Continued.
3.5.1 Results of edge enhancement

In this section, we apply toboggan contrast enhancement to each slice shown in Figure 3.24 to re-enhance the liver edges. All slices of the toboggan contrast enhanced dataset are shown in Figure 3.30.

Figure 3.30: The toboggan enhanced version of Figure 3.24.
The re-enhanced dataset has the value of $\beta$ equal to 5.83. As a positive value of $\beta$ shows that this technique successfully re-enhanced the liver edges, we decide to keep toboggan contrast enhancement as part of our preprocessing chain.

### 3.6 Conclusions

Our preprocessing consists of three stages. The first stage deals with the bias field inhomogeneity correction. Three techniques, namely, homomorphic filtering in the frequency domain, homomorphic unsharp masking, and local enhancement, have been investigated. From section 3.3.4, we found that local enhancement is the most suitable technique to be implemented for our dataset, as this technique reduces the field inhomogeneity significantly, as indicated by the high value of $\alpha$.

However, the result from local enhancement has a significant level of noise, as indicated by the high intra-organ contrast. Thus, we introduced the second stage of preprocessing to deal with the reduction of additive noise. Our work considered median filtering, and Gaussian smoothing. In section 3.4.3, the median filtered dataset gave higher $\varepsilon$ compared with the Gaussianly filtered version. So, we consider median filtering as more suitable for our dataset compared with Gaussian smoothing.

Yet, as the second stage of our preprocessing involves smoothing, the edge of the liver became blurred. We introduced the third stage of the preprocessing chain to re-enhance the liver edges. We used a technique known as toboggan contrast enhancement. From section 3.5.1, we demonstrated that the liver edges in the smoothed dataset can be re-enhanced by applying toboggan contrast enhancement.

![Preprocessing stage](image)

**Figure 3.31:** The block diagram showing the combination to be use for the preprocessing stage.

From section 3.3.4, section 3.4.3 and section 3.5.1, of the methods considered, we decide that our preprocessing chain will consist of local enhancement, followed by median
filtering, and finally toboggan contrast enhancement. This preprocessing chain is shown in Figure 3.31. Next, we will try to extend this 2D preprocessing into 3D. This work will be carried out in Chapter 4.
Chapter 4

3D preprocessing of MRI dataset

MRI datasets are often corrupted by artefacts such as rf bias field inhomogeneity, and random additive noise, which can affect the segmentation result significantly [17, 25, 30, 43]. Thus, we proposed to first preprocess our dataset before undertaking the segmentation process. From previous chapter (i.e. Chapter 3), we decided to use a preprocessing chain that consists of 2D local enhancement, followed by 2D median filtering, and 2D toboggan contrast enhancement. In this chapter, we expanded this 2D preprocessing chain to 3D, and investigate the benefits of doing so.

This chapter is divided into four sections. Section 4.1 reviews the methods used in each stage of 3D preprocessing. Section 4.2 defines the measures of quality we used. Section 4.3 presents the results and evaluation of the technique by using a training dataset. Section 4.4 presents the conclusion of this chapter.

4.1 3D Preprocessing Chain

![Image of 3D preprocessing chain]

Figure 4.1: The proposed 3D preprocessing to be used in this experiment.
Chapter 4. 3D preprocessing of MRI dataset

In this experiment, our preprocessing chain consists of three consecutive stages, which are: (1) 3D local enhancement, (2) 3D median filtering and (3) 3D toboggan contrast enhancement (see Figure 4.1). Each of these stages is described in sections 4.1.1, 4.1.2 and 4.1.3, respectively.

4.1.1 3D local enhancement

We use a local enhancement technique to reduce the bias field inhomogeneity in the dataset. The technique enhances the high frequencies at the expense of low frequencies in the image. Similar to 2D local enhancement that was defined by equation (3.16), 3D local enhancement also works by using a sliding window. Instead of using a square window, 3D local enhancement uses a cubic window. The enhancement can be attained by the following formula:

\[
g(x, y, z) = \frac{kM}{\sigma(x, y, z)} [f(x, y, z) - m(x, y, z)] + m(x, y, z) \tag{4.1}
\]

where \(g(x, y, z)\) is the output voxel value, \(f(x, y, z)\) is its corresponding input value, \(k\) is a constant, \(M\) is the global mean value of the input dataset, while \(m(x, y, z)\) and \(\sigma(x, y, z)\) are the local mean value and the local standard deviation, respectively, inside the sliding cubic window centred at position \((x, y, z)\).

4.1.2 3D median filtering

Median filtering is the most commonly used technique to reduce the additive impulse noise in digital images [125]. In this 3D implementation, the algorithm uses a cubic sliding window. It arranges the intensities within this window in an increasing order, finds the median value of this population (i.e. the value of the middle position in this ordered set), and assigned this value to the corresponding output voxels.

4.1.3 3D toboggan contrast enhancement

Toboggan contrast enhancement has been introduced by Fairfield [40] to enhance the edges in an already smoothed dataset. The edges are improved by using the so called
inheritance algorithm which changes the ramp edges into step edges. The inputs to inheritance algorithm are two datasets, representing the gradient magnitude of the smoothed dataset, and the smoothed dataset itself.

The gradient magnitude at voxel \( p \), \( G_m(p) \), is calculated by using formula:

\[
G_m(p) = \sqrt{(\delta_x(p))^2 + (\delta_y(p))^2 + (\delta_z(p))^2}
\]

(4.2)

where \( \delta_x(p) \) is the component of the gradient vector along the \( x \) direction, \( \delta_y(p) \) is the component of the gradient vector along the \( y \) direction, and \( \delta_z(p) \) is the component of the gradient vector along the \( z \) direction.

Figure 4.2: 3D kernels used in this experiment to calculate the gradient magnitude when the voxel’s dimensions in \( x, y \) and \( z \) direction are equal to 1.0 mm, 1.0 mm and 1.2 mm, respectively.

To avoid the effect of the anisotropic sampling, we calculate these gradient components by using \( 3 \times 3 \times 3 \) kernels, proposed by Zucker and Hummel [184], which rely on the dimensions of the voxels. A brief explanation of these gradient kernels can be found in Appendix C. As an example, Figure 4.2 shows the kernels used to calculate the gradient from a dataset with dimensions of its voxels equal to \( 1 \text{ mm} \times 1 \text{ mm} \times 1.2 \text{ mm} \).
Using the information of the gradient magnitude, the algorithm slides the voxel values towards the lowest gradient voxel. In contrary with the 2D implementation, in 3D implementation, instead of considering only eight neighbours, we need to consider 26 neighbours for this gradient flow.

The voxels can then be grouped together based on this sliding route. All voxels in the group are assigned the same intensity value, which is the intensity of the smoothed dataset at the position where the gradient is the lowest in the group. As a result, this eliminates the variation of intensity around the ramp edges, and produces step edges.

4.2 Measurements

We shall use several measurements in order to tune the parameters to their optimal values. We introduce the measure of the bias field inhomogeneity ($\alpha$), and the measurement of the strength of the edges ($\beta$). The measure of $\beta$ is explained in section 4.2.1 and the measure of $\alpha$ is explained in section 4.2.2. Section 4.2.3 explains the criterion that we used to find the optimal kernel size for smoothing.

4.2.1 Measure of the strength of edges, $\beta$

We use the liver mask to represent the areas of the liver that tend to be uniform in the corrected output dataset, and we use the edge mask to represent the areas that contains the edges around the liver. Both the liver mask and the edge mask are already defined in section 3.1.1 and section 3.1.2.

In order to calculate $\beta$, the average gradient magnitude inside the liver mask, $\delta_L$, and the average gradient magnitude inside the edge mask, $\delta_E$, are needed. This gradient magnitude is calculated by using the kernels described in section 4.1.3.

Then, the significance of the liver edges, $\beta$, is given by:

$$\beta = \delta_E - \delta_L - (\delta_{E_o} - \delta_{L_o})$$

(4.3)

where $\delta_{E_o}$ and $\delta_{L_o}$ represent the average gradient magnitude values inside the edge mask.
and liver mask, respectively, of the input dataset before the process. It is expected as the result of edge enhancement, the value of $\beta$ should be positive.

### 4.2.2 Measure of the bias field inhomogeneity, $\alpha$

In order to calculate how much the inhomogeneity of the field was reduced in the output test dataset, we assume that the inhomogeneity direction can be represented by a vector, $\vec{v}$, that consists of three components, which are $\delta v_x$, $\delta v_y$, and $\delta v_z$. These are the components of the gradient vector direction of the field inhomogeneity.

To calculate $\delta v_x$, we divide equally the body mask volume into two subsections along the $x$-axis, as shown in Figure 4.3(a). We calculate the mean intensity value in volume $A$, $\mu_A$, and the mean intensity value in volume $B$, $\mu_B$. The gradient component in the $x$ direction, $\delta v_x$, is equal to $\mu_A$ minus $\mu_B$. The calculation of $\delta v_y$ and $\delta v_z$ is done similarly, except the division of the volume is done along the $y$-axis and $z$-axis, respectively, as shown in Figure 4.3(b) and Figure 4.3(c).

![Figure 4.3](image)

**Figure 4.3:** An example showing the division of the body mask to calculate (a) $\delta v_x$, (b) $\delta v_y$, (c) $\delta v_z$, and (d) $\alpha$. 
We then estimate two planes, which are normal to vector $\vec{i}$, to divide roughly the body mask volume into three equal subsections (see Appendix D). For each subsection in Figure 4.3(d), we find the average grey value (i.e. $\mu_1$, $\mu_2$ and $\mu_3$). Next, we define $m$ as the range of these local average values, i.e.

$$m = \max_{i=1,2,3} \mu_i - \min_{i=1,2,3} \mu_i \quad (4.4)$$

It is expected as a result of a good bias field correction, $m$ should be near to zero. Then, the measure of homogeneity, $\alpha$, is given as:

$$\alpha = \begin{cases} m_\alpha - m : \beta \geq 0 \\ 0 : \text{otherwise} \end{cases} \quad (4.5)$$

where $m_\alpha$ is the value of $m$ before the bias field correction, and $\beta$ is as defined in section 4.2.1. In this equation, we put the condition ($\beta \geq 0$) because we want the correction to at least retain the strength of the liver edges of the input image. A good bias field correction technique should give high positive $\alpha$ value.

### 4.2.3 Criterion used to find the optimal kernel size for smoothing

The optimal kernel size is found by finding the optimally smoothed output. The optimally smoothed output dataset can be selected according to its average contrast [40]. We convolve the input dataset with median kernels of several sizes to produce several versions of the smoothed dataset.

To calculate the average contrast, 15000 pairs of voxels are selected randomly from the smoothed dataset. If the smoothed dataset was produced by using a smoothing kernel with size $(2N+1) \times (2N+1) \times (2N+1)$, the voxels that represent the pairs should be chosen $0.65907N$ positions apart from each other. The contrast can be represented by the summation of the absolute intensity difference between the voxels in the pairs. The most optimally smoothed dataset can be considered to be that which has the first local minimum average contrast value, as shown previously in Figure 3.6.
4.3 Results and Discussion

4.3.1 The training dataset

In this experiment, a 3D MRI dataset of the abdominal region is used as the training dataset. This dataset is a gradient echo dataset, with parameter TR = 6.8 ms, TE = 2.3 ms, and flip angle = 25°. The dimensions of each voxel are 1.0 mm × 1.0 mm × 1.2 mm. There are one hundred and thirty 256 × 256 axial slices in this dataset. Some selected slices from this dataset are shown in Figure 4.4.

![Figure 4.4: Some selected axial slices from the training dataset used in this experiment.](image)

4.3.2 3D local enhancement

In the implementation of 3D local enhancement, two parameters need to be tuned, namely the size of the kernel and $k$ (see equation 4.1). From preliminary experiment in section 3.3.4, we found that these two parameters have an equal significance. Thus, in this work, we first optimise the choice of the kernel size by keeping parameter $k$ constant (i.e. $k = 10$). The optimal parameter setting is the one that maximise the value of $\alpha$, and this is identified by using the hill climbing method. The process of finding the optimal parameter is shown in Figure 4.5.
Chapter 4. 3D preprocessing of MRI dataset

Figure 4.5: (a) First, we optimise the choice of the kernel size by keeping parameter $k$ constant (i.e. $k = 10$). From this, we found maximum $\alpha$ when the size of kernel equal to $27 \times 27 \times 27$. (b) Then, by keeping the size of the kernel equal to $27 \times 27 \times 27$, we vary the value of $k$. The graph shows that the value of $\alpha$ increases when $k$ increases. Thus, we conclude that a size of $27 \times 27 \times 27$ and $k = 10000$ are adequate for representing the good parameter setting for this order set.

We found that the kernel size of $27 \times 27 \times 27$ and $k = 10000$ are adequate to represent the optimal parameter setting of the local enhancement algorithm. Figure 4.6 shows some of the slices in this processed dataset.

Figure 4.6: Some axial slices to represent the output of 3D local enhancement with kernel size of $27 \times 27 \times 27$ and $k = 10000$. 
4.3 Results and Discussion

4.3.3 3D median filtering

We convolve the optimal output from section 4.3.2 (i.e. the output of 3D local enhancement with size $27 \times 27 \times 27$ and parameter $k = 10000$) with cubic median kernels, with odd dimensions, ranging from size $3 \times 3 \times 3$ to $9 \times 9 \times 9$. Table 4.1 shows the contrast measurement of these output datasets. By using the technique described in section 4.2.3, we find that the median filtering with size $5 \times 5 \times 5$ produces the optimal smoothing result. Some of the slices in this optimal output dataset are shown in Figure 4.7.

<table>
<thead>
<tr>
<th>Size of median filter</th>
<th>$3 \times 3 \times 3$</th>
<th>$5 \times 5 \times 5$</th>
<th>$7 \times 7 \times 7$</th>
<th>$9 \times 9 \times 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contrast</td>
<td>2.99</td>
<td>1.94</td>
<td>2.90</td>
<td>3.51</td>
</tr>
</tbody>
</table>

Table 4.1: The contrast measurement of smoothed datasets.

Figure 4.7: The axial slices that are in Figure 4.6, after a 3D median filtering with size $5 \times 5 \times 5$.

4.3.4 3D toboggan contrast enhancement

We use the optimal median filtered result from section 4.3.3 (i.e. the output of median filtering with size $5 \times 5 \times 5$) as the input of 3D toboggan contrast enhancement. The toboggan contrast enhanced dataset has the value of $\beta$ equal to 1.18, which indicates that this technique successfully re-enhances the liver edges from the smoothed dataset. Figure 4.8 shows the corresponding result.
4.3.5 Discussion

In section 4.3.3, we found that 3D local enhancement with kernel size equal to $27 \times 27 \times 27$ and $k$ equal to $10000$, produces the output with $\alpha$ greater than zero. This indicates that the brightness in the dataset has been distributed evenly, thus the bias field inhomogeneity effect has been reduced. From section 4.3.3, we conclude that median filtering with the kernel size of $5 \times 5 \times 5$ is the optimal smoothing filter for the training dataset used in this experiment. Comparing Figure 4.6 with Figure 4.7, we can see that 3D median filtering removes the variations of the intensities within the organs. However, the images in the dataset become blurred and the edges are now fuzzy. The edges are then re-enhanced by 3D toboggan contrast enhancement. The output of toboggan contrast enhancement process, as shown in Figure 4.8, has the value of $\beta$ greater than zero, indicates that the strength of the liver edges has been improved. Thus, from sections 4.3.2, 4.3.3 and 4.3.4, we demonstrated that 3D preprocessing, similar to its 2D counterpart, enhances the quality of our dataset.

To compare the performance of 3D preprocessing with the corresponding 2D technique, we preprocess the same training dataset with the method described in Chapter 3 (i.e. the 2D preprocessing that consists of 2D local enhancement with kernel size equal to $13 \times 13$ and $k = 10000$, followed by 2D median filtering with kernel size $5 \times 5$ and 2D toboggan contrast enhancement). Some selected slices from this 2D preprocessed
4.3. Results and Discussion

Dataset, corresponding to the slices shown in Figure 4.1, are shown in Figure 4.9.

![Figure 4.9: 2D preprocessed version of images shown in Figure 4.4.](image)

In order to see the significance of the preprocessing towards segmentation, we use a user steered segmentation technique known as intelligent scissors (see section 2.1.12) to segment the liver from three datasets, namely, the training dataset itself, the 2D preprocessed dataset, and the 3D preprocessed dataset. For this implementation of intelligent scissors, we assigned $\omega_Z$, $\omega_C$ and $\omega_D$ of equation (2.3) equal to 0.43, 0.43, and 0.14, respectively, as suggested by Mortensen and Barrett [109, 110] work in a wide range of images.

