Abstract

Deriving liver vessel structure from CT scans manually is time consuming and error prone. An automatic procedure that could help the radiologist in her analysis is therefore needed. We present two algorithms to preprocess and segment the hepatic vessels. The first algorithm processes each CT slice individually, while the second algorithm applies processing on the whole CT scan at once. Matched filtering and anisotropic diffusion is used to emphasise the blood vessels, and entropy based thresholding and segmentation by local mean and variance are used to coarsely position the vessels. Node positions and sizes are derived from the skeleton and the distance transform of the segmentation results, respectively. From the skeleton and node data, interconnections are added forming a vessel graph. At the end, a search is executed to find the most likely vessel graph based on anatomical knowledge. Results have been inspected visually by medical staff and are promising with respect to clinical use in the future.

Acknowledgements

I would like to thank my advisor Lars Aurdal for his guidance throughout this project. I am also grateful for all help given by my co-advisor Keith Downing.

Eigil Samseth, Håkon Heuch, and Tom Mala from the Interventional Centre at the University of Oslo have provided CT and MR datasets as well as guidance in this project. I am very grateful for their support. Also, I would like to thank Rune Bakken for his help.

Special thanks to Birgit Ødegaard for her patience and encouragement during this period.

Contents

Chapter 1

Introduction

The liver is a vital organ with vascular, metabolic, secretory, and excretory functions. It is extensively perfused and during liver surgery special care has to be taken in order to avoid bleedings.

Prior to liver surgery, the patient will typically be examined using CT scans, in particular the position of large hepatic vessels must be determined. The relative position of, for instance, tumours to these vessels is of great importance when planning the procedure and in evaluating the operability of the patient.

Surgeons and radiologists will typically base their evaluation on a visual inspection of the 2D slices produced by a CT scan. It is difficult, however, to deduce a detailed liver vessel structures from such images. Surgeons at the Intervention Centre at Rikshospitalet have found 3D renderings of the liver and its internal vessel structure to be a valuable aid in this complex evaluation phase. Currently, these renderings are based on a largely manual segmentation of the liver vessels. This procedure is time consuming and error prone, and we have sought a way to extract the liver vessel structure automatically from CT scans.

MR scan results are similar to that of CT scans, but contain typically different tissue contrasts. Although the methods we will derive can be applied to MR scans as well, CT scans will be our primary focus in this thesis.

1.1 Problem description

The task is to create a presentation of the hepatic vessels in 3D. The application need to be interactive such that the liver and its vessels can be rotated and inspected from any angle. From an image processing point of view, we need to derive techniques to first emphasise the hepatic vessels, and then methods to create a sound vessel segmentation. From this, a vessel graph should be computed, and anatomical knowledge be used to create a likely vessel representation.

The visualisation of the resulting vessel graph should be presented in real-time, and be highly realistic. In order to take advantage of inexpensive modern graphics cards, a triangle mesh with realistic branching structures should be computed.

1.2 Document structure

Previous work on medical image analysis will be presented in the next chapter. After this, the method chapter will present the two vessel reconstruction algorithms. The final section of this chapter will also present the visualisation technique used. Results and discussion follow next, presenting the outputs of each separate image analysis method and the final visualisation results. Two separate techniques have been developed, and we discuss the results at the end of each technique's section. Finally, conclusions and further work will be outlined in chapter 5.

Chapter 2

Previous work

Medical images have been of particular interest to the image processing community. Several methods have been developed to extract valuable information from CT and MR images in particular. While there exist many theoretical approaches, very few of them have been implemented and used in practice. This is understandable due to the fact that great accuracy is needed when concerning human lives, but as the methods and results from the image processing community evolve, the acceptance of these methods increase.

In this chapter we will primarily describe algorithms previously used in medical imaging. We will first go through a selection of preprocessing/postprocessing algorithms. Segmentation algorithms will be covered next, and finally we will look at a few representation and description techniques.

The reason why we have chosen these three particular disciplines in image analysis is closely related to the task at hand. In order to extract the hepatic vessels from CT or MR images, preprocessing and segmentation techniques are required. The segmentation results may require postprocessing before an adequate description of the results can be made. This description will be further used to construct a likely 3D vessel tree based on anatomical knowledge.

2.1 Preprocessing

A preprocessing stage is usually done before executing a segmentation algorithm. Generally, the motive is to make interesting regions more readable by the segmentation algorithms. After the segmentation, corrections of the results are sometimes needed in a stage typically called the postprocessing stage. These corrections are often performed using the same techniques that are used in the preprocessing stage.

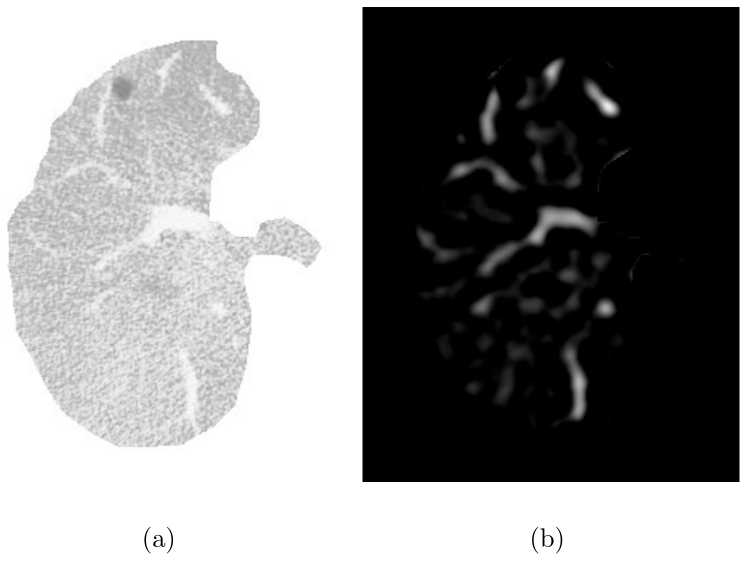


Figure 2.1: Example matched filtered result where the blood vessels in an liver are emphasised. a) Original CT slice masked with a liver mask. b) Matched filtered result of a) using several templates matching the different sized blood vessels in separate headings.

2.1.1 Spatial filters

Spatial filtering through convolution is commonly used in image preprocessing (Gonzalez and Woods, 2002; Sonka et al., 1999). A frequent application is to remove noise by smoothing or blurring an image. Another common use is called matching, which is basically fitting a template to an image (Gonzalez and Woods, 2002). Regions where the template fits are marked bright, while areas that doesn't match will become dark.

Matched filters

Matched filtering were proposed used in medical imaging by Chaudhuri et al. (1989). The idea is that blood vessels in CT images have a Gaussian profile, and thus a Gaussian hill filter can be used to match the blood vessels. Since blood vessels have distinct sizes and are headed in separate directions, the filter has to be scaled and rotated. Filtering results from all the scaled and rotated Gaussian hill templates are summed up. Omholt-Jensen (2002) used matched filters in segmentation of liver vessels. Figure 2.1 shows an example result of matched filtering applied to a CT slice.

2.1.2 Morphological operators

Most morphological operators are based on dilation and erosion (Gonzalez and Woods, 2002; Sonka et al., 1999; Soille, 2003). A structure element is used to define which neighbouring pixels should be included during filtering. Using a flat structure element (Soille, 2003), dilation and erosion are local maxima and minima filters, respectively.