Figure 4.10 shows the plot of time and the seed point required to segment each slice in these three datasets using intelligent scissors. As what we can see from this figure, most of the slices in the training dataset, which is not been applied with any preprocessing, require more segmentation time and seed points. This is because the liver in this dataset cannot be perceived easily by the human operator due to the bias field inhomogeneity. Thus, the human operator need to spend more time to place the seed points on the image. As the liver edges are weak, more seed points are needed in order to avoid the edge segments from attracted to the wrong edges. Preprocessing enhanced the dataset, thus the liver can be perceived easily in 2D and 3D preprocessed dataset. However, the 2D preprocessed dataset, as shown in Figure 4.9, is more sensitive to the small details compared to 3D preprocessed dataset (Figure 4.8). Thus, as these small details attract the edge segment, only short edge segments can be created, and this means more seed
Chapter 4. 3D preprocessing of MRI dataset

points and processing time are needed. Longer edge segments can be obtained in the 3D preprocessed dataset. Thus, 3D preprocessed dataset requires less seed points and preprocessing time.

![Seed Points vs. Slice Number](image1)

![Segmenation Time vs. Slice Number](image2)

Figure 4.10: (a) The seed points, and (b) the segmentation time needed to segment the liver from each slice using intelligent scissors.

In average, each slice in the training dataset requires 7 seed points and 33 seconds of segmentation time, the 2D preprocessed slice requires 6 seed points and 23 seconds segmentation time, and 3D preprocessed slice requires 5 seed points and 16 seconds
4.4 Conclusions

segmentation time. Because 3D preprocessed dataset requires the least seed points and segmentation time, we conclude that 3D preprocessing is more suitable to be used for an automatic segmentation method. Thus, we decided to use 3D preprocessing in our work.

Yet, 3D preprocessing requires more computational time compared to its 2D counterpart, as shown in Figure 4.11. For example, local enhancement with the kernel size of $33 \times 33 \times 33$ requires more than 2 hours to be completed, but, 2D local enhancement with kernel size $33 \times 33$ only requires less than 10 minutes to process all the slices in the dataset completely. However, this processing time is not of great concern to us, as this preprocessing method is fully automatic, and only requires a little human intervention to load the input data into the system.

![Processing time vs. Size of kernel](image)

**Figure 4.11:** The computational time associated with using different kernel sizes of local enhancement technique. This experiment is using a Pentium III computer, with 256 megabytes of physical memory.

4.4 Conclusions

In this chapter, we demonstrated that 3D preprocessing chain that consists of 3D local enhancement, followed by 3D median filtering, and 3D toboggan contrast enhancement, improves the quality of the abdominal MRI dataset. 3D local enhancement with kernel size $27 \times 27 \times 27$ and $k = 10000$ reduces the bias field inhomogeneity in the dataset. Median filtering with size $5 \times 5 \times 5$ lower the noise level, and 3D toboggan contrast enhancement re-enhance the liver edges from smoothed dataset.
We also demonstrated that 3D preprocessing is better than 2D preprocessing in terms of improving the performance of the segmentation using intelligent scissors. 3D preprocessed dataset requires less segmentation time, and less seed points, as this processing less sensitive to small detail. Thus, we decided to use 3D preprocessing for the experiments that follow.

However, the liver edges in the preprocessed dataset are still not strong enough to attract the edge segments of intelligent scissors for an automatic segmentation. Thus, we try to salient the edge map of the liver using the primary visual cortex model, proposed by Li [92, 93]. This work will be carried out in Chapter 5.
Chapter 5

Salienation of the liver edges

Even after preprocessing, the edge map produced from every image slice containing the liver is not good enough for the automatic identification of the outline of the liver. To identify the liver outline, we decided to use a method that salienates features in the edge map which have some perceptual meaning. Salienation process weights the edges depending on their length, curvature, and the state of their closeness. To achieve that we implemented a model proposed recently for the function of the human brain that is responsible for feature salienation.

This chapter is divided into four sections. Section 5.1 gives an overview of the primary visual cortex model proposed by Li [92, 93]. Section 5.2 presents how we implemented this model. Section 5.3 shows the results, and section 5.4 presents the conclusions.

Even after this salienation process, the liver outline cannot be perfectly extracted because the produced edge is fragmented due to variable contrast along its periphery. So, we take the salienation map as one of the cost features used in our implementation of intelligent scissors. We also use salienation to identify the relevant fragments of the outline of the liver, and then use them to identify from them seed points which will be used as input to the intelligent scissors algorithm which finds the complete contour. These will be described in Chapter 6.
Chapter 5. Salienation of the liver edges

5.1 Introduction to the Primary Visual Cortex (V1) Model

Figure 5.1 shows the visual cortices in the human brain. Physiologists assume that before the optical input is transferred to the higher visual areas inside the brain, the process of contour grouping is first taking place in the primary visual cortex (V1) [87]. V1 region is a complex network, which is build up from several layers of different type of cell, that receive the signal, process the information, and send it to the higher level of the visual system [143].

The ability of V1 to enhance the edges has attracted the attention of many researchers to study its mechanism, because such findings may lead to better segmentation techniques [87][92][93][94][165]. V1 produces an output, which highlights the edges depending on their length, curvature [75], and the state of their closeness [80]. V1 also reduces the significance of the unwanted edges that are excited by the background and noise [58]. Li [92][93][94], based on experimental findings, introduced a contour enhancement model which is using only V1 elements.

According to the model proposed by Li [92][93][94], V1 is composed of excitatory and
inhibitory neurons$^1$, where the inhibitory cells are considered as just the interneurons of the excitatory cells. This is based on the assumption that most of the direct input signal is received by the excitatory cells and the ratio between these two types of neuron is one to one.

In this model, V1 consists of many hypercolumns that are arranged in a discrete spatial location (see Figure 5.2(a)). These hypercolumns only respond inside a specified receptive field (RF). In each hypercolumn, there are $K$ pairs of neurons. Each of these neuron pairs has a preference in orientation $\theta$ as shown in Figure 5.2(b). This means that each neuron pair in this hypercolumn might react differently to the input from an edge directed about $\phi$.

![Figure 5.2: Model of hypercolumns, and edge segments. In this figure, we assume that each hypercolumn is made up from twelve neuron pairs, i.e. $K = 12$. (a) The hypercolumns are arranged in discrete spatial locations. (b) Each of edge segments in a hypercolumn has its own $\theta$ orientation. (Based on [93].)]](image)

Li considers that these neuron pairs interact with each other, thus the response of a cell is also depending on its surrounding neighbours. The excitatory cell with preferred orientation $\theta$ (i.e. $\theta = k\pi/K$ for $k = 1, 2, \ldots, K$) inside an RF area centred at $i$, has a membrane potential $V_{ie}(t)$ at time $t$, while its corresponding inhibitory interneuron has a membrane potential $V_{ie}(t)$. The changes of $V_{ie}$ and $V_{ie}$ over time are given by:

$^1$Excitatory and inhibitory neurons in V1 process the input by improving or reducing the strength of the signal, respectively [64].
Chapter 5. Salienation of the liver edges

\[
\begin{align*}
\dot{X}_{i\theta}(t) & = \frac{X_{i\theta}(t) - X_{i\theta}(t - \Delta t)}{\Delta t} \\
& = -\alpha_x X_{i\theta}(t - \Delta t) - \sum_{\Delta \theta} \psi(\Delta \theta) g_Y(Y_{i\theta + \Delta \theta}(t - \Delta t)) + J_o g_X(X_{i\theta}(t - \Delta t)) \\
& \quad + \sum_{j \neq i, \theta'} J_{i\theta, j\theta'} g_X(X_{j\theta'}(t - \Delta t)) + I_i + I_o \\
\dot{Y}_{i\theta}(t) & = \frac{Y_{i\theta}(t) - Y_{i\theta}(t - \Delta t)}{\Delta t} \\
& = -\alpha_y Y_{i\theta}(t - \Delta t) + g_X(X_{i\theta}(t - \Delta t)) + \sum_{j \neq i, \theta'} W_{i\theta, j\theta'} g_X(X_{j\theta'}(t - \Delta t)) + I_c
\end{align*}
\]

where

- \(\alpha_x\): excitatory membrane time constant.
- \(\alpha_y\): inhibitory membrane time constant.
- \(\psi(\Delta \theta)\): inhibitory function within a hypercolumn.
- \(g_X(X_{i\theta})\): excitatory output function of membrane potential \(X_{i\theta}\).
- \(g_Y(Y_{i\theta})\): inhibitory output function of membrane potential \(Y_{i\theta}\).
- \(I_i\): response of the neuron to its direct input.
- \(I_o\): background input to the excitatory cells.
- \(I_c\): background input to the inhibitory cells.
- \(J_o\): self excitation strength of the neuron.
- \(J_{i\theta, j\theta'} g_X(X_{j\theta'})\): excitation signal received from the neighbouring neurons.
- \(W_{i\theta, j\theta'} g_X(X_{j\theta'})\): inhibition signal received from the neighbouring neurons.

For the implementations of equation (5.1) and (5.2), we follow the parameter settings used by Li in her papers [92][93].

As the input of the system is the edges with magnitude \(\hat{I}_{i\phi}\) and direction \(\phi\) (which is equal to the gradient direction rotated 90° clockwise), we estimate the gradient vector along the \(x\)-axis, \(\hat{\delta}_x\), and the gradient vector along the \(y\)-axis, \(\hat{\delta}_y\), by convolving the given input image with the masks shown in Figure 5.3. Thus,

\[
\hat{I}_{i\phi} = \sqrt{\hat{\delta}_y^2 + \hat{\delta}_x^2}
\]

\[
\phi = \arctan\left(\frac{-\hat{\delta}_x}{\hat{\delta}_y}\right)
\]
5.1. Introduction to the Primary Visual Cortex (V1) Model

Figure 5.3: Kernels used in this experiment to calculate the gradient. (a) Mask used to compute the gradient component in the \( x \) direction, \( \delta_x \). (b) Mask used to compute the gradient component in the \( y \) direction, \( \delta_y \).

Based on [92], we set the value of \( K \) equal to 12, which means that at each hypercolumn, there are 12 neuron pairs with preferred angle sensitivities 15° apart from each other (i.e. \( \theta = 15^\circ, 30^\circ, \ldots, 165^\circ, 180^\circ \)).

The response of neuron, \( I_{\theta\phi} \), to the direct input, \( I_{\phi} \), is

\[
I_{\theta\phi} = I_{\phi} \exp \left( -\frac{22.5°}{|\theta - \phi|} \right) \quad (5.5)
\]

where \(|\theta - \phi| \leq \pi/2\). This equation means that \( I_{\theta\phi} \) gives a good response when the value of \( I_{\phi} \) is high, or when the input is almost parallel with the direction preferred by the neuron pair (i.e. \( \phi \) is almost equal to \( \theta \)).

The non-linear weighting function \( \psi(\theta) \) used in equation (5.1) is given by

\[
\psi(\theta) = \begin{cases} 
1 & \text{when } \theta = 0 \\
0.8 & \text{when } |\theta| = \pi/K = 15^\circ \\
0.7 & \text{when } |\theta| = 2\pi/K = 30^\circ \\
0 & \text{otherwise} 
\end{cases} 
\quad (5.6)
\]

The background input to the excitatory cell, \( I_o \), is given by

\[
I_o = 0.85 - 2.0 \left[ \sum_{\beta \in S_i} \sum_{\theta'} g(x(x_{\beta\theta'}))^2 \right] / N_i 
\quad (5.7)
\]

In this equation, \( S_i \) is the area containing the hypercolumns that are less than or equal to two grid distances from hypercolumn \( i \), as shown in Figure 5.4. \( N_i \) represents the number of pixels in \( S_i \).
Figure 5.4: Area defined by $S_i$. The hypercolumns in the marked area are less than or equal to two grid distances from hypercolumn $i$.

The output of the excitation membrane potential $\mathcal{X}$, $g_\mathcal{X}(\mathcal{X})$, and the output of the inhibition membrane potential $\mathcal{Y}$, $g_\mathcal{Y}(\mathcal{Y})$, are modelled by

$$g_\mathcal{X}(\mathcal{X}) = \begin{cases} 0 & \text{if } \mathcal{X} < 1 \\ (\mathcal{X} - 1) & \text{if } 1 \leq \mathcal{X} \leq 2 \\ 1 & \text{elsewhere} \end{cases}$$

(5.8)

$$g_\mathcal{Y}(\mathcal{Y}) = \begin{cases} 0 & \text{if } \mathcal{Y} < 0 \\ 0.21\mathcal{Y} & \text{if } 0 \leq \mathcal{Y} \leq 1.2 \\ 2.5\mathcal{Y} - 2.748 & \text{elsewhere} \end{cases}$$

(5.9)

Consider that there is a line connecting hypercolumn $i$ and hypercolumn $j$. This line has an angle $\theta_{ij}$ relative to the x-coordinate axis, and has a length $d_{ij}$. An example is shown in Figure 5.5.

Next, consider a line segment $i\theta$ in hypercolumn $i$, and a line segment $j\theta'$ in hypercolumn $j$. We find the smallest angles of these line segments with the line connecting these hypercolumns, denoted by angle $\Phi_i$ and angle $\Phi_j$, respectively, where $|\Phi_i| \leq \frac{\pi}{2}$.
5.1. Introduction to the Primary Visual Cortex (V1) Model

Hypercolumn

A grid unit = arc (5.2361 grid unit)

Figure 5.5: An example to show a line connecting hypercolumn $i$ with hypercolumn $j$. This line has a length of $d_{ij}$, and has an angle $\theta_{ij}$ relative to the $x$-coordinate axis.

and $|\Phi_j| \leq \frac{\pi}{2}$, i.e.

$$\Phi_i = \arctan \left( \frac{\tan \theta_{ij} - \tan \theta_i}{1.0 + \tan \theta_{ij} \tan \theta_i} \right)$$

(5.10)

$$\Phi_j = \arctan \left( \frac{\tan \theta_{ij} - \tan \theta_j}{1.0 + \tan \theta_{ij} \tan \theta_j} \right)$$

(5.11)

Then,

$$\theta_1 = \begin{cases} \Phi_i & \text{if } |\Phi_i| \leq |\Phi_j| \\ \Phi_j & \text{otherwise} \end{cases}$$

(5.12)

and

$$\theta_2 = \begin{cases} \Phi_j & \text{if } |\Phi_i| \leq |\Phi_j| \\ \Phi_i & \text{otherwise} \end{cases}$$

(5.13)

Based on equation (5.12) and equation (5.13), we define parameter $\gamma$ as follows

$$\gamma = 2|\theta_1| + 2\sin(|\theta_1 + \theta_2|)$$

(5.14)

The influence of edge element $j\theta'$ to edge element $i\theta$ (i.e. via $J_{i\theta,j\theta'}$ and $W_{i\theta,j\theta'}$) are defined by

$$J_{i\theta,j\theta'} = \begin{cases} 0.126e^{-((\pi/d_{ij})^2 - 9(\gamma/d_{ij})^2 - d_{ij}^2)/900} & \text{if } (0 < d_{ij} \leq 10.0 \text{ and } \gamma < \pi/2.69) \\
& \text{or } (0 < d_{ij} \leq 10.0 \text{ and } \gamma < \pi/11) \\
& \text{and } |\theta_1| < \pi/5.9 \text{ and } |\theta_2| < \pi/5.9 \\ 0 & \text{otherwise} \end{cases}$$

(5.15)
Chapter 5. Salienation of the liver edges

\[ W_{i\theta, j\theta'} = \begin{cases} 
0 & \text{if } d_{ij} = 0 \text{ or } \gamma < \pi/11 \\
\text{or } |\Delta \theta| \geq \pi/3 \\
\text{or } |\theta_1| < \pi/11.999 \\
\text{or } d_{ij}/\cos(\gamma/4) \geq 10 \\
0.14(1 - e^{-0.4(\gamma/d_{ij})^{1.5}})e^{-((\Delta \theta/\pi)^{1.5})} & \text{otherwise}
\end{cases} \] (5.16)

where \( d_{ij} \) is the distance between neuron \( i\theta \) and neuron \( j\theta' \), \( \Delta \theta = \theta - \theta' \) (with \( |\theta - \theta'| \leq \pi/2 \)), while parameters \( \theta_1, \theta_2, \) and \( \gamma \) are taken from equation (5.12), equation (5.13), and equation (5.14), respectively.

The self-excitatory strength of the cell, \( J_o \), is set to 0.8, the background input to the inhibitory cell, \( J_c \), is set to 1.0, \( \alpha_x \) is set to 1.0, and \( \alpha_y \) is set to 1.0.

We set the value of \( \chi(-\Delta t) = 0 \), and the value of \( \gamma(-\Delta t) = 0 \), which means the membranes do not have any response at the beginning of the execution. At the end of every iteration \( t \), the values of \( X_{i\theta}(t) \) and \( Y_{i\theta}(t) \) are updated based on equation (5.1) and equation (5.2), which for the discrete case take the following form:

\[ \dot{X}_{i\theta}(t) = \frac{X_{i\theta}(t) - X_{i\theta}(t-\Delta t)}{\Delta t} \Rightarrow X_{i\theta}(t) = X_{i\theta}(t-\Delta t) + \dot{X}_{i\theta}(t)\Delta t \] (5.17)

\[ \dot{Y}_{i\theta}(t) = \frac{Y_{i\theta}(t) - Y_{i\theta}(t-\Delta t)}{\Delta t} \Rightarrow Y_{i\theta}(t) = Y_{i\theta}(t-\Delta t) + \dot{Y}_{i\theta}(t)\Delta t \] (5.18)

The output pixel \( Output(i,t) \), which represents the saliency of the edge at time \( t \), is represented by the maximum response of the excitation cells, \( g_x(X_{i\theta}(t)) \). This is given by

\[ Output(i,t) = \max_{\theta} g_x(X_{i\theta}(t)) \] (5.19)

Based on a direct conversation with Dr. Li herself, we decided to follow her suggestion and use a value of \( \Delta t \) less than 1.0. The result, which is the saliency of the edges, \( Output(i,t) \), is obtained after a few iterations of the system.
5.2 Implementation of the V1 Model

In order to synchronously update the value of $X_{\theta}(t)$ and $Y_{\theta}(t)$, for computational purposes, each input image needs $4 \times K$ intermediate image arrays, which correspond to every preferred orientation $\theta$ of the excitatory and inhibitory neurons (i.e. $X_{\theta}$ and $Y_{\theta}$), at time $(t)$, and also at time $(t - \Delta t)$. We refer to these arrays as $X_{\theta}(\text{now})$, $Y_{\theta}(\text{now})$, $X_{\theta}(\text{previous})$, and $Y_{\theta}(\text{previous})$. In this implementation, 48 intermediate images are needed, as $K$ has been set to 12.

The implementation of the V1 model is following these steps:

Step 1: Define $\Delta t$ and the required total response time $T$. Set current time $t$ equal to $-\Delta t$.

Step 2: Initialise as zero all pixels in $X_{\theta}(\text{now})$, $Y_{\theta}(\text{now})$, $X_{\theta}(\text{previous})$, and $Y_{\theta}(\text{previous})$.

Step 3: Convolve the input image with the kernel shown in Figure 5.3. Calculate the edge magnitude, $I_{\theta}$, and edge direction, $\phi$, using equation (5.3) and equation (5.4).

Step 4: Calculate the response of the neuron, $I_{\theta}$, using equation (5.5).

Step 5: Using equation (5.7), calculate the background input to the excitatory cell, $I_e$.

Step 6: Calculate $X_{\theta}(\text{now})$ using equation (5.1), with $\alpha_x = 1.0$ and $I_e = 0.8$.

Step 7: Calculate $Y_{\theta}(\text{now})$ using equation (5.2), with $\alpha_y = 1.0$ and $I_e = 1.0$.

Step 8: Calculate $X_{\theta}(\text{now})$ using equation (5.17), i.e. $X_{\theta}(\text{now}) = X_{\theta}(\text{previous}) + \Delta t \dot{X}_{\theta}(\text{now})$

Step 9: Calculate $Y_{\theta}(\text{now})$ using equation (5.18), i.e. $Y_{\theta}(\text{now}) = Y_{\theta}(\text{previous}) + \Delta t \dot{Y}_{\theta}(\text{now})$

Step 10: Update time, i.e. $t = t + \Delta t$.

Step 11: If $t > T$, go to Step 13. Else, go to Step 12.

Step 12: Set $X_{\theta}(\text{previous}) = X_{\theta}(\text{now})$, $Y_{\theta}(\text{previous}) = Y_{\theta}(\text{now})$, and go to Step 5

Step 13: Calculate output using equation (5.19).

5.3 Results and Discussion

Because the preprocessing stage in the saliennation algorithm it is not judged necessary to use large edge detection filters. However, in order to show the added value in using saliennation, we present here the outputs of a Canny edge detector using filters of size $7 \times 7$ and the result of the saliency algorithm applied after the filtering process.
Figure 5.6 shows some axial slices used in this experiment. This dataset has been preprocessed using the 3D preprocessing chain described in Chapter 4, in order to reduce the bias field inhomogeneity artifact, and to improve the strength of the liver edges in the dataset.

Figure 5.6: Some axial slices which represent the 3D input liver dataset used in this experiment.

Figure 5.7 shows the output of the Canny edge detector, while Figure 5.8 shows the output of the salienation algorithm. For both figures the same Canny filters of size $7 \times 7$ were used to estimate the gradient magnitude and orientation of the pixels. In both cases the outermost rim of the body part in Figure 5.6 was omitted when scaling the output values for presentation, so the internal details of each image may be enhanced without the domination of the strong outer edge. We can clearly see from these results the important role of the salienation algorithm in enhancing the long edges of interest.

Figure 5.7: The output of a Canny edge detector using filter of size $7 \times 7$. (The intensity of each image has been inverted and scaled for displaying purposes.)
5.4 Conclusions

In this chapter, it was shown that the primary visual cortex model (i.e. the V1 model) proposed by Li [92, 93] can be used to identify salient features in an image. The edgels in the output saliency map are assigned values between 0 to 1, based on their length, curvature, and good continuity. The usage of the produced saliency map towards an automatic segmentation of the liver surface will be described in Chapter 6.

Figure 5.8: The corresponding outputs of the saliency algorithm after two iterations. In this case, \( \Delta t \) is set to 0.075. (The intensity of each image has been inverted for displaying purposes.)
Chapter 6

Automatic intelligent scissors for MRI liver segmentation

In this chapter, we describe a method to execute automatically intelligent scissors, which is a hitherto interactive segmentation technique (see section 2.1.12), for segmenting the liver from a preprocessed 3D MRI dataset. The preprocessing technique used is a 3D preprocessing, such as given in Chapter 4. One of the cost functions used in this implementation of intelligent scissors is the saliency map, produced from the visual primary cortex model described in Chapter 5. The technique segments the liver on a slice-by-slice basis (which is in 2D space), but using information from the previously segmented slice (which is in 3D space). So, we consider this technique as a 2.5D segmentation technique.

This chapter is divided into several sections. Section 6.1 describes the novelty of our technique compared to the original intelligent scissors. Each step in the technique will be described further in section 6.2 to section 6.5. Results and discussion are given in section 6.6. Section 6.7 presents the conclusion.

6.1 Methodology

The idea of a user-steered semi-automatic segmentation, which is known as intelligent scissors, was originally proposed by two independent research groups, namely, Falcao
et al [42], and Mortensen et al [109][110]. Here we prefer to apply the implementation by Mortensen et al, because it is easier to understand, and less heuristic.

Intelligent scissors can be considered as a combination of edge-based and dynamic programming segmentation. This technique requires the human operator to determine the starting and the termination points for the edge segments, interactively. The resultant edge segment, which connects two defined points, is created based on the lowest cost path according to Dijkstra's dynamic programming algorithm [109][110].

Intelligent scissors can reduce the time that is required to segment the liver in 2D slices compared with manual segmentation. However, in order to segment the liver entirely from a big 3D dataset, intelligent scissors is still considered a tedious technique.

In contrast to the original method, our intelligent scissors automatically identify the seed points in each slice. Thus, segmentation using intelligent scissors become automatic. The main idea of our technique is to propagate a contour from an initial axial slice to the entire slices in 3D dataset. The method is following these stages:

- **STAGE 1**: Selection of an axial slice as the initial slice.
- **STAGE 2**: Segmentation of the initial slice.
- **STAGE 3**: Segmentation of the remaining axial slices in the dataset.
- **STAGE 4**: Refining the output by combining segmentation results from orthogonal direction.

All stages, except the first stage that requires minimal human intervention, are entirely automatic. The first stage is described in section 6.2. Stage two is described in section 6.3. Stage three is described in section 6.4, and stage four is described in section 6.5.

### 6.2 Stage 1: Selection of the Initial Slice

This stage is the only stage which requires human intervention. The human operator needs to select an axial slice from the dataset as the initial slice. The selection of the initial slice is based on visual inspection. Two criteria for this selection are:

1. The liver has a good contrast with its surroundings.
2. The shape of the liver preferably resembles a circle or an ellipse.
These criteria are important to be followed in order to create the correct seed points on the initial slice. The process of automatically generating the seed points on the initial slice is described in section 6.3.

6.3 Stage 2: Segmentation of the Initial Slice

In order to generate the seed points automatically on the initial slice (that has been selected from section 6.2), the corresponding saliency map to that slice is used (please refer to Chapter 5 for salienation method). The process of identifying the seed points consists of three steps, namely:

1. Thinning of the saliency map.
2. Estimation of the centre of the curve we wish to identify.
3. Identification of the seed points along the curve.

6.3.1 Thinning of the saliency map

We apply simple thresholding to the saliency map, as follows:

\[ h(x, y) = \begin{cases} 
0 & : \text{Output}(x, y) \geq T \\
255 & : \text{elsewhere} 
\end{cases} \]  

(6.1)

The result of this thresholding assigns to the edges value 0, and to the background pixels value 255. We select \( T = 1.0 \), which means that we consider a pixel as the edge only if any membrane potential at this position, \( g_x(x') \), has the maximum response, i.e. \( g_x(x') = 1.0 \).

In order to make the analysis easier, we skeletonize the edges, and make them one pixel wide. To do this, we use mathematical morphology and iteratively erode the edges using the structuring elements shown in Figure 6.1, consequently, until there is no changes in the output compared with its previous one [44].

After the thinning process, we delete any short edges. The length of short edges is dependent on image size. For our image that is in size of \( 256 \times 256 \), we delete any
edges which are less than 25 pixels in length. This can be done by using a connected
component analysis.

\[
\begin{array}{cccccccc}
1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 6.1: Structuring elements used to skeletonize the edges. Pixels with intensity 0 represent the
edgels, while pixels with intensity 1 represent the background. The pixel in grey is the origin of the
structuring element. (Based on [44].)

6.3.2 Estimation of the centre of the curve

We select randomly a triplet of edgels from the edges detected in section 6.3.1. Let us
assume that these triplet of points has coordinates \((x_1, y_1), (x_2, y_2)\) and \((x_3, y_3)\). The
centre of the circle they define has coordinates \((x_c, y_c)\), as shown in Figure 6.2.

\[
\begin{align*}
(x_2, y_2) & \quad (x_3, y_3) \\
(x_1, y_1) & \quad (x_c, y_c)
\end{align*}
\]

\[l_1, l_2\]

Figure 6.2: A triplet of edgels \((x_1, y_1), (x_2, y_2),\) and \((x_3, y_3)\), define a circle with centre at \((x_c, y_c)\).

From this figure, line \(l_1\) is perpendicular to the line that connects point \((x_1, y_1)\) with
point \((x_2, y_2)\). Thus, the slope of \(l_1, m_1\), can be derived as:

\[
m_1 = \frac{y_c - \frac{(y_1 + y_2)}{2}}{x_c - \frac{(x_1 + x_2)}{2}} = \frac{x_2 - x_1}{y_2 - y_1}
\]

Similarly, the slope of \(l_2, m_2\), which is perpendicular to the line connecting point \((x_2, y_2)\)
and point \((x_3, y_3)\), is given by:

\[
m_2 = \frac{y_c - \frac{(y_2 + y_3)}{2}}{x_c - \frac{(x_2 + x_3)}{2}} = \frac{x_3 - x_2}{y_3 - y_2}
\]
6.3. Stage 2: Segmentation of the Initial Slice

From equation (6.2),

\[ y_c = \left( x_c - \frac{x_1 + x_2}{2} \right) \left( \frac{x_1 - x_2}{y_2 - y_1} \right) + \frac{y_1 + y_2}{2} \]  \hspace{1cm} (6.4)

In the same way, from equation (6.3),

\[ y_c = \left( x_c - \frac{x_2 + x_3}{2} \right) \left( \frac{x_2 - x_3}{y_3 - y_2} \right) + \frac{y_2 + y_3}{2} \]  \hspace{1cm} (6.5)

The left-hand-sides of equations (6.4) and (6.5) are equal, thus,

\[ \left( x_c - \frac{x_1 + x_2}{2} \right) \left( \frac{x_1 - x_2}{y_2 - y_1} \right) + \frac{y_1 + y_2}{2} = \left( x_c - \frac{x_2 + x_3}{2} \right) \left( \frac{x_2 - x_3}{y_3 - y_2} \right) + \frac{y_2 + y_3}{2} \]

\[ x_c = \frac{(y_3 - y_1)(y_3 - y_2)(y_2 - y_1) + (x_2^2 - x_3^2)(y_2 - y_1) - (x_2^2 - x_1^2)(y_3 - y_2)}{2[(x_1 - x_2)(y_3 - y_2) - (x_2 - x_3)(y_2 - y_1)]} \] \hspace{1cm} (6.6)

Substituting equation (6.6) into equation (6.4), we get,

\[ y_c = \frac{(x_3 - x_1)(x_3 - x_2)(x_2 - x_1) + (y_3^2 - y_2^2)(x_2 - x_1) - (y_3^2 - y_1^2)(x_3 - x_2)}{2[(y_1 - y_2)(x_3 - x_2) - (y_2 - y_3)(x_2 - x_1)]} \] \hspace{1cm} (6.7)

For equations (6.6) and (6.7) to be valid, the denominator must not be 0, i.e. \((x_1 - x_2)(y_3 - y_2)\) must not be equal to \((x_2 - x_3)(y_2 - y_1)\), which means that the triplet of points must not form a straight line.

In order to estimate the best centre from the fragments of the contour of the liver, we use \(3N\) unique triplets, where \(N\) is the number of the edgels in the image, and create a 2D accumulator array, which represents a 2D histogram of the \((x_c, y_c)\) values computed from each triplet. From this accumulator array, we select the position of the peak of the histogram. This position represents the ideal centre of the edgel fragments we have.

6.3.3 Identification of the seed points

Using the edgels from section 6.3.1, we project straight lines every 15° emanating from the centre that was detected in section 6.3.2. We identify as seed points the first intersection points these lines have with the edge fragments, as shown in Figure 6.3. Intelligent scissors are then used to find the edges between two seed points in sequence.
Chapter 6. Automatic intelligent scissors for MRI liver segmentation

6.4 Stage 3: Propagation of the Contour

The contour of the initial slice, produced from section 6.3, provides the initial information for selecting appropriate seed points in the subsequent slice. Section 6.4.1 describes the technique that we use to estimate the seed points in the slice other than the starting slice. Section 6.4.2 introduces the property of point snap. Section 6.4.3 explains how we can restrict the computation to be local.

6.4.1 Selection of the seed points

To simplify the method, we skeletonize the detected contour of the previously processed slice, in order to make sure that the edgels are connected based on the eight
neighbourhood scheme. In this scheme, we expect that each contour edgel connects to no more than two neighbouring edgels. We then select the right most, and the top most edgel as the starting point of the link. From this point, we label the edgels with increasing numbers. An example is shown in Figure 6.4. From this labelled contour, two techniques of selecting the seed points may be used. They are (1) based on the contour curvature, and (2) based on the contour length.

Selection based on the edge curvature

We may select the seed points based on the inflection points of the contour. In order to find the inflection points, we need to calculate the angle of every edgel of the curve. An angle on a digital curve may be calculated using a technique known as $k$-cosine, proposed by Rosenfeld and Johnston [139]. The algorithm is given below.

Step 1 Define $m = n/10$, where $n$ is the number of points in the contour.