These two basic operators can prove powerful in preprocessing and post-processing stages. Some example uses are region filling, matching through the hit-or-miss transform, boundary extraction, finding object skeletons, and pruning unimportant pixels. Also regarded as a morphological operator is the distance transform, which is used to compute the distances to the nearest background pixels in an image.

In medical imaging, Aykac et al. (2003) makes use of morphological closing and erosion in a preprocessing step to identify candidate airway locations. Further, morphological opening, closing, and skeleton (thinning followed by pruning) are used by Thomas et al. (1991) to measure the fetal femur length in ultrasound images. The length is then compared to a growth chart to calculate the age of an unborn child.

2.1.3 Bayesian image processing

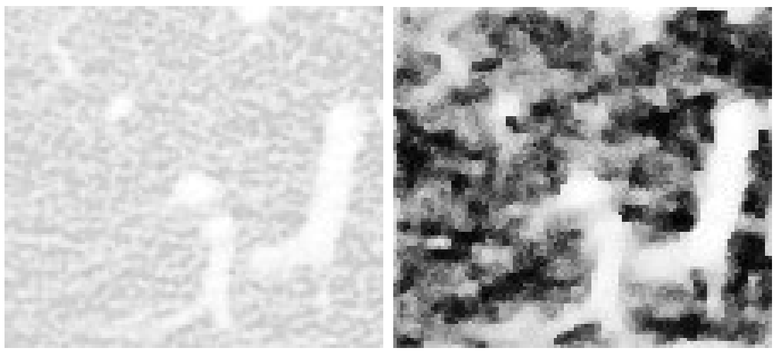
Bayesian image restoration was introduced by Geman and Geman (1984). The idea was to make use of the Bayes formula (Duda et al., 2001) to classify pixels:

$$P(X|Y) = \frac{p(Y|X)P(X)}{P(Y)} \tag{2.1}$$

In this formula, $P(Y|X)$ is called the likelihood distribution, which specify how X has been degraded from Y. The last part of (2.1), $P(X)$, is the prior probability that defines how neighbouring pixels are related in X. Figure 2.2 shows an example of vessel restoration that we computed while testing various preprocessing algorithms.

This technique can be used to reconstruct images in a preprocessing stage as well as for segmentation purposes. The unknown reconstructed image is then corresponding to X, and the original image is set to be Y. The pixels in the new image is updated iteratively after the likelihood distribution and the prior probability have been defined.

Energy minimising methods such as simulated annealing is ordinarily used to find the optimal estimation of image X. Refer to Winkler (1995); Geman and Geman (1984); Hokland (2002) for further reading.



(a) (b)

Figure 2.2: a) A part of the liver. b) Result of restoration using Bayesian image analysis.

2.1.4 Level set methods

The level set method introduced in Sethian (1997, 1996), is primarily a model based segmentation algorithm and will be more fully described in section 2.2.2. This approach, however, can also be used in noise removal as proposed by Malladi and Sethian (1996). Suppose we look at the intensity of the image pixels as heights in a topographic map, then by applying motion by curvature, small contours will disappear over time as seen in Sethian (2004b). An example is preprocessing of a digital subtraction angiogram (DSA) as shown in figure 2.3.

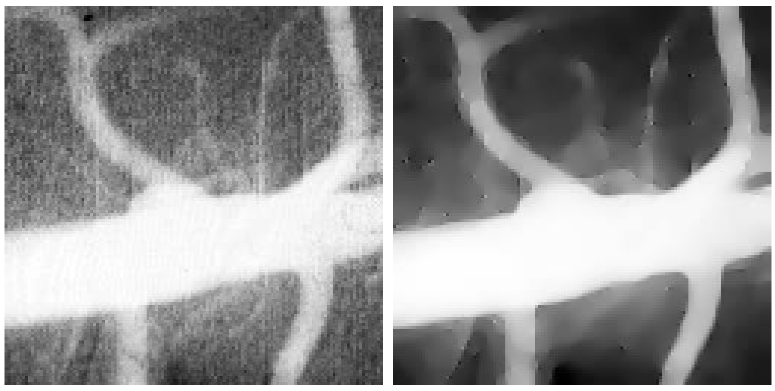
2.1.5 Anisotropic diffusion

The use of anisotropic diffusion in image processing was introduced by Perona and Malik (1990), and the technique has been frequently used in image analysis ever since. The anisotropic diffusion equation is as follows:

$$\frac{\partial I}{\partial t} = \nabla(c\nabla I) \tag{2.2}$$

where the diffusion coefficient c varies in space but not time. If c is instead constant, the equation is reduced to the isotropic heat diffusion equation:

$$\frac{\partial I}{\partial t} = c\nabla^2 I \tag{2.3}$$



(a) (b)

Figure 2.3: a) A digital subtraction angiogram of an artery. b) Preprocessing result of a) using motion by curvature. Images courtesy of Sethian (2004b)

Applying isotropic heat diffusion on an image is equivalent to running a Gaussian filter on the image. By using anisotropic diffusion with varying c it is possible to specify the magnitude of blurring with respect to the contents of the image.

Typically during image preprocessing, we want to blur roughly homogeneous regions, while preserving the edges. This can for instance be achieved in anisotropic diffusion by setting $c = g(||\nabla I(x, y, t)||)$, and thus vary c with respect to the edges in an image. Additionally, if the function g is chosen properly, the edges in an image can be sharpened as well (Perona and Malik, 1990).

There are several example uses of anisotropic diffusion in medical imaging. For instance, Soler et al. (2001) uses anisotropic diffusion in a preprocessing phase before segmentation of the liver, blood vessels, and possible liver tumours from CT scans. Another example is Chung and Sapiro (2000), where anisotropic diffusion is used before segmenting skin lesions.

Anisotropic diffusion is also used in the gradient vector flow procedure (Xu and Prince, 1998, 2000), which is typically used before applying a deformable model segmentation (See section 2.2.2).

2.2 Segmentation

Image segmentation is typically referred to as the process of finding regions in an image that have one or more common properties. The common properties may for instance be intensity values, local mean and variance, texture, and even shape.

Generally, we group segmentation techniques into two main groups, local and global approaches. Local approaches are exclusively based on information contained in the image itself. The image is assumed to be "self-contained", i.e. it has all the information necessary to retrieve the objects of interest. On the other hand, global methods utilise related knowledge about the image in the segmentation approach.

2.2.1 Local approaches

In this section we will cover some of the segmentation algorithms based on local knowledge in a dataset.

Thresholding

Thresholding is the simplest and most frequently used segmentation algorithm. The basic idea is to mark pixels having intensity values within a predetermined range (Gonzalez and Woods, 2002; Sonka et al., 1999). A slightly more advanced usage of this algorithm is described in Székely and Gerig (2000). Here, a two dimensional intensity distribution from a spin-echo MR image pair is computed, and the two distributions are used to decide which pixels should be segmented. A more accurate segmentation is usually accomplished using more than one spectrum of the same scene (for instance colour images).

The main challenge concerning thresholding is to select the most desirable intensity range. More often than not this is a nontrivial problem. One way to solve this is through an approach called optimal thresholding (Gonzalez and Woods, 2002; Sonka et al., 1999). The optimal threshold is said to be the threshold that causes the smallest number of pixels to be incorrectly segmented. In optimal thresholding, Gaussian curves are fitted on the histogram of an image, and thresholds are set where the curves cross. Soler et al. (2001) makes use of this method in segmenting CT images.