Step 2 For each point $i$ (until $i = n$)

1. for $(k = 1; k \leq m; k = k + 1)$
   - Calculate vector $\vec{a}_{ik}$ that connects point $i$ and point $(i+k)$, i.e. $\vec{a}_{ik} = (x_i - x_{i+k}, y_i - y_{i+k})$.
   - Calculate vector $\vec{b}_{ik}$ that connects point $i$ and point $(i-k)$, i.e. $\vec{b}_{ik} = (x_i - x_{i-k}, y_i - y_{i-k})$.
   - Calculate the cosine value, $c_{ik}$, of the angle defined by vector $\vec{a}_{ik}$ and $\vec{b}_{ik}$, i.e. $c_{ik} = \cos(\theta_{a_{ik}} - \theta_{b_{ik}}) = (\vec{a}_{ik} \cdot \vec{b}_{ik})/(|\vec{a}_{ik}||\vec{b}_{ik}|)$

2. The optimal cosine value at point $i$, $c_i$, is defined as the last local maximum of $c_{ik}$ from $k = 1$ to $k = m$. The optimal gap size is given by $h$. To find $c_i$ and $h$, we set $c_{10} = -1.0$, and define parameter $F = 0$.

3. for $(k = m; k \geq 1 \text{ OR } F = 1; k = k - 1)$
   - if $c_{ik} > c_{ik-1}$, set $c_i = c_{ik}$, $h = k$, $F = 1$.

Step 3 Find the inflection points, which indicated by curvature minima (i.e. $c_i \leq c_j$ for $|i - j| < h/2$).

As an example, we use the contour shown in Figure 6.4(b). This contour consists of 20 edgels, thus $n = 20$, and $m = 20/10 = 2$. Table 6.1 shows the calculation of $c_{1k}$, (which is the values of cosine at point 1), in order to find $c_1$. From this table, the last local maximum occurs when $k = 2$, thus, $h = 2$, and $c_1 = -0.4472$. We use the same
Chapter 6. Automatic intelligent scissors for MRI liver segmentation

technique to find the cosine values of the other points in the contour (i.e. \( c_2, c_3, \ldots, c_{20} \)).

| \( k \) | \( \delta_{1k} \) | \( \delta_{1k} \) | \( |\delta_{1k}| \) | \( |\delta_{1k}| \) | \( c_{1k} \) |
|---|---|---|---|---|---|
| 0 | - | - | - | - | -1.0000 |
| 1 | (-1,1) | (0,-1) | \( \sqrt{2} \) | 1 | \( (0 - 1)/\sqrt{2} \) = -0.7071 |
| 2 | (-2,1) | (0,-2) | \( \sqrt{5} \) | 2 | \( (0 - 2)/(2\sqrt{5}) \) = -0.8844 |

Table 6.1: Calculation of \( c_{1k} \) of the contour shown in Figure 6.4(b).

Table 6.2 shows the list of \( c_i \) and \( h \) for every contour point shown in Figure 6.4(b). From this table, it is shown that the inflection points are points 4, 7, 13, 17, and 19. Thus, there may be five segments in this example. First, intelligent scissors connect point 4 with point 7. Then, they connect point 7 and point 13. This is followed by connecting point 13 with point 17, point 17 with point 19, and finally, point 19 with point 4.

<table>
<thead>
<tr>
<th>( i )</th>
<th>( c_i )</th>
<th>( h )</th>
<th>Local minimum?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.4472</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>0.0000</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>-0.3162</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>-1.0000</td>
<td>1</td>
<td>YES</td>
</tr>
<tr>
<td>5</td>
<td>-0.7071</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>6</td>
<td>-0.7071</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>-0.8000</td>
<td>2</td>
<td>YES</td>
</tr>
<tr>
<td>8</td>
<td>-0.7071</td>
<td>1</td>
<td>no</td>
</tr>
<tr>
<td>9</td>
<td>-0.5000</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>10</td>
<td>-0.4472</td>
<td>2</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 6.2: List of \( c_i \) and \( h \), to identify the inflection points of the contour shown in Figure 6.4(b).

Selection based on the edge length

The simplest way to select the seed points is by dividing the contour into a number of segments. To simplify the explanation, let us take the contour shown in Figure 6.4(b) as our example.
This contour consists of 20 edgels. If we decide to divide the contour into five segments, we shall have 20 edgels/5 segments = 4 edgels per segment. Thus we select the edgels with labels 4, 8, 12, 16, and 20. For the segmentation, in this case, first the intelligent scissors connect point 4 with point 8, then point 8 with point 12, point 12 with point 16, point 16 with point 20, and finally, point 20 with point 4.

![Figure 6.5: The directions axial, sagittal, and coronal.](image)

We use this technique for segmenting the data in the coronal and sagittal directions (see Figure 6.5). This is because the liver areas defined in these directions are sometimes too narrow. The selection based on the inflection points produces only a few seed points, which usually in this case, are not sufficient to segment the liver correctly. We show an example in Figure 6.6.

![Figure 6.6: Segmenting the data in the coronal direction.](image)

(a) The reference contour and the generated seed points based on the inflection points of this contour. (b) The segmentation result using the seed points shown in (a). (c) The same reference contour as (a) but the seed points are generated by dividing the contour into 10 segments. (d) The segmentation result by using the seed points shown in (c). The segmentation result shown in (d) is preferable compared with that shown in (b).
6.4.2 Point snap

In intelligent scissors, the location of the seed points is crucial. The ideal seed points are the one that lie on the edge of the object. Thus, in the original implementation of intelligent scissors, the human operator needs to put the seed points in a very close proximity to the object edge, and this increases the human effort in the segmentation process [109].

Mortensen et al have introduced the property of cursor snap to simplify the usage of their application [110]. With this property, the user is not required to put the seed points directly on the object's boundary, thus the user effort is reduced. It is commonly known that the edges are associated with the high gradient magnitude. The gradient magnitude feature, $f_G$, of intelligent scissors is inversely proportional to the actual gradient magnitude value. Thus, when the user defines a seed at point $p$, the program checks for the minimum $f_G$ among the neighbouring pixels within windows of specific sizes (i.e. from size $1 \times 1$ to $19 \times 19$, depending on the requirements of the user). In this window, the position of the seed point moves from $p$ to the position where $f_G$ is minimum, where the pixels are more likely to be an edge of the object.

Figure 6.7: (a) The circle in this image indicates an initial seed point, that has been identified as described in section 6.4.1. The pixel value represents the gradient magnitude, $f_G$. The grey area represents a $7 \times 7$ window. (b) The point snap property moves the seed point to the point with the lowest gradient magnitude within this window.

Following the same idea, for each seed point identified in section 6.4.1, we create a $7 \times 7$ window, centred at this point. The point then snaps to the position of the minimum $f_G$ inside this window (see Figure 6.7 for an example). We select the size of $7 \times 7$ because we assume that the boundary of the liver is located near the boundaries of other organs.
Thus, selecting a bigger window may cause intelligent scissors to follow the wrong boundary. Selecting a smaller window did not produce any significant improvement of the segmentation.

6.4.3 Local computation

Schenk et al. [147] propose a local computation for segmenting the liver images from CT or MRI dataset. To improve the accuracy and save computation time in the intelligent scissors algorithm, the cost function is only calculated around the contour copied from the nearest adjacent slice. The relevant region is identified by using a Euclidean distance transform. Each output pixel is considered relevant if its value is less than the preset threshold value. However, Schenk et al. [147] do not state the threshold value used in their implementation.

In contrast to the implementation of Schenk et al. [147], we estimate the local region using a morphology operation as this technique is a lot easier. For connecting point $P_1$ with point $P_2$, we use the shortest edge segment taken from the contour of the previous slice. We dilate this segment with a $3 \times 3$ structuring element 15 times for segmenting the data in the axial direction, while 7 times for segmenting the data in the coronal and sagittal directions. The local region is the area defined by this dilated edge segment. An example is shown in Figure 6.8.

![Figure 6.8](image-url)
6.5 Stage 4: Combining the Segmentation Results

In this section, we combine the segmentations obtained using axial, coronal, and sagittal slices, in order to improve the reliability of the segmentation result.

6.5.1 Stand-alone segmentation in the axial direction

The user selects one of the simplest axial slices to initiate the segmentation. Using the technique described in section 6.3, the contour for this slice is generated.

This contour is used as the reference contour to segment its successive neighbouring axial slice(s). The seed points, which are the contour inflection points, are generated automatically, as described in subsection 6.4.1. Then, we create the mask, as described in subsection 6.4.3, and apply the point snap. Intelligent scissors is applied, and the liver contour for this slice is produced.

This contour is then taken as the starting point to segment the neighbouring slice. The process is repeated until the segmentation of the entire dataset is completed.

6.5.2 Combination of axial-coronal and axial-sagittal results

We execute two segmentations independently, i.e. axial-coronal segmentation, and axial-sagittal segmentation. If we consider the volume defined by axial-coronal segmentation as \( V_{ac} \), and the volume defined by axial-sagittal segmentation as \( V_{as} \), the combined output volume, \( V_{ac\&as} \), is given by:

\[
V_{ac\&as} = V_{ac} \cap V_{as}
\]

which means that we consider the voxels as valid liver voxels only when they are detected in both segmentations.

Axial-coronal segmentation

First we run the process described in subsection 6.5.1 completely, in order to get the volume of the liver. We use this segmented volume to create preliminary liver contours.
6.5. Stage 4: Combining the Segmentation Results

in the coronal slices. In each coronal slice, we divide equally the preliminary contour into 11 segments, and produce the seed points as described in section 6.4.1. We create the local mask, and apply the point snap algorithm. Then, the liver is segmented by intelligent scissors. We repeat the method for all the slices in the coronal direction.

Axial-sagittal segmentation

The axial-sagittal segmentation is done in the same way as the axial-coronal segmentation, except, instead of segmenting the slices in the coronal direction, we segment them in the sagittal direction.

6.5.3 Axial-coronal-sagittal segmentation

In this experiment, first we execute the axial-coronal segmentation as described in subsection 6.5.2. Then, we use the resultant segmented volume to extract preliminary contours in the sagittal direction. From these contours, we generate the seed points, and create the corresponding local masks. The seed points are estimated on the basis of the length of the curve, as described in section 6.4.1.

6.5.4 Axial-sagittal-coronal segmentation

In the axial-sagittal-coronal segmentation, we first execute the axial-sagittal segmentation as described in subsection 6.5.2. We use the resultant segmented volume to extract preliminary contours in the coronal direction. We generate the seed points, create the local mask, apply the point snap algorithm, and segment the liver using intelligent scissors. In this approach, the seed points are estimated using the method of subsection 6.4.1.
6.6 Results and Discussion

6.6.1 The input dataset

In this experiment, we use a dataset that has been pre-processed using the technique described in Chapter 4. This dataset consists of one hundred and thirty $256 \times 256$ axial slices, where each voxel is $1.0\text{mm} \times 1.0\text{mm} \times 1.2\text{mm}$ in dimensions. We label these axial slices with a numbering system, which indicates the position relative to the top most slice. The top most slice is given number 001, while the bottom most slice is given number 130. Figure 6.9 shows some selected slices from this dataset.

![Figure 6.9: Some axial slices taken from the input dataset used in this experiment. In most slices, the liver can be seen as the largest organ on the left side of the image.](image-url)
6.6. Results and Discussion

6.6.2 The ground truth

In this experiment, we use two sets of ground truth, for the purpose of evaluation. The ground truth is shown in the following subsections.

The ground truth provided by a radiologist

We only have 15 axial slices of ground truth that have been marked by a radiologist. These slices are shown in Figure 6.10. This set of ground truth shows the liver boundary (represented by the green contour), its maximum boundary limit (shown as the red contour), and its minimum boundary limit (shown as the yellow contour).
Figure 6.10: All 15 slices that represent the ground truth set, which is provided by a radiologist. The green contour represents the possible best contour of the liver. The red contour shows the maximum liver boundary limit, while the yellow contour represents the minimum liver boundary.
The ground truth for all the slices in the dataset

In order to evaluate the performance of our intelligent scissors in a more thorough manner, we manually segment the liver on every slice in the dataset (i.e. 130 individual axial slices) to create the ground truth. Some of these slices are shown in Figure 6.11.

Figure 6.11: Some axial slices to represent the ground truth that we use to evaluate the segmentation for the entire dataset.
6.6.3 Modification of the cost function

The cost function used in the implementation of intelligent scissors by Mortensen et al [109][110], is given by equation (2.3). From this equation, it is shown that their local cost on the directed link from pixel $p$ to neighbouring pixel $q$, $l(p, q)$, is constituted of the Laplacian zero crossing feature, $f_Z$, the gradient magnitude feature, $f_G$, and the gradient direction feature, $f_D$.

Mortensen et al [109][110] use the concept of multiple size kernels in order to calculate
the gradient and zero crossing features of the image. In their approach, the input image is convolved with Gaussian kernels of different sizes, from size $3 \times 3$ to $13 \times 13$, and produces several versions of smoothed images. Each of these smoothed image versions are then convolved with a Laplacian kernel, or first derivative convolution kernels in order to estimate $f_Z$, $f_G$, and $f_D$. The advantage of this technique is that the small kernels are suitable for finding weak edges, while the larger kernels improve the performance in noisy images.

In this experiment, our MRI liver dataset has already been pre-processed, thus, it can be considered as a noise-free dataset (see chapter 4). As a consequence, we may omit the stage of multiple size kernels in our implementation, and thus, save processing time.

The gradient magnitude feature, $f_G$, in this experiment is calculated by convolving the input slice with the simple first derivative kernels shown in Figure 5.3. Similar to the implementation in [110], the calculated gradient magnitude is inverted (in order to give the edges low cost value), and is scaled in the range 0 to 1.

Stalling and Hege [159], in their implementation of intelligent scissors for medical image segmentation, use only $f_G$ as their cost feature, and dismiss the use of $f_D$ and $f_Z$ from equation (2.3). They claim that, for medical images, $f_D$ does not play any significant role in finding suitable edge, and $f_Z$ is very sensitive to noise.

We were interested with this claim and decided to investigate its validity. Thus, we created a simple simulated image (see Figure 6.12(a)). In this image, there is an ellipse with six small circles inside it. The ellipse has intensity 200, while the circles have intensity 50. The background is given intensity 100. First, we corrupted this image with Gaussian noise (mean equal to zero, and $\sigma$ equal to 15). Then we filtered the image with a median filter of size $5 \times 5$, and re-enhanced the edges by using the toboggan enhancement technique.

Figure 6.12 shows the outputs from different setting of weight in equation (2.3). From this figure, it is clear that both $f_D$ and $f_Z$ requires a lot of computational time, but fail to find the correct edges, even in this very simple image. Thus, following what is proposed by Stalling and Hege [159], we dismiss the use of $f_Z$ and $f_D$ in our application.
Figure 6.12: Image (a) is the input image. Image (b) to (e) shows edge segments connecting points (250, 145) and (105, 85) using different weight setting of equation (2.3). (b) $w_z = 0.43$, $w_G = 0.43$, and $w_D = 0.14$, as suggested by Mortensen and Barrett [109]. 9 seconds is required to connect these points. (c) $w_z = 1$, $w_G = 0$, and $w_D = 0$. 12 seconds is required to connect these points. (d) $w_z = 0$, $w_G = 1$, and $w_D = 0$. 7 seconds is required to connect these points. (e) $w_z = 0$, $w_G = 0$, and $w_D = 1$. 14 seconds is required to connect these points. Figure (c) and (e) show that $f_z$ and $f_G$ fail to find the correct edge, even in this simple image. In contrast, $f_G$ requires the least computational time, but detects the right edge.

However, in reality, $f_G$ alone is not strong enough to attract the edge segment of intelligent scissors to follow the correct liver contour. This is because the parenchyma of the liver, which is represented by the step edges after the toobogganning process, has a significant high gradient magnitude value (i.e. low $f_G$). Although an individual edgels of the liver parenchyma has higher $f_G$ compared to the edgels on the liver surface, the accummulated value of $f_G$ when connecting two far seperated seed points commonly minimum by passing through the liver parenchyma rather than follows the liver surface. An example of this condition is shown in Figure 6.13(a).

The solution to this problem is to enhance further the edge strength of the liver surface,
while reduce the significance of the liver parenchyma. Fortunately, because the internal structure of the liver has shorter edgels and the edgels are located near to each other, the strength of this type of edgels can be reduced by using salienation process as described in Chapter 5. This is because the salienation process degrees the edges based on their length, curvature and the state of their closeness. Thus, the strength of liver parenchyma, reduces, while the strength of the liver edges, which are long edges, enhances. So, in our implementation, we use the cost function that is defined as:

\[ l(p,q) = \omega_G f_G(q) + \omega_S f_S(q) \]  \hspace{1cm} (6.9)

where \( f_S \) is the saliency map of the edgels, and \( \omega_S \) is its corresponding weight, i.e. \( \omega_G + \omega_S = 1 \). Figure 6.13 shows the advantage of using this equation compared by using \( f_G \) alone.

![Figure 6.13: Connecting points (100,105) and (15,125). Image (a) is the result when the cost only rely on \( f_G \). The minimum cost path according Dijkstra's algorithm passing through the liver parenchyma, and not follow the liver surface. Image (b) shows the result when we use equation (6.9) with \( \omega_G = 0.05 \) and \( \omega_S = 0.95 \). Using this equation, the edge segment now follows the right liver contour.](image)

Before intelligent scissors can be applied, the weighting factors in equation (6.9) have to be set. Guided by the example which is shown in Figure 6.14, we decide to set \( \omega_S = 0.95 \), and \( \omega_G = 0.05 \). This is because, \( f_S \) is more reliable in attracting the contour to the edges. However, we cannot eliminate \( f_G \) completely from the cost function as this feature may be useful in connecting segments which are not defined by \( f_S \), because they are not salient enough.
Figure 6.14: (a) A simulated input image (which is taken from Figure 6.12(a)). (b) The corresponding gradient magnitude feature, \( f_c \). (c) The corresponding pre-attentive segmentation feature, \( f_s \). The results of segmenting the elliptic object, by connecting point \((60, 110)\) with point \((220, 150)\), when (d) \( \omega_C = 1.00, \omega_S = 0.00 \), (e) \( \omega_C = 0.95, \omega_S = 0.05 \), (f) \( \omega_C = 0.75, \omega_S = 0.25 \), (g) \( \omega_C = 0.50, \omega_S = 0.50 \), (h) \( \omega_C = 0.25, \omega_S = 0.75 \), and (i) \( \omega_C = 0.05, \omega_S = 0.95 \). This example shows that the number of edgels on the boundary of the ellipse (shown by the number on the top right of each corresponding image), which indicates the quality of the segmentation, increases as the value of \( \omega_S \) increases.
6.6. Stage 1: Selection of the initial slice

Based on the criteria listed in section 6.2, we select the twelfth slice from the top as the initial slice. This slice is shown in Figure 6.9(b). We select the initial slice among the top slice because the liver in these slices always resembles a circle or an ellipse. Also on these slices, the liver is surrounded by the lung. Thus, in these slices, the liver always has a good contrast with its surrounding.

6.6.5 Stage 2: Segmentation of the initial slice

Figure 6.15 shows the corresponding saliency map of image 6.9(b) after thresholding, thinning, and deletion of short edge segments. It is shown that unwanted edge segments in this slice can be reduced significantly using this technique.

Figure 6.15: Results of each step described in section 6.3.1. Starting with the saliency map shown in (a), the image is thresholded. Image (b) shows this output. Next, this output is skeletonized, producing image (c). Then, we remove edge segments which are less than 25 pixels in length. The output is shown in (d).
Chapter 6. Automatic intelligent scissors for MRI liver segmentation

The dimensions of this axial image is $256 \times 256$. As the liver is located in the left side of the image, the estimated centre of the liver curve also lies in the left side of the image. Thus, we restrict the solution of $(x_c, y_c)$ to be in this region, i.e. $0 \leq x_c \leq 128$, and $64 \leq y_c \leq 192$.

There are 2220 edgels in image 6.15(c). Thus, we randomly select 6660 (i.e. $3 \times 2220$) triplets in order to find the centre. We use an accumulator array with size $64 \times 64$, which mean that each cell of this array represents $4 \times 4$ pixel$^2$ area.

Figure 6.16 shows the accumulator array for the image shown in Figure 6.15(c). It is shown that the peak of this histogram is at accumulator cell $(15, 31)$, which means that $(15 \times 4 = 60) \leq x_c < (16 \times 4 = 64)$ and $(31 \times 4 = 124) \leq y_c < (32 \times 4 = 128)$. Thus we assume that the centre, $(x_c, y_c)$, is $(62, 126)$.

![Accumulator Array](image)

When we project straight lines at every $15^\circ$ from $(x_c, y_c)$, 24 seed points are obtained. These seed points are shown in Figure 6.17(a). The segmentation of the liver, by using the intelligent scissors technique, for these seed points produce a good result, as shown in Figure 6.17(b).

We also tried to project the straight lines emanating from the centre using other than $15^\circ$. We found that if we use the projection degree in between $5^\circ$ (i.e. 72 seed points) to $45^\circ$ (i.e 8 seed points), intelligent scissors produce the same output contour. This is because on the initial slice, the liver has a very strong edges. Thus, the edge segments
of intelligent scissors attracted to the same contour, and produces the same result.

In order to see the realibility of the technique, we tried to use Slice 014 instead of Slice 012, where criteria stated in section 6.2 is not fulfilled. The liver in this slice does not really resembles an ellipse, and a part of the liver surface touches the heart. This means the corresponding saliency map does not form a closed liver contour. Yet, the algorithm still produces a visually acceptable result as shown in Figure 6.18.

![Figure 6.17](image1.png)

Figure 6.17: (a) 24 seed points detected by projecting straight lines at every 15° from the centre (indicated by a grey box). (b) The result of intelligent scissors segmentation, using the seed points shown in (a), to delineate the liver.

![Figure 6.18](image2.png)

Figure 6.18: (a) The salienation map for Slice 014. (b) The centre and the corresponding seed points estimated from this centre. (c) The result of intelligent scissors segmentation using the seed points shown in (b).
6.6.6 Stage 3: Propagation of the contour in the axial direction

Figure 6.19 shows some of the results of the segmentation in the axial direction. From this figure, we can see that most of the slices are over segmented. However, under segmentation also does occur. For example, in slice 055, the segmentation misses to include the top right part of the liver, which appears relatively brighter compared with the remaining liver area.

Figure 6.19: Some axial slices of the resultant segmentation in the axial direction.
6.6. Results and Discussion

6.6.7 Stage 4: Combining the segmentation results

Results from the combination of axial-coronal and axial-sagittal segmentations

The results by combining the axial-coronal segmentation with the axial-sagittal segmentation, using equation (6.8), are shown in Figure 6.20.
Figure 6.20: Some axial slices of the resultant segmentation of the combination of the axial-coronal segmentation with the axial-sagittal segmentation.
6.6. Results and Discussion

Results of axial-coronal-sagittal segmentation

Figure 6.21 shows some of the results of the segmentation in the axial-coronal-sagittal direction, as described in section 6.5.3.

Figure 6.21: Some axial slices of the resultant axial-coronal-sagittal segmentation.
Figure 6.21: Continued.
6.6. Results and Discussion

Results of the axial-sagittal-coronal segmentation

Figure 6.22 shows some of the results of the segmentation in the axial-coronal-sagittal direction, as described in section 6.5.4.
Figure 6.22: Continued.
6.6.8 Evaluation of the results

In this section, we refer to the axial segmentation as $S_A$, the combination of axial-coronal and axial-sagittal as $S_{AC-AS}$, the axial-coronal-sagittal segmentation as $S_{ACS}$, and the axial-sagittal-coronal segmentation as $S_{ASC}$.

By inspecting as individual slices, we may consider $S_A$, $S_{AC-AS}$, $S_{ACS}$, and $S_{ASC}$ produce the acceptable results, because the liver contours do not deviate much from the ground truth in most slices. The results are useful for 3D visualization (see Figure 6.23), but, as the detected contours, especially on the bottom slices, do not exactly lie on the liver boundary, the results cannot be applied to the applications where an accurate measurement of liver volume is needed.

![Figure 6.23: The 3D models created from (a) the ground truth data, (b) the results of $S_A$ (see Figure 6.19), (c) the results of $S_{AC-AS}$ (see Figure 6.20), (d) the results of $S_{ACS}$ (see Figure 6.21), and (e) the results of $S_{ASC}$ (see Figure 6.22). All of these models are created by using a marching cube algorithm, and are viewed from the same viewing direction, i.e. from the coronal direction.](image)

Figure 6.24 shows the segmented area on each slice of the dataset, and the corresponding accumulated volume, relative to the top most slice. This figure gives us the first impression that the results from $S_{AC-AS}$, $S_{ACS}$, and $S_{ASC}$ are almost identical, and better than $S_A$. This figure also shows that the liver in all cases, especially the bottom part of it, is never under segmented.
The main reason why all methods tested over-segment the bottom slices is because there are so many organs contained in these slices compared to those in the middle and top sections of the dataset. By referring to Figure 6.9(v), for an example, the right kidney, which has a very similar intensity range with the liver, merged with the liver area. As there is no strong edges available to separate these organs, the segmentation methods we used do not have any choice except to include the right kidney as a part of the liver area. The problem is aggravated as the intestine, which has a relatively strong contrast, is also located near to the liver. This condition makes the intelligent scissors select the edges of intestine rather than the edges of the liver. Thus, the segmented liver area is bigger than what we expect.

In order to find the best method among $S_A$, $S_{AC-AS}$, $S_{ACS}$ and $S_{ASC}$, we evaluate the quality of the segmentation by computing the segmentation errors. The best method is the one that gives the least total segmentation error, $E_t$. For this purpose, we use both sets of the ground truth mentioned in section 6.6.2.

First, we quantify the segmentation error by using the ground truth defined by the radiologist. In order to do this, we consider only 15 slices, those from slice 020 to slice 034. Over segmentation error, $E_o$, and under segmentation error, $E_u$, are defined by
considering the Venn Diagram shown in Figure 6.25.

**Figure 6.25:** The Venn Diagram used to estimate the segmentation errors using the ground truth provided by the radiologist in section 6.6.2. Region \((A \cup B \cup C \cup D)\) presents the maximum liver volume defined by the radiologist. The minimum liver volume defined by the radiologist is represented by region \((B \cup C)\). The volume defined by the intelligent scissors' segmentation is represented by region \((C \cup D \cup E)\).

The error measurements based on the maximum liver volume defined by the radiologist, are given by:

\[
E_{o1} = \frac{E}{(C \cup D \cup E)} \times 100
\]
\[
E_{u1} = \frac{A \cup B}{(A \cup B \cup C \cup D)} \times 100
\]
\[
E_{e1} = E_{o1} + E_{u1}
\]  
(6.10)

The error measurements based on the minimum liver volume defined by the radiologist, are given by:

\[
E_{o2} = \frac{(E \cup D)}{(C \cup D \cup E)} \times 100
\]
\[
E_{u2} = \frac{B}{(B \cup C)} \times 100
\]
\[
E_{e2} = E_{o2} + E_{u2}
\]  
(6.11)

Table 6.3 gives the measurements of these errors for every technique applied in this experiment. From equation (6.10) and (6.11), we define:

\[
E_{o_{\text{min}}}=\min\{E_{o1},E_{o2}\} \quad E_{o_{\text{max}}}=\max\{E_{o1},E_{o2}\}
\]
\[
E_{u_{\text{min}}}=\min\{E_{u1},E_{u2}\} \quad E_{u_{\text{max}}}=\max\{E_{u1},E_{u2}\}
\]
\[
E_{e_{\text{min}}}=\min\{E_{e1},E_{e2}\} \quad E_{e_{\text{max}}}=\max\{E_{e1},E_{e2}\}
\]  
(6.12)
Table 6.3: Measurements of errors using the ground truth provided by the radiologist.

<table>
<thead>
<tr>
<th>Technique</th>
<th>$E_{o1}$</th>
<th>$E_{u1}$</th>
<th>$E_{o2}$</th>
<th>$E_{u2}$</th>
<th>$E_{t}$</th>
<th>$E_{t2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_A$</td>
<td>9.86</td>
<td>7.18</td>
<td>20.10</td>
<td>1.46</td>
<td>17.04</td>
<td>21.56</td>
</tr>
<tr>
<td>$S_{AC-AS}$</td>
<td>2.48</td>
<td><strong>11.99</strong></td>
<td><strong>10.95</strong></td>
<td><strong>3.82</strong></td>
<td><strong>14.47</strong></td>
<td><strong>14.77</strong></td>
</tr>
<tr>
<td>$S_{ACS}$</td>
<td>3.35</td>
<td>11.38</td>
<td>12.14</td>
<td>3.52</td>
<td>14.73</td>
<td>15.05</td>
</tr>
<tr>
<td>$S_{ASC}$</td>
<td>3.53</td>
<td>9.27</td>
<td>13.94</td>
<td>2.95</td>
<td>12.80</td>
<td>16.79</td>
</tr>
</tbody>
</table>

Thus, $E_o$ in the range of $E_{omin}$ to $E_{omax}$, $E_u$ in the range of $E_{umin}$ to $E_{umax}$, and $E_t$ in the range of $E_{tmin}$ to $E_{tmax}$. Figure 6.26 shows the plot of these ranges of errors for every technique applied in this experiment.

![Figure 6.26](image)

Figure 6.26: The bar graph showing the ranges of errors (min, max) for every technique applied in this experiment.

From Figure 6.26, it can be seen that the result from $S_A$ is the poorest as this technique has the highest $E_o$ and $E_t$. It is also shown that any combination of segmentation from different directions reduces the $E_o$ and $E_t$. We can see that $S_{AC-AS}$ has the lowest $E_{tmax}$, and thus it can be considered as the best technique for this input.

Next, we evaluate the performance over all the slices in this 3D dataset. In order to do this, we use the ground truth from section 6.6.2 that we manually segment. As in the previous evaluation, we also judge the performance based on $E_o$, $E_u$, and $E_t$. However, these calculations are slightly different. We consider the Venn Diagram shown in Figure 6.27 to calculate these errors.
Figure 6.27: The Venn Diagram to estimate the segmentation errors using the ground truth for all slices in the dataset. Region \((A \cup B)\) represents the liver volume from the ground truth, while region \((B \cup C)\) represents the liver volume from the segmentation.

From this figure, \(E_o\), \(E_u\), and \(E_t\) are given by:

\[
\begin{align*}
E_o &= \frac{C}{(B \cup C)} \times 100 \\
E_u &= \frac{A}{(A \cup B)} \times 100 \\
E_t &= E_o + E_u
\end{align*}
\]  

Table 6.4 gives the measurements of these errors for every technique applied in this experiment. The results in this table again show that \(S_A\) produces the highest \(E_o\) and \(E_t\), and \(S_{AC-AS}\) produces the lowest \(E_t\).

<table>
<thead>
<tr>
<th>Technique</th>
<th>(E_o)</th>
<th>(E_u)</th>
<th>(E_t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S_A)</td>
<td>24.71</td>
<td>7.03</td>
<td>31.75</td>
</tr>
<tr>
<td>(S_{AC-AS})</td>
<td>18.14</td>
<td>9.79</td>
<td>27.93</td>
</tr>
<tr>
<td>(S_{ACS})</td>
<td>19.51</td>
<td>8.79</td>
<td>28.30</td>
</tr>
<tr>
<td>(S_{ASC})</td>
<td>19.32</td>
<td>9.05</td>
<td>28.37</td>
</tr>
</tbody>
</table>

Table 6.4: Measurements of errors using the ground truth of section 6.6.2.

As \(S_{AC-AS}\) produces minimum \(E_t\) in both Table 6.3 and Table 6.4, we conclude that this technique is the best method among those four methods tested in this experiment.

6.7 Conclusions

In this experiment, we adapted the intelligent scissors algorithm in order to allow it to perform automatically. In the cost function, we eliminated the gradient direction feature, \(f_D\), and zero crossing feature, \(f_Z\), because these features were not significant in the segmentation. However, we introduced saliency map feature, \(f_S\), because this feature is better than \(f_Z\) for edge localisation. We considered that our input dataset is a noise free dataset, so, we did not use the idea of multiple kernels for calculating
the gradient magnitude feature, $f_G$. These alterations simplified the complexity of the cost function.

We demonstrated that the segmentation in the axial direction only produces poor results. Thus, we tried to reduce these errors by combining the segmentation results from three orthogonal directions.