Another significant thresholding algorithm is based on the entropy of an image's histogram (Kapur et al., 1985). An entropy diagram is obtained by an average entropy measure (Gonzalez and Woods, 2002). Each local maximum on the entropy diagram represents a potential best threshold. This method

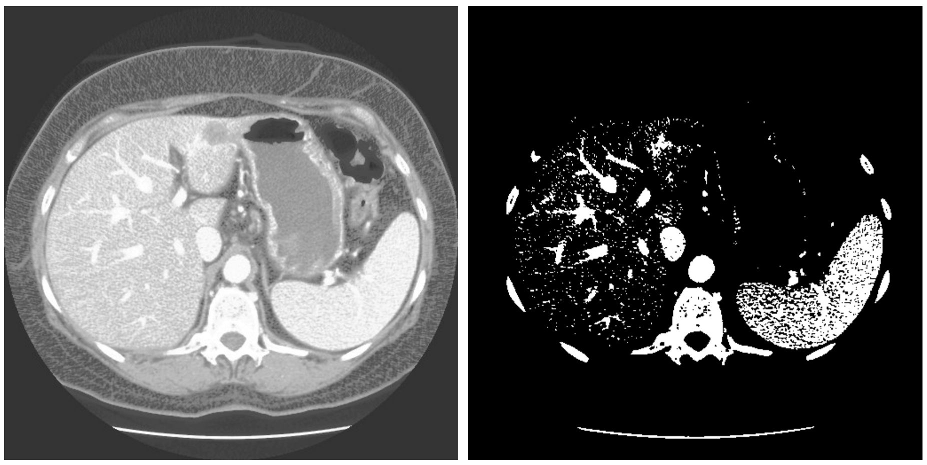


Figure 2.4: a) A CT image. b) Resulting segmentation using thresholding based on entropy.

is reported to be successful in segmenting liver vessels in Glombitza et al. (1999), and Omholt-Jensen (2002). See figure 2.4 for an example use of entropy based thresholding.

Mean shift segmentation

Mean shift segmentation is an interesting, recently proposed segmentation technique. Here, pixels in an image are instead represented as points in a feature space. Density estimation in the feature space is processed using the Parzen window technique described in Comaniciu and Meer (2002); Duda et al. (2001), and a mean shift procedure is used to follow the density gradient in the feature space to a local maximum (Comaniciu and Meer, 2002).

These maxima represents segmentation identities, and all pixels leading to the same local maximum is grouped into the same segment. The number of maxima is dependant on the Parzen window size, and thus the number of segments does not need to be known beforehand.

Although not mentioned in section 2.1, the mean shift procedure can be used in image preprocessing as well (Comaniciu and Meer, 2002; Fernández et al., 2003).

Edge based segmentation

Edge based segmentation involves following edges in an image and mark pixels within closed boundaries. The simplest approach is to analyse the immediate neighbourhood of each pixel in an image, and label pixels having similar gradient magnitude and direction (Gonzalez and Woods, 2002). Closed contours, containing equally labelled pixels, are eventually filled and transformed into segmented regions.

The procedure above, however, consider only local adjacent pixels when tracking the edges in an image. Instead, graph theoretic techniques (Gonzalez and Woods, 2002; Sonka et al., 1999) can be utilised to find the least expensive path according to a given cost function. Dynamic programming (Bellman, 1957; Sonka et al., 1999) can be used to find the global minimum, and thereby the most appropriate edge graph from a given starting point.

Another example of an edge based segmentation algorithm is the Hough transform (Hough, 1959; Gonzalez and Woods, 2002). Using various formulas, it is possible to search an image for simple geometric shapes. For instance, to search an image for circles with radius 1 we calculate the following Hough transform:

$$H(a,b) = \int\int f(x,y)\delta((x-a)^2 + (y-b)^2 - 1)dxdy \tag{2.4}$$

where $f(x,y)$ represents the gradient of an image, and (a,b) corresponds to the positions of the sought circles. The commonly used delta function, $\delta(u)$, returns 1 when u is 0, and 0 otherwise. High values in $H(a,b)$ represents positions (a,b) where circles are found in $f(x,y)$.

An elliptical Hough transform was used to identify axon centre in Fok et al. (1996). The purpose of this paper was to count the number of axons in nerve cells as well as extract each axon's size and shape. After the initial identification of the axon centre, an active contour model was used to refine the axon contours.

Region based segmentation

A second subset of traditional segmentation techniques is region-based techniques. These procedures rely on common properties between adjacent pixels as previously mentioned. Starting with a few seed points, regions are typically expanded until a property criteria is no longer met, or until a region boundary collides with another region's boundary. In addition to the traditional region growing scheme (Gonzalez and Woods, 2002; Sonka et al., 1999), the watershed segmentation (Soille, 2003), shown in figure 2.5, represents a

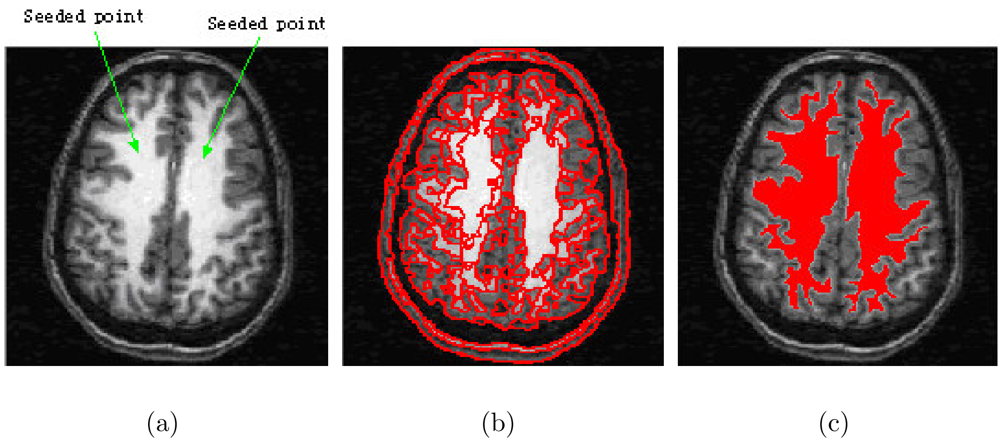


Figure 2.5: Segmentation of the brain white matter. a) Seed points. b) Resulting oversegmentation using watershed. c) A level set method, which will be described in section 2.2.2, is used to make the final segmentation using the watershed results from b). Images courtesy of MIPG medical image processing group.

similar technique. In Haris et al. (1999), the watershed transform was used to segment the coronary arterial tree.

Example applications are region growing (Martinez-Perez et al., 1999) and 3D region growing (Tuduki et al., 2000) used to segment blood vessels in medical images. In (Krivanek and Sonka, 1998), the watershed segmentation is used to automatically measure the size and shape of follicles from ultrasound images.

Additionally, several articles propose region based methods for delineating the liver vessels automatically through region based segmentation algorithms (Chaudhuri et al., 1989; Kapur et al., 1985; Inaoka et al., 1992; Zahlten et al., 1995; Soler et al., 2001). Most promising, with respect to creating a 3D model of the liver vessel structure, are the methods by Zalthen et al. and Soler et al.

Zalthen et al. use a voxel based region-growing-algorithm to extract the portal vein, but the algorithm requires an initial seed point and is therefore not fully automatic. In the article by Soler et al. however, the portal trunk is located using its general anatomical position. The portal vein skeleton is calculated utilising methods from Bertrand and Malandain (1994); Malandain et al. (1993), and is corrected by pruning vessel segments that do not confirm with a set of predetermined properties.