We investigated three types of combination method. First, the combination of axial-coronal results with axial-sagittal results ($S_{AC-AS}$). The second method was the axial-coronal-sagittal method ($S_{ACS}$). The third method was the axial-sagittal-coronal method ($S_{ASC}$). The results from these three methods are almost identical. However, the errors in $S_{AC-AS}$ are lower than for the other two methods. Thus, we consider this technique to be the best.
Chapter 7

Evaluation of the algorithm

The idea in Chapters 4 – 6 is to develop a general purpose methodology from one set of data. In this chapter, we shall test this methodology using the remaining sets of data provided to us.

Based on the previous chapters, our proposed approach to segment the liver from 3D MRI datasets can be represented by the block diagram shown in Figure 7.1. We keep the parameter values in each stage of processing shown in this figure exactly the same as what have been derived from Chapters 4 – 6.

The 3D preprocessing chain used in this chapter consists of local enhancement, median filtering, and toboggan contrast enhancement. As concluded in Chapter 4, the local enhancement kernel that we shall use is of size $27 \times 27 \times 27$, and parameter $k$ in equation (4.1) is set to 10000. The 3D median filtering approach will use a kernel size of $5 \times 5 \times 5$.

The outputs from the preprocessing stage will be channeled to the salienation process, with time step, $\Delta t$, equal to 0.075. We shall take the saliency map at time $t$ equal to 0.075 as the output of the process. The saliency map will be used as one of the cost features in the automatic intelligent scissors. The saliency map also will be used to determine the seed points in the starting axial slice, which initiises the segmentation process.
The automatic intelligent scissors will use the cost function defined by equation (6.9), with \( \omega_S = 0.95 \) and \( \omega_G = 0.05 \). As concluded in Chapter 6, the output of this segmentation method will be the combination of results from orthogonal directions (i.e. \( S_{AC-AS} \)) as defined by equation (6.8).

The performance of the method will be evaluated based on the segmentation outputs. The results will be compared with the corresponding segmentation outputs from the original implementation of intelligent scissors, as described in section 2.1.12. The evaluation will be done by computing the segmentation errors (i.e. over segmentation error \( E_o \), under segmentation error \( E_u \) and total segmentation error \( E_t \)) as defined by equation (6.13), and also by reporting the computing time.

Figure 7.1: The block diagram showing the process that will be carried out in this evaluation.
7.1 Input Datasets

In this evaluation, eight complete 3D MRI liver datasets have been used. These datasets were acquired on a SIEMENS Magnetom Vision Scanner at the CRC Clinical Magnetic Resonance Centre, Institute of Cancer Research, London. Eight datasets are enough for an evaluation purpose, as many researchers use less than five datasets in their work, (for example [39, 48, 51, 147]). We refer to these testing datasets as Dataset01, Dataset02, ..., Dataset08. Unlike the training dataset (i.e. the dataset that we used in Chapters 4 - 6), which is the scan of a volunteer, these testing datasets are scans of patients. So, in order to keep the patients' details unknown, these datasets were supplied without any header in their files. Unfortunately, this also removed the information about the acquisition settings used for these datasets.

In addition to this, each testing dataset, except Dataset08, only consists of thirteen 256 X 256 axial slices (Dataset08 consists of eighteen axial slices). The dimensions of each voxel are 1mm x 1mm x 8mm. However, from our preliminary experiments, with this slice thickness, the algorithm fails to segment the liver due to the violation of the basic assumption on which the presented work lies, namely that the slices are dense enough so the shape of the liver does not change significantly from one slice to the next. Thus, in order to increase the sampling rate of the datasets along the z direction, we interpolate two successive axial slices to produce seven new slices in between. (Thus, each dataset, except Dataset08, now consists of 97 axial slices. Dataset08 now consists of 137 slices). After the interpolation, we can assume that each voxel in this dataset is 1mm x 1mm x 1mm in size. The interpolation technique used is defined as follows:

\[ g(x, y) = \frac{(8 - r)f_1(x, y) + rf_2(x, y)}{8} \]  

(7.1)

where \( g(x, y) \) is the intensity of the interpolated slice at position \((x, y)\), while \( f_1(x, y) \) and \( f_2(x, y) \) are the intensities of the first and the second input slices, respectively. The value of \( r \) is the distance between \( g(x, y) \) and \( f_1(x, y) \).

Some selected axial slices from each dataset, including the results from 3D preprocessing, are shown in Figures 7.2 to 7.9. These figures show that the inputs vary in edge quality and contrast.
Figure 7.2: Dataset01. (a) Raw data. (b) Preprocessed version of (a). (c) Histogram of the raw data. (d) Histogram of the preprocessed data.
Figure 7.3: Dataset02. (a) Raw data. (b) Preprocessed version of (a). (c) Histogram of the raw data. (d) Histogram of the preprocessed data.
Figure 7.4: Dataset03. (a) Raw data. (b) Preprocessed version of (a). (c) Histogram of the raw data. (d) Histogram of the preprocessed data.
Figure 7.5: Dataset04. (a) Raw data. (b) Preprocessed version of (a). (c) Histogram of the raw data. (d) Histogram of the preprocessed data.
Figure 7.6: Dataset05. (a) Raw data. (b) Preprocessed version of (a). (c) Histogram of the raw data. (d) Histogram of the preprocessed data.
Figure 7.7: Dataset06. (a) Raw data. (b) Preprocessed version of (a). (c) Histogram of the raw data. (d) Histogram of the preprocessed data.
Figure 7.8: Dataset07. (a) Raw data. (b) Preprocessed version of (a). (c) Histogram of the raw data. (d) Histogram of the preprocessed data.
Figure 7.9: Dataset08. (a) Raw data. (b) Preprocessed version of (a). (c) Histogram of the raw data. (d) Histogram of the preprocessed data.
Chapter 7. Evaluation of the algorithm

Subfigures (c) in Figures 7.2 – 7.9 present the histograms of the raw datasets. These histograms show the variability of the intensity distributions among the input datasets we use. Although some of these datasets have two distinct peaks in their histogram, none of these peaks specifically represents the intensity of the liver. Thus, a simple thresholding technique, such as what is commonly used to segment anatomical structures from a CT dataset, cannot be used to detect the liver. An example is shown in Figure 7.10.

![Images of histograms and thresholding results](image)

**Figure 7.10:** This figure shows Slice 68 of Dataset 06 and the output of a simple thresholding technique. Figure (b) shows the result when the threshold is set to 50, which is the value on the valley between the two peaks in the histogram shown in Figure 7.7(c). Figure (c) shows the best result we can get by using the thresholding technique, where the threshold value is selected based on a trial and error process. Both threshold values fail to detect the liver correctly. (The black regions in (b) and (c) represent the areas with intensity values less than the threshold value, while the regions in white represent the areas with intensity values greater or equal to the threshold value.)

Subfigures (d) in Figures 7.2 – 7.9 present the histograms of the preprocessed datasets. These histograms show that the intensity ranges after the preprocessing become smaller, and the peaks that are visible in the corresponding histograms of the raw datasets, disappear. The reason for this is because the preprocessing we use reduces the intensity variation in the datasets in order to decrease the inhomogeneity artifact. Although this condition makes the segmentation based on a simple thresholding technique become more difficult, the strength of the liver edges is enhanced in most slices. An investigation whether preprocessing is a necessary step or an optional add-on to our segmentation method, will be carried out in section 7.2.2.
7.2 Results and Discussion

7.2.1 Results of the segmentation

In this section, we segment the liver from the already preprocessed datasets. It must be stressed that the algorithm was run with all its parameter values fixed to be the same for all datasets. Table 7.1 shows the initial slices which we selected to initialise the segmentation in each dataset. These slices are selected based on visual inspection, guided by the criteria listed in section 6.2, which are: (1) the liver has a very good contrast with its surroundings, and (2) the shape of the liver resembles a circle or an ellipse. We found that the liver surrounded by the lung always fulfils both requirements. Thus, the initial slices are selected among the first one third of each dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Initial slice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset01</td>
<td>Slice023 from 97 slices in the dataset</td>
</tr>
<tr>
<td>Dataset02</td>
<td>Slice021 from 97 slices in the dataset</td>
</tr>
<tr>
<td>Dataset03</td>
<td>Slice027 from 97 slices in the dataset</td>
</tr>
<tr>
<td>Dataset04</td>
<td>Slice016 from 97 slices in the dataset</td>
</tr>
<tr>
<td>Dataset05</td>
<td>Slice025 from 97 slices in the dataset</td>
</tr>
<tr>
<td>Dataset06</td>
<td>Slice018 from 97 slices in the dataset</td>
</tr>
<tr>
<td>Dataset07</td>
<td>Slice034 from 97 slices in the dataset</td>
</tr>
<tr>
<td>Dataset08</td>
<td>Slice016 from 137 slices in the dataset</td>
</tr>
</tbody>
</table>

Table 7.1: Initial slices used to initialise the segmentation.

By using the initial slices listed in Table 7.1, we execute the automatic intelligent scissors to segment the liver surface from all preprocessed datasets. In order to compare the performance of our method with a baseline model, we also segment these preprocessed datasets using interactive intelligent scissors, as described in section 2.1.12. The results from both segmentation techniques, together with the corresponding ground truth are shown in Figure 7.11 to Figure 7.18. (We create these ground truths by manually tracing the liver in the raw datasets).
Figure 7.11: Dataset01. (a) The ground truth, superimposed to the raw input image. (b) The 3D shaded surface model representation of the ground truth. (c) The segmentation result using automatic intelligent scissors, superimposed to the preprocessed dataset.
Figure 7.11: (Continued.) Dataset01. (d) The 3D shaded surface model representation of the segmentation result using automatic intelligent scissors. (e) The segmentation result using interactive intelligent scissors, superimposed to the preprocessed image. (f) The 3D shaded surface model representation of the segmentation result using interactive intelligent scissors.
Figure 7.12: Dataset02. (a) The ground truth, superimposed to the raw input image. (b) The 3D shaded surface model representation of the ground truth. (c) The segmentation result using automatic intelligent scissors, superimposed to the preprocessed dataset.
Figure 7.12: (Continued.) Dataset02. (d) The 3D shaded surface model representation of the segmentation result using automatic intelligent scissors. (e) The segmentation result using interactive intelligent scissors, superimposed to the preprocessed image. (f) The 3D shaded surface model representation of the segmentation result using interactive intelligent scissors.
Figure 7.13: Dataset03. (a) The ground truth, superimposed to the raw input image. (b) The 3D shaded surface model representation of the ground truth. (c) The segmentation result using automatic intelligent scissors, superimposed to the preprocessed dataset.
Figure 7.13: (Continued.) Dataset 03. (d) The 3D shaded surface model representation of the segmentation result using automatic intelligent scissors. (e) The segmentation result using interactive intelligent scissors, superimposed to the preprocessed image. (f) The 3D shaded surface model representation of the segmentation result using interactive intelligent scissors.
Figure 7.14: Dataset04. (a) The ground truth, superimposed to the raw input image. (b) The 3D shaded surface model representation of the ground truth. (c) The segmentation result using automatic intelligent scissors, superimposed to the preprocessed dataset.
Figure 7.14: (Continued.) Dataset04. (d) The 3D shaded surface model representation of the segmentation result using automatic intelligent scissors. (e) The segmentation result using interactive intelligent scissors, superimposed to the preprocessed image. (f) The 3D shaded surface model representation of the segmentation result using interactive intelligent scissors.
Figure 7.15: Dataset05. (a) The ground truth, superimposed to the raw input image. (b) The 3D shaded surface model representation of the ground truth. (c) The segmentation result using automatic intelligent scissors, superimposed to the preprocessed dataset.
Figure 7.15: (Continued.) Dataset05. (d) The 3D shaded surface model representation of the segmentation result using automatic intelligent scissors. (e) The segmentation result using interactive intelligent scissors, superimposed to the preprocessed image. (f) The 3D shaded surface model representation of the segmentation result using interactive intelligent scissors.
Figure 7.16: Dataset06. (a) The ground truth, superimposed to the raw input image. (b) The 3D shaded surface model representation of the ground truth. (c) The segmentation result using automatic intelligent scissors, superimposed to the preprocessed dataset.
7.2. Results and Discussion

Figure 7.16: (Continued.) Dataset06. (d) The 3D shaded surface model representation of the segmentation result using automatic intelligent scissors. (e) The segmentation result using interactive intelligent scissors, superimposed on the preprocessed image. (f) The 3D shaded surface model representation of the segmentation result using interactive intelligent scissors.
Chapter 7. Evaluation of the algorithm

Figure 7.17. Dataset07. (a) The ground truth, superimposed to the raw input image. (b) The 3D shaded surface model representation of the ground truth. (c) The segmentation result using automatic intelligent scissors, superimposed to the preprocessed dataset.
7.2. Results and Discussion

Figure 7.17: (Continued.) Dataset07. (d) The 3D shaded surface model representation of the segmentation result using automatic intelligent scissors. (e) The segmentation result using interactive intelligent scissors, superimposed to the preprocessed image. (f) The 3D shaded surface model representation of the segmentation result using interactive intelligent scissors.
Figure 7.18: Dataset 58. (a) The ground truth, superimposed to the raw input image. (b) The 3D shaded surface model representation of the ground truth. (c) The segmentation result using automatic intelligent scissors, superimposed to the preprocessed dataset.
Figure 7.18: (Continued.) Dataset08. (d) The 3D shaded surface model representation of the segmentation result using automatic intelligent scissors. (e) The segmentation result using interactive intelligent scissors, superimposed to the preprocessed image. (f) The 3D shaded surface model representation of the segmentation result using interactive intelligent scissors.
Chapter 7. Evaluation of the algorithm

We can say that the results from Dataset01, Dataset02, and Dataset03 are perfect segmentation results because all the detected contours lie near to the correct boundary of the liver (i.e., within margins of the same order of magnitude as the deviations of the contours created manually by the radiologist). Results from Dataset04 and Dataset05 may be considered still acceptable, because only a few slices from these datasets were badly segmented. Results from Dataset06, Dataset07, and Dataset08 are rejected because the majority of the slices were segmented badly. As five out of eight datasets may be accepted, we can conclude that the segmentation technique we use is promising to segment the liver despite the large variation in edge quality and contrast.

By using equation (6.13), we measure the over detection error, $E_o$, under detection error, $E_u$, and total detection error, $E_t$. Table 7.2 gives the error values for every dataset used in this study. From this table, we can see that the error values vary significantly between datasets. This is because the input datasets we used are different from each other, and vary in edge quality and contrast.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Automatic Intelligent Scissors</th>
<th>Interactive Intelligent Scissors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$E_o$</td>
<td>$E_u$</td>
</tr>
<tr>
<td>Dataset01</td>
<td>9.78</td>
<td>1.39</td>
</tr>
<tr>
<td>Dataset02</td>
<td>8.21</td>
<td>3.46</td>
</tr>
<tr>
<td>Dataset03</td>
<td>7.71</td>
<td>4.63</td>
</tr>
<tr>
<td>Dataset04</td>
<td>17.84</td>
<td>7.33</td>
</tr>
<tr>
<td>Dataset05</td>
<td>22.73</td>
<td>3.05</td>
</tr>
<tr>
<td>Dataset06</td>
<td>14.03</td>
<td>47.73</td>
</tr>
<tr>
<td>Dataset07</td>
<td>13.37</td>
<td>60.45</td>
</tr>
<tr>
<td>Dataset08</td>
<td>24.25</td>
<td>39.92</td>
</tr>
</tbody>
</table>

Table 7.2: Measurements of errors for all datasets tested in this evaluation.

The results with $E_t$ greater than 30% underestimate the liver, as the internal structures of the liver in these datasets, such as the blood vessels, have relatively strong edges compared with the liver surface. Thus, this attracts the edge segments to lie on the blood vessels rather than the liver surface.
7.2. Results and Discussion

The outputs from interactive intelligent scissors (except Dataset01) always have errors smaller than those detected by automatic intelligent scissors. This is because interactive intelligent scissors allow the user to control the edge segments to follow the desired contour. Yet, the automatic segmentation method is preferable than the interactive segmentation method. Although automatic segmentation requires a lot of time, in case there are more than one computers available, many segmentation processes can be executed in parallel, and this can be controlled by only a single radiologist. An interactive segmentation, on the other hand requires a lot of concentration from the radiologist, and the result depends highly on the human factor, such as the skill and level of tiredness of the radiologist at that time.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Automatic Intelligent Scissors</th>
<th>Interactive Intelligent Scissors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset01</td>
<td>12:30</td>
<td>1:50</td>
</tr>
<tr>
<td>Dataset02</td>
<td>12:45</td>
<td>2:10</td>
</tr>
<tr>
<td>Dataset03</td>
<td>12:10</td>
<td>2:20</td>
</tr>
<tr>
<td>Dataset04</td>
<td>12:30</td>
<td>2:00</td>
</tr>
<tr>
<td>Dataset05</td>
<td>12:00</td>
<td>2:00</td>
</tr>
<tr>
<td>Dataset06</td>
<td>12:30</td>
<td>2:30</td>
</tr>
<tr>
<td>Dataset07</td>
<td>12:15</td>
<td>2:20</td>
</tr>
<tr>
<td>Dataset08</td>
<td>17:30</td>
<td>2:50</td>
</tr>
</tbody>
</table>

Table 7.3: Segmentation time.