2.2.2 Global approaches

In this section we will present global segmentation methods that utilise additional knowledge than that contained in the image itself.

Deformable models

The first article on this topic was Kass et al. (1988). Here a model called *snake* was introduced. In short, a snake is a spline influenced by internal and external forces seeking an energy minima. Internal forces typically controls the tension and rigidity of the snake, and external forces draws the snake towards edges of an image (McInerney and Terzopoulos, 2000; Heuch, 2003). External forces from the image $I(x, y)$ is usually derived from:

$$P(x, y) = -c|\nabla[G_\sigma * I(x, y)]| \tag{2.5}$$

where c is the magnitude of the force, ∇ is the gradient, and G_σ is a Gaussian smoothing filter. $G_\sigma * I(x, y)$ means that $I(x, y)$ is convolved with G_σ. External forces can also include the so-called balloon force that expands the snake to find far edges. Another way of pulling the snake to remote edges is to implement gradient vector flow (Xu and Prince, 1998, 2000) that was briefly mentioned in section 2.1.5.

An interesting alternative to the original snake model is the discrete dynamic contour model (Lobregt and Viergever, 1995). The structure of this model is a set of connected vertices. Instead of a computationally expensive numeric solution used in the traditional snake model, these vertices are manipulated by simple vector operations. We implemented this model using balloon forces to segment the liver from CT images. Figure 2.6 shows a typical execution of this algorithm.

Snakes are becoming more and more common in medical imaging. For instance Heuch (2003) uses snakes to segment the liver from CT images, and Kelemen et al. (1998); Kelemen and Székely (1999); Székely and Gerig (2000) utilise snakes to segment the basal gonglia of the human brain. Moreover, snakes with a modified gradient vector flow were used to track white blood cells in citetray02.

Level set methods

Deformable models are difficult to handle when the topology of the contour changes (Sethian, 1997). The level set method (Sethian, 1997, 1996) solves this problem by viewing the contour as a graph instead of a parameterised curve, and by adding an additional dimension, time, to the interface. If we

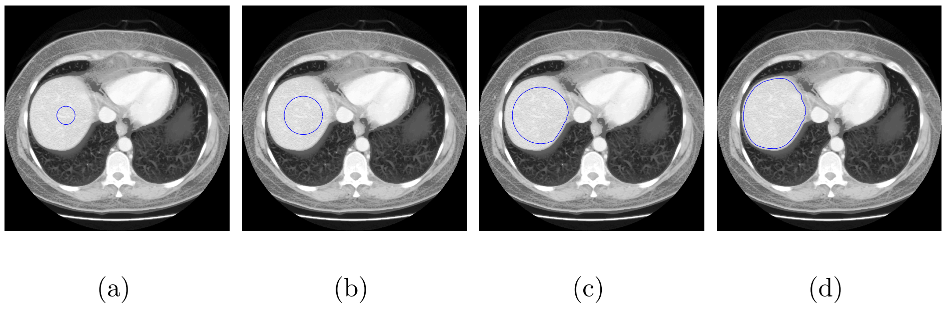


Figure 2.6: a) Initialisation of the contour model, in blue, on a CT image. b), c), d) The contour alters shape to fit the liver .

consider the third dimension as height instead of time, we can represent the model as a surface. Then, by setting the contour at time z equal to the contour at hight z, we can easily represent a changing contour. Merging contours can accordingly be easily formulated.

Level set methods have been demonstrated useful in outlining the stomach from CT images, and in segmenting the structure of arterial trees from DSA images (Malladi et al., 1995; Malladi and Sethian, 1996). Examples of beating heart segmentation, femurs and surrounding soft tissue segmentation, and brain reconstruction can in addition be found in Sethian (2004a).

Statistical models

Unlike the active contour models and the level set method, statistical models represents the general shape of the sought object. In order to build a statistical model, manually segmented training samples are needed. This makes implementing statistical model algorithms time-consuming and resource demanding. However, many segmentation tasks require statistical models in order to be adequately solved. In such cases, models such as snakes will more often than not result in an incorrect segmentation.

In applications, statistical models have two common uses. First, they can be used as a segmentation initialisation by matching reconstructed models onto structures in an image. After this, the segmentation is refined using another segmentation method, for instance active contours. Székely and Gerig (2000); Kelemen et al. (1998); Kelemen and Székely (1999) use principal component analysis (Duda et al., 2001) to create a small set of statistical models, eigenmodes, expressing the main variations of a larger training set. The eigenmodes are built from the eigenvectors with the largest corresponding

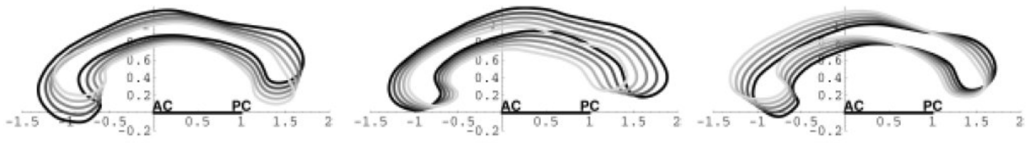


Figure 2.7: Statistical models built by principal component analysis. The figures show the models reconstructed from the three eigenvectors with the highest corresponding eigenvalues. Images courtesy of Székely and Gerig (2000).

eigenvalues.

The second use of statistical models are in criteria functions to compute the statistical validity of a segmentation result. In (Sclaroff and Liu, 2001), a multidimensional unimodal Gaussian distribution of the training set is assumed. Deviation from the mean is penalised by an amount that is proportional to the Gaussian distribution function.

Example uses of statistical models are Székely and Gerig (2000); Kelemen et al. (1998); Kelemen and Székely (1999) where 2D and 3D models of the corpus callosum are used (see figure 2.7). In Soler et al. (2001), a statistical model of lesions were defined to locate tumours in the liver. Furthermore, statistical models were used to automatically segment the three main structures of the heart from MR scans in Frangi et al. (2002).

Bayesian image segmentation

In section 2.1.3, we described methods that could be used to restore lost information in images.

Similar methods are proposed in medical image segmentation (Choi et al., 1991). First, a manual segmentation and classification is conducted. Afterwards, Gaussian curves are fitted to the probability distributions for each class. From these Gaussian curves, the mean μ and variance σ is measured. Automatic segmentation can then be executed. For each pixel, a probability $P(X|Y)$ is calculated for each class, and typically through inversion a class is selected for the corresponding pixel. Inversion is explained in Ripley (1987).

2.3 Shape representation and description

In image recognition, classification of the segmented shapes in an image is essential. Before the shapes can be classified, however, they need to be described in an appropriate numeric feature vector for the classifier.

Shape description has other uses as well. Since it transforms the segmented shapes into more useful representations, the results can be more easily processed and analysed. In this section, we will outline such representation and description techniques with emphasis on shape skeletons.

2.3.1 Contour based representation and description

Shapes can be described by their boundary. Various features can be used such as chain codes, boundary length, curvature, bending energy, signature, and chord distribution (Sonka et al., 1999; Gonzalez and Woods, 2002).

2.3.2 Shape invariants

The shapes of objects in an image may change depending on viewpoint. This problem may be overcome by shape invariants, which represent geometric configurations that remain unchanged under an appropriate class of transforms (Sonka et al., 1999).