Table 7.3 shows the total segmentation time required to segment each dataset in this work. The automatic segmentation method requires a lot of computational time, but actually, for each dataset used, only less than five minutes of user intervention are needed to load the input to the system. Thus, we conclude that the automatic segmentation method is better than the interactive segmentation method, because this technique reduces the intervention of the user significantly.

\[^1\text{No effort has been made to make the programs fast. Most of the computing time is taken up by the salienation algorithm, which takes about 4 minutes for each iteration for each slice. However, recently, it has been reported that this algorithm was implemented to run in near real time [129]}\]
Chapter 7. Evaluation of the algorithm

7.2.2 The role of preprocessing towards segmentation

In order to see the significance of preprocessing to the segmentation result, we tried to segment the liver directly from the original unprocessed input dataset. To initialise the segmentation in each dataset, we use the initial slice listed in Table 7.1. However, our segmentation method failed even to generate the seed points correctly in the initial slice of all datasets. The generated seed points always lied inside the liver, and not on the liver edges, and too close with each other, which restricted the deformation of the intelligent scissors. This is because the generation of the seed points in the initial slice, as described in section 6.2, uses the information from the saliency map, and the unprocessed raw data produces a bad saliency map. An example of how bad the saliency map is in the case of raw data is shown in Figure 7.19.

![Figure 7.19: Saliency maps of the initial slice to segment Dataset01. (a) The saliency map of the raw data, without any preprocessing. (b) The saliency map of the preprocessed data. The saliency map shown in (b) is better compared with (a) because it contains fewer inner structures of the liver.](image)

Yet, as we want to know whether the liver edges in the raw datasets have a significant strength to attract the intelligent scissors to follow the true liver boundary or not, we placed manually the seed points on the initial slice of the unprocessed input dataset. We choose all initial seed points to be on the liver boundary, and we made sure that these seed points produced an acceptable reference contour. If the liver edges in the raw datasets were strong enough, the segmented contour should be lying near the true boundary. However, for all datasets tested, this was not the case.

Rather than following the correct boundary, the lowest accumulated cost path, com-
7.2. Results and Discussion

The results are computed by the intelligent scissors (using Dijkstra’s dynamic programming algorithm), is the one that passes through the liver parenchyma. Thus, all results that we got had a very high under detection error, $E_u$. The reason for this is because we employed the saliency map as one of the cost features. As we saw previously in Figure 7.19(a), the saliency map of the raw data is condensed with the short edges inside the liver. Instead of the liver contour, these short edges contribute to the segmentation process.

![Image of the liver segmentation process](image)

(a) The ground truth  (b) Without preprocessing  (c) With preprocessing

Figure 7.20: The 3D models represent the segmentation results of Dataset01. All models are shown in the same scale, and are viewed from the same viewing direction. This figure shows that preprocessing improves the segmentation result significantly.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$E_o$</th>
<th>$E_u$</th>
<th>$E_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset01</td>
<td>0.00</td>
<td>97.55</td>
<td>97.55</td>
</tr>
<tr>
<td>Dataset02</td>
<td>0.00</td>
<td>96.87</td>
<td>96.87</td>
</tr>
<tr>
<td>Dataset03</td>
<td>0.00</td>
<td>91.43</td>
<td>91.43</td>
</tr>
<tr>
<td>Dataset04</td>
<td>0.00</td>
<td>97.66</td>
<td>97.66</td>
</tr>
<tr>
<td>Dataset05</td>
<td>0.00</td>
<td>97.55</td>
<td>97.55</td>
</tr>
<tr>
<td>Dataset06</td>
<td>0.00</td>
<td>90.07</td>
<td>90.07</td>
</tr>
<tr>
<td>Dataset07</td>
<td>0.00</td>
<td>92.08</td>
<td>92.08</td>
</tr>
<tr>
<td>Dataset08</td>
<td>0.00</td>
<td>96.89</td>
<td>96.89</td>
</tr>
</tbody>
</table>

Table 7.4: Measurements of errors associated with the segmentation of the raw datasets.

Figure 7.20 shows how bad the segmentation result is using the raw dataset, and Table 7.4 shows the associated error values based on the same ground truths we used in section 7.2.1. As all datasets we tested had total segmentation error, $E_t$, higher than 90%, this strongly indicates that the results of using the raw datasets are not acceptable. Comparing these values with the values obtained using the preprocessed datasets...
(see Table 7.2), we conclude that the preprocessing chain is a necessary step for the automatic intelligent scissors method.

7.3 Conclusions

The segmentation method was tested on eight datasets, and five out of eight of them produce an impressively close approximation to the true liver boundary (with total segmentation error less than 30%), despite of the large variation in edge quality and contrast, and the lack of any adjustment of the parameters of the algorithm.

Compared with interactive intelligent scissors, which is guided directly by the user, the automatic intelligent scissors produces higher segmentation errors. However, automatic intelligent scissors have advantages on reducing the intervention needed. Thus, the automatic segmentation can reduce the burden of the radiologist.

In this chapter also, we demonstrated that a preprocessing chain which consists of local enhancement, followed by median filtering and tobgogen contrast enhancement, significantly improves the segmentation result. For the segmentation method we use, we found that this preprocessing stage is necessary in order to produce an acceptable result.
Chapter 8

Conclusions and future work

8.1 Summary and Conclusions

Hepatic MRI image data are often used for surgical planning and for discriminating patients into suitable and not suitable for surgical treatment in the area of hepatic cancer resection. 3D visualisation of the patient's liver can help surgeons plan suitable treatment by making the location of diseased regions more apparent. The first stage of creating a 3D model for this purpose is to segment the target region from the tomographic data.

A brief literature survey on segmentation was presented in Chapter 2. However, most of the work presented in this survey was concerned with the segmentation of the liver from a CT dataset, which can be considered rather easier to segment compared with MRI data. From this review, we decided to develop an implementation of a user-steered segmentation based on the intelligent scissors algorithm, which has previously been used in real clinical pre-surgery planning, in order to segment the liver with minimal user interaction.

MRI images often suffer from bias field inhomogeneity which makes their segmentation based on intensity values, difficult. In addition to the field bias problem, soft tissue exhibits strong intra-organ variation, making organ segmentation based on edge detection complicated. Thus, in Chapter 3, several 2D methods for the preprocessing
stage are systematically evaluated using a real dataset and an optimised combination of methods and parameter values was chosen. From these experiments on an exemplar dataset, it was concluded that local enhancement, followed by median filtering and toboggan contrast enhancement, appeared the best preprocessing chain for MRI liver data in order to reduce the bias field artifact.

The adopted preprocessing chain was extended to 3D in Chapter 4. From this chapter, it was shown that 3D preprocessing performs better than 2D preprocessing in term of improving the performance of the segmentation using intelligent scissors. The experiment with interactive intelligent scissors, using a test dataset with 130 axial slices, demonstrated the segmentation of the liver from a 3D preprocessed dataset requires less segmentation time and fewer seed points compared with the segmentation of 2D preprocessed dataset. Thus, we decided to use 3D preprocessing for the experiments that follow.

Before we segment the liver, we applied another step which was designed to salient the edges on the basis of their length, curvature and proximity. The algorithm proposed by Li [93] imitates the working of V1, based on modelling the interactions among excitatory and inhibitory neurons. Implementation of this model was presented in Chapter 5. It was concluded that this model produces better results compared with a straightforward edge detection method, such as the Canny edge detector, because the produced saliency maps were less responsive to the inner structure of the organ.

In order to produce a closed contour defining the boundary of the hepatic region in each slice, we made use of a user-steered segmentation technique known as intelligent scissors [109]. One of the features of intelligent scissors is that the edge segments are estimated by computing the minimum edge cost, between two user-defined points, using a combination of edge magnitude and direction. However, the implementation of intelligent scissors in this study used the information of gradient magnitude feature, and the saliency map. This combination of costs is more robust to the false edges compared with the originally proposed cost function.

The segmentation process using intelligent scissors starts when the user selects one of the top most axial liver slices as the initial slice. In Chapter 6, a technique for
estimating the seed points in this initial slice was presented. This technique consists of thinning the edges, followed by the estimation of the liver "centre", and the projection of straight lines from this centre. We identify as seed points the first intersection these lines have with the edge fragments. It is shown that these seed points, when used with intelligent scissors, can produce a good reference liver contour. However, this technique is limited only to the slice where the edges of the liver are strong, and the shape of the liver resembles a circle or an ellipse.

We assume that the shape of the liver does not change dramatically in the next adjacent slice. Thus, in Chapter 6, we devised a method for propagating segmented contours to adjacent slices and concomitant refinement of these contours. It was shown that the segmentation in the axial direction only, produces inferior results using test data. Thus, we improved upon this segmentation by combining results from three orthogonal directions. In this experiment, three types of combination methods were investigated. The outcomes showed that the combination of axial-coronal results with axial-sagittal results provided the best results.

In Chapter 7, we segmented the liver from eight abdominal MRI datasets using the technique described above. The segmentation was run with all its parameter values fixed to be the same for all datasets. Six out of eight datasets gave a very close approximation to the liver boundary, indicating that this technique appears to be reliable for segmenting the liver, despite large variation in edge quality and contrast. We also demonstrated that the preprocessing chain defined in Chapter 4 can improve the segmentation result significantly.

8.2 Future work

The preprocessing routines used in the preprocessing chain may themselves be improved. Local enhancement is good for reducing the inhomogeneity in the image, but it increases the additive noise significantly. Median filtering reduces the additive noise, but at the same time, some information of the liver edges is lost. Toboggan contrast enhancement can re-enhance the edges back from the smoothed dataset, but this technique groups the liver parenchyma into smaller clusters, introducing step edges inside
this region which are sometimes strong enough to pull the edge segments of the intelligent scissors from following the desired contour. Thus, some other methods may be investigated to be added into the preprocessing chain, or to replace the existing preprocessing components. For example, some combinations of greyscale morphology operations might replace the median filtering and toboggan contrast enhancement, as this technique has some potential for reducing noise [100] and improving the edges [123].

In order to make the algorithm fully automatic, the initial slice for initiating intelligent scissors should be selected automatically, and not based on visual inspection. If this can be done, the segmentation technique would be 100 percent automatic. The selection of the initial slice should be among the top slices, where the image contains the right lung. In these images, the liver always resembles a circle or an ellipse, and has a very strong contrast with its surrounding because the lung is normally represented by very low intensity values. Thus, one possible automation strategy might be to identify the initial slice by analyzing the ratio of the area of the lung over the area of the abdominal cavity.

Introducing a deformable model into the segmentation track might further refine the liver surface. Deformable models such as the one used by Soler et al. [156] may be implemented. The external forces are calculated based on the gradient magnitude and the intensity of the voxel. However, we can also introduce the saliency map into this force. The result from intelligent scissors segmentation can provide a rough approximation to the shape and location of the initial model template. In order to strap the model from being attracted back to the high contrast blood vessels, or other organs in the dataset, the external force of the surface could be calculated for a few voxels away from the surface template only. As the model is then initiated to a very near true surface, we might expect this model to converge quickly, and thus avoid excessive extra computational time whilst producing a smoother 3D liver model with minimum segmentation error.

The segmented liver surface can be used to create a 3D liver model for presurgical planning. However, in an attempt to make this 3D visualisation more useful, segmentation
of the major blood vessels and the diseased areas is needed. In order to create additional landmarks that can give more understanding as to the nature inside the patient's abdominal cavity, segmentation of the spine and other organs, such as the kidneys and spleen, may also be carried out. As the performance of the liver segmentation using intelligent scissors might be affected by the strong edges from other organs, one probably will have to segment the tissue with the highest contrast first before segmenting the liver. Thus, unrelated edgels can be identified and extracted from the dataset.

The 3D models that have been created may then be exported to a virtual reality environment for presurgical planning, where advance manipulations of these liver models are permitted. Works by [38], [138] and [149], can be used as guidance.
Chapter 8. Conclusions and future work
Appendix A

Literature review on 3D visualisation techniques

3D visualisation of a patient's liver can help surgeons to understand the tumour's anatomy, and plan suitable treatment. This should bring improvement in surgical procedures [45, 138, 149]. This section describes the common reconstruction techniques used in 3D visualisation for medical data.

A.1 Maximum Intensity Projection (MIP)

MIP is the earliest method in volume visualisation. In MIP, for every ray of projection, the maximum intensity value in the volume data is projected to the viewing plane. Because the MIP technique is insensitive to the traversing direction, the ray can be either projected from front to back or from back to front. However, MIP cannot depict the overlapping vessels correctly, as large bright structures along the ray path can prevent projection of other structures from both directions [7, 144, 158].

A.2 Local Maximum Intensity Projection (LMIP)

Because MIP is limited in its depiction of overlapping vessels, LMIP has been introduced by Sato et al. [144]. The algorithm works by first selecting a threshold value.
This threshold value is selected based on the features which need to be displayed. Next, similarly to MIP, rays are projected into 3D data space. (However, these projections must start from the user’s viewpoint. Thus, LMIP is dependent on the ray traversing direction.) Then, the first local maximum encountered (see Figure A.1), which is greater than the preselected threshold, is projected to the viewing plane.

![Image profile](image_profile.png)

Figure A.1: Principle of LMIP. The profile represents the intensity variations of 3D data along the optical ray. Unlike the MIP procedure, which selects the global maximum intensity value, LMIP selects the first local maximum intensity that is larger than the threshold level. (Based on [144].)

![Rendered images](rendered_images.png)

Figure A.2: Rendered images using 3D data of CT angiography around a left kidney. (a) Volume-rendered image. Some of the blood vessels cannot be depicted clearly (arrows). (b) Shaded surface display image. Some of the blood vessels are clearly depicted (arrows), while some of them seem to be merged together (arrowhead). (c) MIP image showing high intensity bone structure occluding blood vessels under investigation. (d) LMIP image clearly showing blood vessels structure, discriminated from bone structure. (Taken from [144].)

Because the local threshold can suppress unwanted high intensity structures, LMIP can generate superior vessel images compared with other techniques (see Figure A.2). However, LMIP is still computationally expensive, and thus it cannot be used for real time visualisation. In addition, it is also difficult to find the optimal threshold value.
A.3 Volume Rendering

In volume rendering, each voxel in the 3D dataset is assigned an opacity value, $\alpha$, using a classification process. An $\alpha$ value of 1 stands for fully opaque and 0 for fully transparent. An $\alpha$ value between 0 and 1 is given to semitransparent voxels.

![Volume Rendering Diagram](Image)

Figure A.3: Ray is projected to the observer's eye. The ray needs to be resampled at each voxel. (Images taken from [122].)

Volume rendering usually uses a ray casting approach. In this approach, rays are cast into the observer's eye (see Figure A.3). The output ray colour at each voxel sample is calculated by using:

$$C_{out,\lambda}(U_i) = C_{in,\lambda}(U_i)(1 - \alpha(x_i)) + c_\lambda(x_i)\alpha(x_i)$$  \hspace{1cm} (A.1)

where $C_{out,\lambda}(U_i)$ is the outgoing light's colour, $C_{in,\lambda}(U_i)$ is the ingoing light's colour, $c_\lambda(x_i)$ and $\alpha(x_i)$ is the colour and the opacity of the voxel being sampled [122].

In volume rendering, the easiest way to display a surface is by assigning fully transparent values to voxels outside the objects of interest. An example is shown in Figure A.4.

Volume rendering is computationally expensive because this algorithm needs to accumulate all the voxels' data along each projection line [45, 138, 162]. Previously, volume rendering was considered as unsuitable for real time display, but more recently, computer hardware has become available to support this technique [91]. In addition, batch processing can also be used to overcome this problem [45]. But volume rendering is still sensitive to missing structure parts and opacity parameters [138, 144].
Appendix A. Literature review on 3D visualisation techniques

A.4 Surface Rendering

Organ models which are created using surface rendering techniques are often preferred by surgeons compared with models constructed by volume rendering techniques. Unlike the other techniques, surface rendering only contains the information about the contours of the objects, but it can provide sufficient geometric information about the location, size, and shape of the body structure involved [138]. Surface rendering is commonly used because there is a lot of hardware available to support this technique [138, 144].