One simple example of an invariant feature is cross ratio (Sonka et al., 1999). Cross ratio is a projectively invariant image feature, meaning that the feature is not altered by a projection transform. It is based on the fact that a straight line is always projected as a straight line, and thus four points on a line will have the following ratio both before and after a projection:

$$I = \frac{(A-C)(B-D)}{(A-D)(B-C)} \tag{2.6}$$

where A, B, C, and D are sequential points on a straight line.

2.3.3 Shape based representation and description

There are numerous ways to describe the region itself. Some example descriptions from Sonka et al. (1999) are shape area, Euler's number, projections, eccentricity, elongatedness, rectangularity, shape direction, compactness, and convex hull.

Another important region descriptor is region moments (Papoulis, 1991). Here, the image function is interpreted as a probability density of a 2D random variable. Properties of this random variable can be described using statistical characteristics, namely moments. Different degrees of invariance can be achieved depending on which moments are used. Scaled central moments, for instance, are translation and scale invariant.

2.3.4 Shape skeleton

A shape skeleton represents a minimal representation of a shape without changing its topology. In 2D, the skeleton is typically defined as the medial axis of a shape. Blum (1967) proposed a medial axis transform, where each pixel in a shape is marked as a medial axis if the pixels have two or more smallest distances to the background. An extension of this definition to 3D is possible, however, surfaces instead of lines may then be characterised as medial axes. Thus, in 3D, it is common to distinguish 3D skeletons into medial surfaces and medial lines.

2D skeletons

In this subsection we will look at algorithms used to create 2D skeletons. We briefly mentioned skeletonisation through morphological thinning in subsection 2.1.2. In morphological thinning, hit-and-miss templates are first used to find shape boundaries (Soille, 2003). The boundaries are removed successively until only the skeleton of the shape remains. Postprocessing of the skeleton may consist of a pruning phase where small end branches of the skeleton is removed.

Another example of an iterative skeleton algorithm is described in Guo and Hall (1989). This algorithm produces a minimal skeleton that does not need pruning. Unlike many skeleton methods, this technique does not have a set of templates to match the neighbourhood of each pixel. Instead, a set of criteria functions have to be fulfilled before a pixel is marked for deletion.

Iterative skeleton algorithms can often be parallelised, and specialised hardware may increase the speed of these algorithms. Refer to Lam et al. (1992) for further reading on iterative 2D skeleton algorithms.

Non-iterative skeleton algorithms have also been derived. These are typically based on medial axis and distance transforms (for instance Blum (1967)), or line following and run length encoding (Lam et al., 1992).

Figure 2.8 shows a rectangle and its resulting skeletons using thinning by morphology, and the algorithm given in Guo and Hall (1989).

3D skeletons

Existing templates or criteria functions used in many 2D skeleton algorithms cannot be directly extended into 3D. However, much research effort has been put into 3D skeletons recently due to the increased availability of 3D image modalities. Lobreget et al. (1980) first presented a skeleton algorithm based on preservation of Euler characteristics. Further, Ma and Sonka (1996) proposed a boundary thinning algorithm utilising deleting templates, and Saha

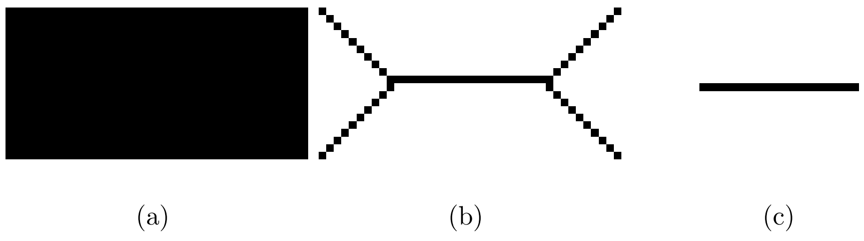


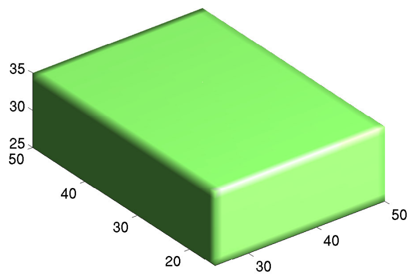
Figure 2.8: The skeleton of a rectangle using two different algorithms. a) The rectangle to be thinned. b) Skeleton of a) using morphological thinning. c) Skeleton of a) using the algorithm given in Guo and Hall (1989).

et al. (1997) developed a thinning algorithm that preserves the number of object components, cavities, and tunnels. More recent example publications of 3D thinning algorithms can be found in Borgefors et al. (1999); Palagyi et al. (2001); Xie et al. (2003).

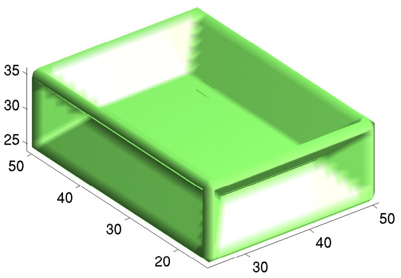
Palagyi et al. (2001); Xie et al. (2003) both make use of a simple point definition derived in Malandain and Bertrand (1992). A point is simple if its removal does not affect the topology of the shape. This simple point characterisation is mathematically sound, but can only be used to compute 6-connected skeletons. The simple point definition is insufficient, however, to create a sound thinning algorithm. The thinning algorithm also depends on the deletion order of the simple points. Palagyi et al. (2001), for instance, proposes a method that mark all pixels for deletion in a separate pass before deleting them. This procedure is repeated for each heading; northern boundary pixels are deleted first, then southern, etc. Figure 2.9 shows an example medial surface and medial line of a cube. The medial line is produced using the simple point definition in Malandain and Bertrand (1992), and by thinning boundary pixels a predetermined number of iterations.

2D and 3D skeletons in medical applications

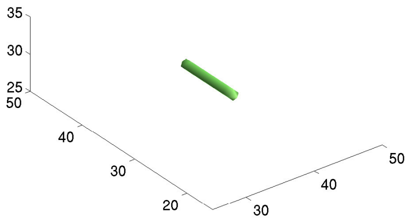
2D and 3D skeletons have been used extensively in medical image analysis. In Yim et al. (2000), vessel skeletons are used to analyse vessel paths and branching patterns of vascular trees from magnetic resonance angiography (MRA). Similar operations were performed in Palagyi et al. (2001), but from Spiral Computed Tomography (S-CT) volumes. More recently, Volkau et al. (2005) uses 3D shape skeletons to construct the human normal cerebral arterial system from various 3D datasets. Other examples include Nyström and Smedby (2001); Tom et al. (1994); Gomberg et al. (2000).



(a)



(b)



(c)

Figure 2.9: This figure shows the medial surface and an example medial line of a cube. a) The cube before 3D thinning. b) Medial surface of a). c) Medial line of a) using the simple point definition described in Malandain and Bertrand (1992).

Chapter 3

Methods

This chapter is separated into three sections. We will first describe two distinct techniques that have been derived to construct liver vessels from CT images. Both techniques are graph based, where an initial graph is first constructed through a set of preprocessing, segmentation, and graph representation techniques. After this, cost functions representing anatomical knowledge is used to improve the vessel graph.

A prerequisite to both techniques is that the liver is segmented beforehand. This is presently an unsolved problem, which will not be focused on in this thesis. In our work, the liver was segmented partially by hand through the application implemented in Heuch (2003).

The third section outlines the mesh generation algorithm used to visualise the vessel graph. The derived mesh generation method creates natural looking vessels and vessel branches that improves the presentation of the final results.

3.1 2D vessel reconstruction

In this section, a continuation of the work in Omholt-Jensen (2002) is presented where each CT slice is processed separately in the preprocessing and segmentation stages. After segmentation, a graph is constructed and improved through a global search mechanism.

New algorithms have been derived that lead to better segmentation results, a more improved vessel graph initialisation, and more appropriate cost functions for the global search method. Parts of this work have been published previously in Eidheim et al. (2004a) and Eidheim et al. (2004b).

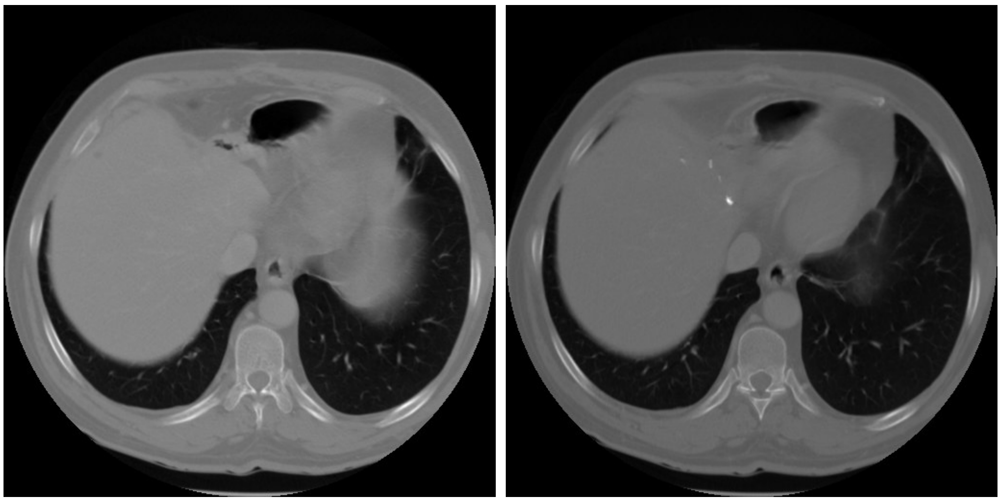


Figure 3.1: Two CT images before histogram equalisation.

3.1.1 Preprocessing

During the preprocessing step, the blood vessels are emphasised through the set of methods outlined in this subsection. The results are next processed by the segmentation algorithms described in subsection 3.1.2.

Histogram equalisation

CT sequences may contain images with slightly different gray value contrast. An example is shown in figure 3.1. This may represent a problem when processing the images with equal parameter choices.

We chose to normalise the gray value contrast through a method called histogram equalisation (Gonzalez and Woods, 2002). The new image is obtained from:

$$s_k = \sum_{j=0}^{k} \frac{n_j}{n} \qquad k = 0, 1, 2, ..., L-1 \tag{3.1}$$

where s_k is the new pixel intensity for pixel intensity k, L is the number of possible gray levels, n is the number of pixels in the image, and n_j is the number of pixels having intensity j. Figure 3.2 shows the result of histogram equalisation of the images in figure 3.1.

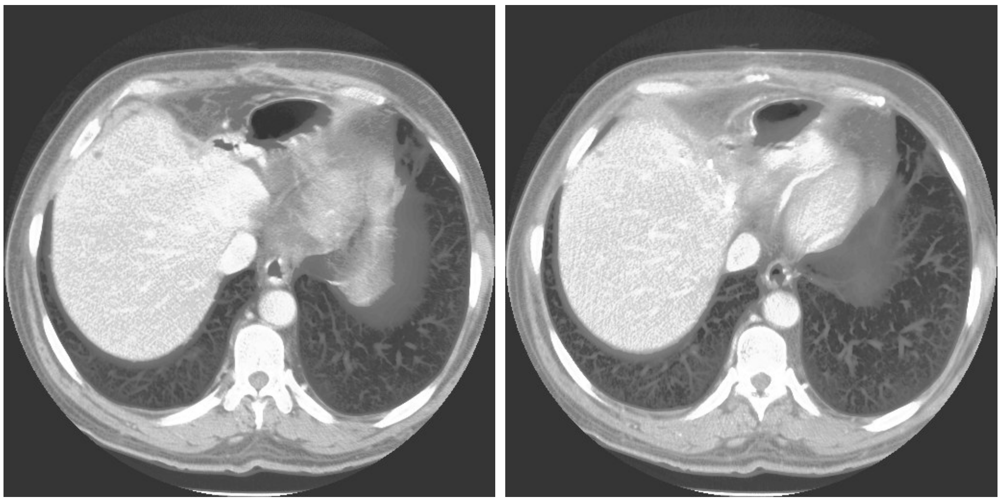


Figure 3.2: Two CT images after histogram equalisation using equation (3.1).

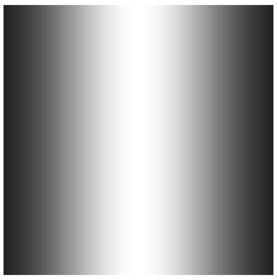


Figure 3.3: Gaussian hill template.

Matched filtering

The use of matched filtering to segment blood vessels was introduced by Chaudhuri et al. (1989). The basic idea was that 1D profiles of blood vessels can be approximated by a Gaussian curve. The Gaussian curve can be extended to 2D by forming a Gaussian hill, which in turn can be used as a template to match vessels in 2D images. The function for a Gaussian curve is:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{3.2}$$

where σ is the standard deviation, and μ is the mean. The Gaussian hill template in figure 3.3 is simply a Gaussian curve repeated in the y direction.

Since blood vessels are headed in separate directions and are dissimilar

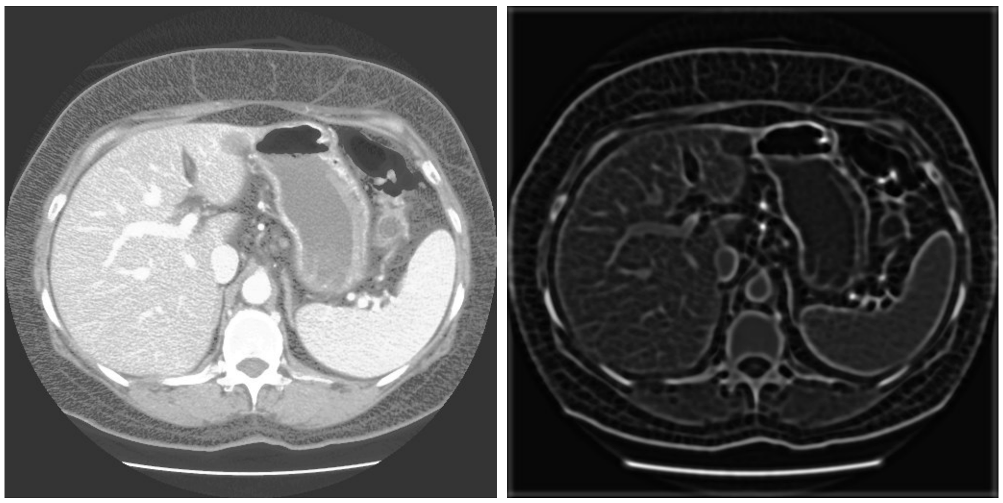

(a) (b)

Figure 3.4: a) A CT image. b) Result of matched filtering.

in size, the Gaussian hill template has to be rotated and scaled. The results from filtering with each rotated and scaled template is summarised at the end. See figure 3.4.

3.1.2 Segmentation

Two separate segmentation techniques were used in order to improve the overall result. In the following subsection these two techniques will be described in detail.