Although the surface of the region of interest can be extracted using deformable surfaces, the marching cube algorithm is more popular. This algorithm works on three dimensional datasets, first by dividing the dataspace into cubes. Given some intensity threshold value, $T_s$, the algorithm then marches through this arrangement of cubes from front to back, top to bottom and left to right to find the isosurfaces corresponding to this threshold. During the propagation, each pair of connected corners in each cube is checked for the threshold cross-over [7].

Figure A.4: In volume rendering, an organ's surface can be selected by manipulating the voxels' opacity value. Top histograms show pixel distributions and the white plots show the opacity functions. Lower images show the result of volume rendering using these distributions and functions. (Images taken from [140])
A.4. Surface Rendering

As a cube has eight vertices, this means there are $2^8 = 256$ combinations of isosurfaces possible through each cube. An isosurface is represented by triangles. The 256 possible combinations can be reduced into only 15 combinations by considering symmetry (see Figure A.5). However, further special representations are needed to handle cases that correspond to multiple surfaces within the same cube [130].

![Figure A.5: 15 possibilities of an isosurface through a cube. Points denote the corners that exceed the threshold value. (Based on [130])](image)

The quality of the output 3D surfaces from this technique is related to the size of the cube units. Smoother surfaces can be obtained by dividing the 3D dataspace into smaller cubes. However, this will produce a large number of triangles, and, if there is noise in the image, this may produce false jagged surface segments. Although surface rendering can be supported by a hardware accelerator, the hardware can only compute and display a limited number of triangles (about 50,000 complex polygons) per frame in real-time display [138].

To reduce the number of triangles, a process known as *decimation* is needed. This process groups together neighbouring triangles which have some similar properties (based on surface normals), deletes them and replaces the resulting space with bigger triangle(s). The output of this process contains a minimum number of triangles, but the original topology of the object is maintained [104].
To produce more realistic 3D visualisation, the resulting surfaces from a surface rendering technique can be shaded. Some researchers, for example Neyret and Heiss [115], also use texture mapping to their 3D model to increase visual realism (see Figure A.6).

Figure A.6: To make the surface of 3D liver model more realistic, texture mapping can be applied. The triangles show the texture used in this model. (Taken from [115])
Appendix B

3D visualisation using marching cube algorithm

There are several common techniques used in three dimensional (3D) visualisation, namely surface rendering, volume rendering, shell rendering, maximum intensity projection (MIP), and local intensity projection (LMIP). In this project, a surface rendering technique has been implemented.

Surface rendering need a more complicated algorithm compared with MIP or LMIP, but this technique normally requires less computational time to completely render 3D objects. This is because unlike the other techniques, surface rendering only contains the information about the contours of the objects (which still can provide enough geometric information about the location, size and the body structure involved), yet only computation on the surfaces is needed. In addition, various hardware (e.g. graphic cards [117], etc.) and software libraries such as Open Graphic Library (OpenGL) [120] and Visualisation Tool-Kits (VTK) [171] are available nowadays to support this technique.

Although the surface of the region of interest can be extracted using a deformable surfaces model, the marching cube algorithm is more popular. And thus, the marching cube algorithm has been implemented in this work.
B.1 Methodology

The marching cube algorithm works with a three dimensional dataset [15, 130]. The algorithm begins by dividing the dataset into smaller cubes, as shown in Figure B.1. The size of the cubes is defined by the users, depending on how accurate or how fast they want the result to be. Larger size cubes will produce rough surfaces, but they need less processing time as the objects are only represented by a few triangles. Smaller size cubes will produce accurate surfaces, but more triangles are needed to represent the object, thus will increase processing time. An example of the effect of the cubes' size to the quality of the output image is shown in Figure B.2.

![Original dataset](image1.png)

![After sub-division](image2.png)

Figure B.1: The first step of marching cube algorithm is to subdivide the dataset into smaller cubes.

![Grid size: 10, 70 Faces](image3.png)

![Grid size: 5, 220 Faces](image4.png)

![Grid size: 2, 1700 Faces](image5.png)

![Grid size: 1, 6800 Faces](image6.png)

Figure B.2: When the smaller size of sub-cubes is used, more details of the object can be detected and displayed. However, the number of triangles needed to represent the object will also increase, and thus more processing time is required. (Taken from [15].)

This algorithm then will march through this arrangement of sub-cubes from front to back, top to bottom, and left to right. However, each sub-cube will be treated independently from the others. For easier understanding, each sub-cube can be considered as being represented by eight vertices and twelve edges. This is shown in Figure B.3.

Associated with each sub-cubes is a cube index. This index is represented by eight
bits, where each bit represents the condition of each vertex in that cube. The default value for each bit is equal to zero. When vertices with intensity values that represent the objects need to be displayed, the corresponding bits will be set to 1.

Figure B.3: Each sub-cube is represented by eight vertices (indicated by letters), and twelve edges (E1, E2, E3, ..., E12). The cube index is computed on the basis of the combinations of the intensity values of the vertices.

There are several ways to indicate whether a vertex represents the object or the background. Usually, unsegmented data are used as the input for the marching cube algorithm. The user needs to define a threshold value [15, 130] before beginning processing the dataset. Vertices which have intensity value greater than a pre-set threshold, will be assumed as lying within the object of interest. However, in this experiment, segmented images have been used. This condition has several advantages, as the input images are "cleaner" compared with normal image. In addition, it is also easier to differentiate between object and the background voxels. In this experiment, all background voxels have been set to the value 255. Thus, if the intensity value at a vertex is not equal to 255, the corresponding bit in the cube index will be set to 1.

Figure B.4: In this example, only vertex D lay in the object. Thus, the cube index is equal to 00001000 = 8. In the look-up table, it is indicated that edges E3, E4, and E12 are involved in surface calculation. Triangle vertices are computed along these vertices, and a triangle then is created.

Actually, this cube index is used to index the look-up table, which contains the infor-
information about the involved edges in triangle creation. By using the look-up table, it helps to speed up the processing time. The information about the edges is needed as the vertices of the triangles are assumed located along that edge. An example is shown in Figure B.4.

The location of the triangle vertices can be calculated by using linear interpolation [15]. The implementation by [130] locates the vertices of the triangles in the middle of the edges. However, in our implementation, as segmented images have been used, the vertices of the triangles are located in the exact location of the surface.

**B.2 Results and Discussion**

In this experiment, we use a segmented dataset as the input to the marching cube algorithm. The segmented data consist of forty five $256 \times 256$ axial slices, where the liver regions are surrounded by a white background with intensity 255. An example is shown in Figure B.5.

![Example of the input image](image.png)

(a) Unsegmented. (b) Segmented.

Figure B.5: An example of the input image. Segmented images, such as that shown in (b) have been used as input in this experiment.

Two different ways were applied to define the location of the vertices of the triangles. The first technique roughly estimates the surfaces of the desired object. This can be done by assuming that all the vertices of the triangles are located in the middle of the sub-cubes' edges. In the second technique, the actual position of the surfaces were used to locate those vertices. The first technique is faster, but the second technique produces smoother surfaces.
B.2. Results and Discussion

B.2.1 Results by roughly estimating the surfaces

The result produced by locating the vertices of the triangles in the middle of the sub-cube edges, is shown in Figure B.6.

Figure B.6: The image on the left shows the triangle mesh used to represent the liver. The image on the right shows the rendered image. In this case, the sub-cubes dimension is equal to $4 \times 4 \times 4$ voxels.

B.2.2 Results by using the actual location of the surfaces

Figure B.7 shows the result of a marching cubes algorithm by assigning the vertices of the triangles to their exact location on the liver surface. Although both Figure B.6 and Figure B.7 are represented by the same number of triangles, the liver surface shown in Figure B.7 appears smoother, and offers a better representation of the true surface.

Figure B.7: The image on the left shows the triangle mesh used to represent the liver. The image on the right shows the rendered image. In this case, the sub-cubes dimension is equal to $4 \times 4 \times 4$ voxels.

The example shown in Figure B.8 demonstrates why the liver surface shown in Figure B.7 has better quality compared with Figure B.6.
Appendix B. 3D visualisation using marching cube algorithm

B.2.3 The effects of cube size on the rendered model.

The dependence of the result on the cube size is shown in Figure B.10. The rendered object will show more details if smaller cube size is used. However, the display can be refreshed faster by using bigger cubes, as the model is represented by fewer triangles. Note that Figure B.10.(e) can be considered having sub-pixel accuracy. However, the rendered object does not have a smooth surface because the input dataset has rough edges. Thus, in this experiment, the results of using a cube size $8 \times 8 \times 8$ and $4 \times 4 \times 4$ are best. The surface seems smoother, and the rendered object can be manipulated faster, compared with the result by using cube size $2 \times 2 \times 2$.

Figure B.8: By considering a two dimensional example, it is shown that by locating the vertices of triangles in the exact location of the surfaces, the accuracy of the output image will be increased. Figure (c) shows that the result has almost the same surface as the input.

Figure B.9: The rendered object from the same input dataset, but by using a different cube size.
Figure B.10: Continued.
B.3 Conclusions

The marching cubes algorithm can be used to display three dimensional organs from medical image datasets. Although the organs are displayed as empty objects (i.e. only represented by their surfaces, and do not contain any volumetric information) the results can be considered acceptable, and easily understood. The surfaces can be rendered quickly (which is usually less than one minute on Pentium III 1GHz PC with 256 Mbytes of RAM and graphic hardware support), and thus be incorporated in an interactive program. The quality of the 3D output surfaces from the marching cubes algorithm technique is related to the size of the cubes. More accurate surfaces can be obtained by dividing the 3D dataset into smaller cubes.

However, in this experiment, in addition to the size of the cubes, the smoothness of the surface is also very dependent on the output of the segmentation. Precise segmentation outputs will lead to smoother and more accurate results. The performance speed of this algorithm is dependent on the number of triangles used to represent the object. Thus, in the future, an implementation of the algorithm which can represent the same objects with fewer triangles, such as a marching triangles algorithm, should be investigated. Another possibility is to apply decimation to the triangle meshes which are created from the marching cubes algorithm. Development of more accurate segmentation tools are also needed to improve the quality of resultant 3D image.
Appendix C

Calculation of the 3D gradient kernels

First, we assume that the voxels in our dataset have dimensions $X$, $Y$, and $Z$ along the $x$, $y$ and $z$ direction respectively. Then we define:

\[
\begin{align*}
A &= X(X^2 + Y^2 + Z^2)^{-0.5}, \\
B &= X(X^2 + Z^2)^{-0.5}, \\
C &= X(X^2 + Y^2)^{-0.5}, \\
D &= Y(X^2 + Y^2 + Z^2)^{-0.5}, \\
E &= Y(Y^2 + Z^2)^{-0.5}, \\
F &= Y(X^2 + Y^2)^{-0.5}, \\
G &= Z(X^2 + Y^2 + Z^2)^{-0.5}, \\
H &= Z(Y^2 + Z^2)^{-0.5}, \\
I &= Z(X^2 + Z^2)^{-0.5}
\end{align*}
\]

According to [184], the weighting of gradient $x$ kernel, $\omega_x(x, y, z)$, is given by:

\[
\omega_x(x, y, z) = \frac{x}{|r|} \tag{C.2}
\]

where $|r| = \sqrt{x^2 + y^2 + z^2}$. Similarly, $\omega_y(x, y, z) = y/|r|$ and $\omega_z(x, y, z) = z/|r|$. Thus, the $3 \times 3 \times 3$ gradient kernels (i.e. $\omega_x$, $\omega_y$, and $\omega_z$) may be represented by Figure C.1.

However,

\[
(4A + 2B + 2C + 1) \neq (4D + 2E + 2F + 1) \neq (4G + 2H + 2I + 1) \tag{C.3}
\]

when $X \neq Y \neq Z$. Thus, we need to normalize these kernels, i.e.

\[
\begin{align*}
\omega_x(x, y, z) &= \frac{\omega_x(x, y, z)}{(4A + 2B + 2C + 1)} \tag{C.4} \\
\omega_y(x, y, z) &= \frac{\omega_y(x, y, z)}{(4D + 2E + 2F + 1)} \tag{C.5} \\
\omega_z(x, y, z) &= \frac{\omega_z(x, y, z)}{(4G + 2H + 2I + 1)} \tag{C.6}
\end{align*}
\]

where $\omega_x$, $\omega_y$, and $\omega_z$ are the normalized version of $\omega'_x$, $\omega'_y$, and $\omega'_z$ respectively.
**Appendix C. Calculation of the 3D gradient kernels**

Figure C.1: 3D kernels used to calculate the gradient vector.
Appendix D

The division of the body mask in 3 equal volumes along an arbitrary direction

We have a vector, \( \vec{i} = (\delta v_x, \delta v_y, \delta v_z) \), which defines the direction of inhomogeneity in the dataset. We then create a plane which is perpendicular to this vector, as shown in Figure D.1(a).

![Diagram](image)

Figure D.1: (a) The plane is orthogonal to vector \( \vec{i} \). (b) The origin of axes \( O \), the foot, \( A \), of the normal vector on the plane from the origin and a random point \( B \) on the plane.

We introduce vectors \( OA \) and \( AB \) (see Figure D.1(b)) in order to workout the equation
of the plane in terms of the components of vector $\vec{i}$ and its parametric distance $k$ from the origin of the axes. Vector $\vec{OA}$ is vector $\vec{i}$ scaled by a constant $k$ (i.e. $\vec{OA} = k\vec{i} = (k\delta v_x, k\delta v_y, k\delta v_z)$). We assume that point $B$ lies on the plane at position $(x, y, z)$ and thus, vector $\vec{AB}$ also lies on the plane. This means that vectors $\vec{OA}$ and $\vec{AB}$ are orthogonal to each other. So,

$$\vec{OA}.\vec{AB} = 0$$

$$\begin{align*}
(k\delta v_x, k\delta v_y, k\delta v_z).((x - k\delta v_y, y - k\delta v_y, z - k\delta v_z) &= 0 \\
k\delta v_x(x - k\delta v_x) + k\delta v_y(y - k\delta v_y) + k\delta v_z(z - k\delta v_z) &= 0 \\
\delta v_x x + \delta v_y y + \delta v_z z - k(\delta v_y^2 + \delta v_y^2 + \delta v_z^2) &= 0 \quad \text{(D.1)}
\end{align*}$$

The location of the plane changes when the value of $k$ changes.

We create two parallel planes (i.e. Plane1 and Plane2). Plane1 has the $k$ value equal to $k_1$, and Plane2 has the $k$ value equal to $k_2$. We increase or decrease the value of $k_1$ and $k_2$ until Plane1 and Plane2 divide the body mask into three sub-volumes, where each sub-volume consists of about one third of the total number of voxels inside the body mask.

To count the number of voxels which make up each subvolume, we compute for each voxel $(x, y, z)$ functions

$$f_1(x, y, z) = \delta v_x x + \delta v_y y + \delta v_z z - k_1(\delta v_y^2 + \delta v_y^2 + \delta v_z^2)$$
$$f_2(x, y, z) = \delta v_x x + \delta v_y y + \delta v_z z - k_2(\delta v_y^2 + \delta v_y^2 + \delta v_z^2) \quad \text{(D.2)}$$

The voxels which make up each subvolume satisfy the following criteria:

- **Subvolume 1**: $f_1(x, y, z) < 0$ and $f_2(x, y, z) < 0$
- **Subvolume 2**: $f_1(x, y, z) > 0$ and $f_2(x, y, z) < 0$
- **Subvolume 3**: $f_1(x, y, z) > 0$ and $f_2(x, y, z) > 0$
Bibliography


[66] Surgical Planning Laboratory Brigham & Women's Hospital. Case of the moth: May 1999, 3D visualization for planning of liver cryosurgery [online]. Available
[Accessed on 01 October 2001].


[68] irishhealth.com. Liver cancer (hepatic cancer) [online]. Available from:


the active watershed method. In *Proceedings of Medical Image Understanding and Analysis (MIUA)*, 2002.


[162] S. Takahashi, Y. Unemura, T. Kurosaki, A. Uchiyama, and N. Suzuki. Support system for hepatectomy allowing interactive segmentation directly on three-