Automatic thresholding using entropy of the histogram

An automatic segmentation algorithm is needed that can segment the remaining vessels after a matched filtering process. In section 2.2.1 we briefly described some thresholding algorithms used in medical image segmentation. Our method of choice is the entropy based algorithm (Kapur et al., 1985) that is reported to be successful in segmenting liver vessels in both Glombitza et al. (1999) and Omholt-Jensen (2002). Furthermore, the system developed by Glombitza et al. (1999) is claimed to be in clinical use.

Entropy is a measure of change (Gonzalez and Woods, 2002). Homogeneous images result in zero entropy, while noisy images lead to high entropy.

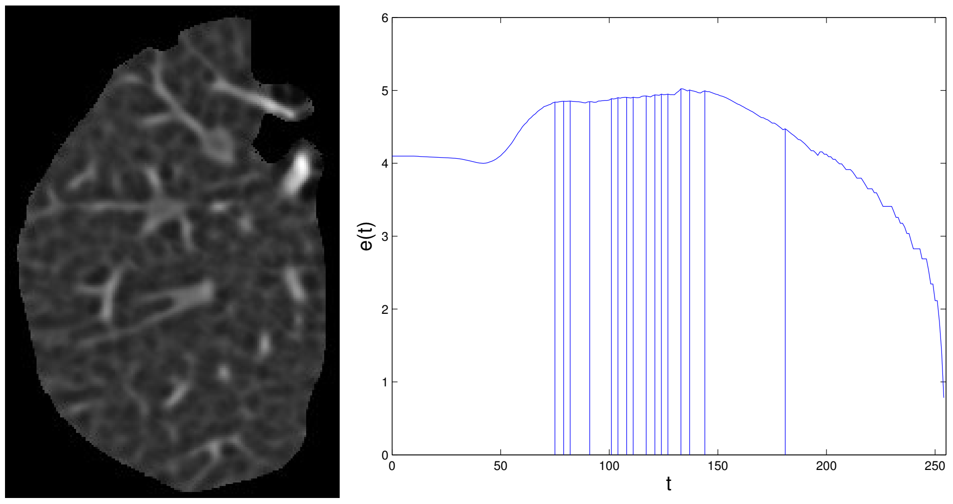


Figure 3.5: Masked result of a match filtered liver and its entropy diagram. Vertical lines mark the local maxima.

The average entropy measure, which is used in the thresholding algorithm, is defined as:

$$e = -\sum_{i=0}^{L-1} p(z_i) ln(p(z_i)) \tag{3.3}$$

where z_i is the gray value, L is the number of distinct gray levels, and $p(z_i)$ is the histogram values normalised between $[0, 1]$. Entropy based thresholding sums the entropy of the thresholded object and its background. Higher entropy yields more information about the image, and so we are looking for thresholds where the entropy peaks. Our slightly modified thresholding algorithm involves four steps:

1. An entropy is computed for each possible threshold, resulting in an entropy diagram.
2. The diagram is modified to remove less essential maxima.
3. Local maxima are found in the modified diagram.
4. A threshold is selected for the most suited local maximum.

Figure 3.5 shows an example result of steps one and three.

Due to the many local maxima we introduce an addition to the algorithm proposed by Glombitza et al. (1999). Before we find the maxima and apply the knowledge based selection, we smooth the entropy diagram by convolving

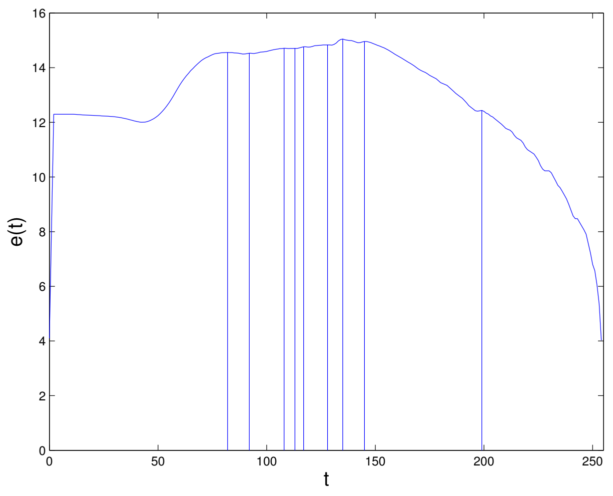


Figure 3.6: Smoothed entropy diagram from figure 3.5. Vertical lines mark the local maxima.

it with a simple $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ filter. In this way we remove the insignificant maxima, but keep those of interest. The resulting maxima are shown in figure 3.6.

A morphological transformation is used to find the local maxima (Soille, 2003). See section 3.1.2 at page 34 for details.

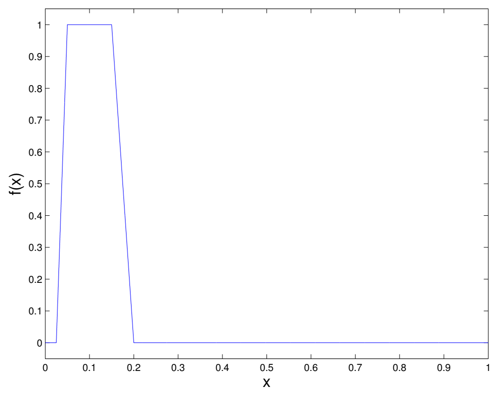
Usually more than one local maxima exists, and a solution to pick the best threshold is necessary. Glombitza et al. (1999) suggests a knowledge based algorithm that selects the threshold by comparing the segmented vessels and the liver (also reported used in Omholt-Jensen (2002)). The volume of the vessels are modulated to be from 5% to 15% of the liver, and the bounding boxes of both the vessels and the liver are to be approximately the same. The thresholded result that corresponds most to this model is selected.

After completing the first three steps in the thresholding algorithm, we then apply the algorithm from Glombitza et al. (1999) to find the optimal threshold. Two fuzzy functions are proposed to validate the candidate thresholds (see figure 3.7).

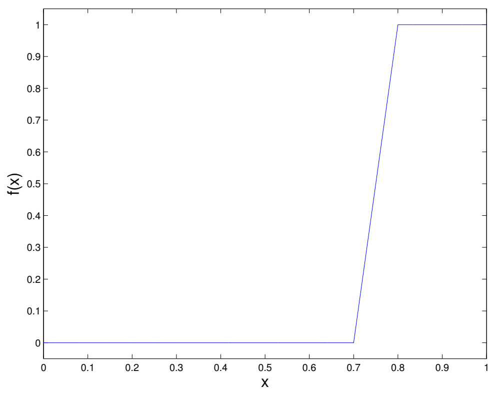
The first function returns > 0 if the ratio between the vessels and the liver are likely, and the second function results in 0 if the bounding boxes of the vessels and the liver are too diverse. These two functions are used to derive F:

$$F = f_V(\frac{Volume_{vessels}}{Volume_{liver}}) + f_B(\frac{BoundingBox_{vessels}}{BoundingBox_{liver}}) \tag{3.4}$$

where F represents the "correctness" of the threshold. The threshold with



(a)



(b)

Figure 3.7: a) Fuzzy function for volume comparison, $x = Volume_{vessels}/Volume_{liver}$. b) Fuzzy function for bounding box comparison, $x = BoundingBox_{vessels}/BoundingBox_{liver}$.

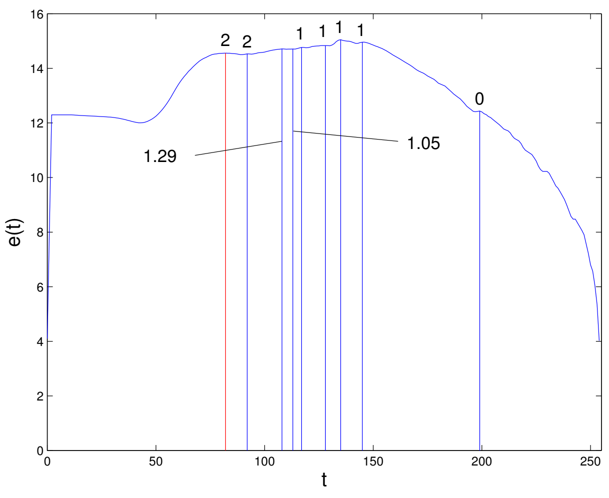
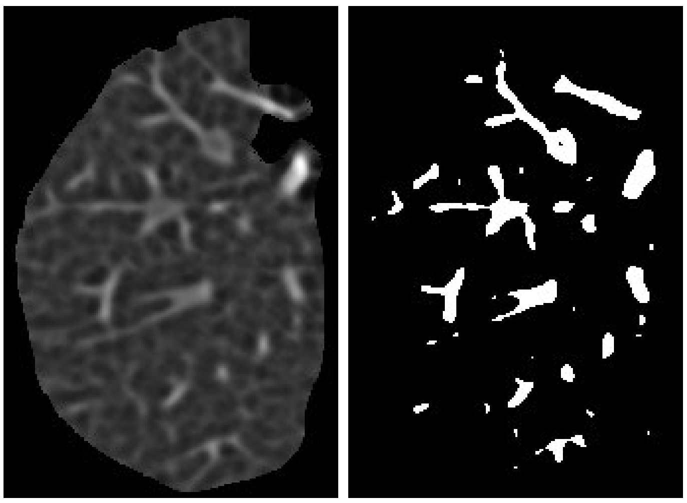


Figure 3.8: The entropy diagram shown in figure 3.6, but values F from equation (3.4) are added. Red line indicates optimal threshold t=82.

highest value F is selected. In figure 3.8 the values F are plotted on the same entropy diagram as shown in figure 3.6.

As seen in the figure, the red vertical line indicates the best threshold according to Glombitza et al. (1999). The resulting thresholded liver is shown



(a) (b)

Figure 3.9: a) Masked result of a match filtered liver. b) Thresholded result of a) using entropy.

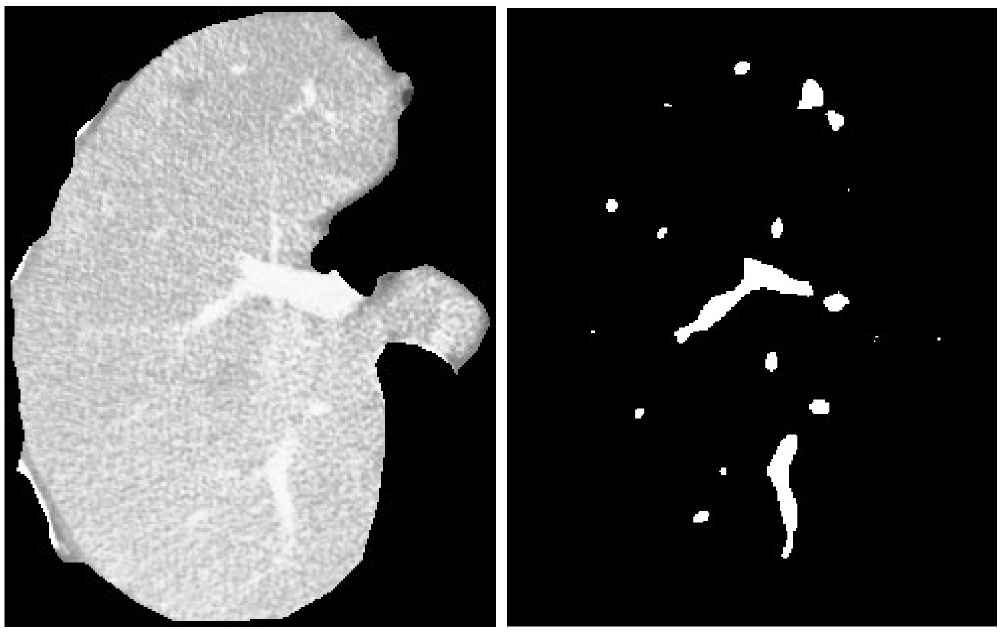
in Figure 3.9.

Segmentation based on local mean and variance

Segmentation results from the previously described preprocessing algorithms and the following thresholding are not always correct. In particular, large vessels entering the liver may be corrupted as shown in figure 3.10. The false segmentation is due to the edge effect that the boundary of the liver is causing to the matched filtering procedure.

We developed another segmentation algorithm to improve these segmentation flaws. This algorithm is based on the local variance of each pixel from the histogram equalised images. By observing the large hepatic vessels in CT images, it can be seen that the local variance within the pixels constituting the larger vessels is smaller than elsewhere in the liver. We used this to formulate the following algorithm:

1. From a histogram equalised image, find the local variance for each pixel.
2. Convolve the local variances with a small averaging filter.
3. For the pixels with intensity values above a predetermined value, store



(a) (b)

Figure 3.10: Segmentation corruption due to liver boundary. a) Masked liver from a CT image after histogram equalisation. b) After applying matched filtering and the described segmentation algorithm, large vessels entering the liver are corrupted.

the pixels that have local variance larger than value a in a new image $mask$.

4. For each pixel in $mask$, store the pixels that have local variance larger than value $a + b$ in a new image $marker$.

5. Perform reconstruction by dilation (see section 3.1.2) with the mask and marker images derived in step 3 and 4.

6. Dilate the results with a small circular structure element to better match the original size of the hepatic vessel.

The idea behind this algorithm is first that the marker pixels position the centre of the large vessels. By expanding these centres within the mask image, we rule out noise included in the mask image while at the same time include

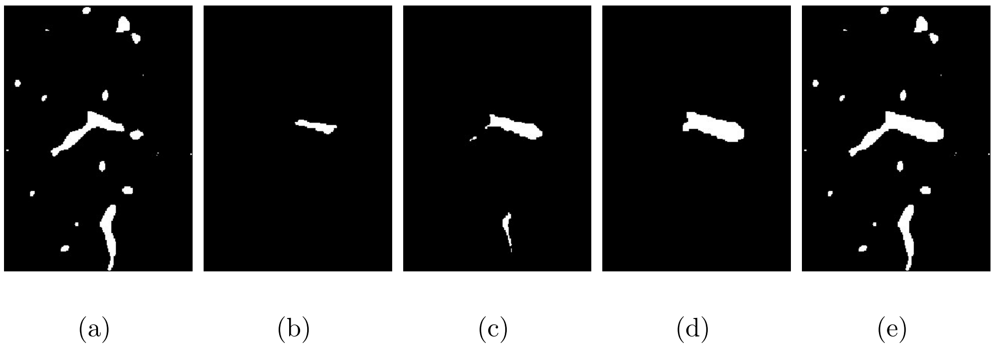


Figure 3.11: a) Segmentation resulting from matched filtering and automatic thresholding. Large blood vessels entering the liver are corrupted in the filtering process. b) Marker used in the second segmentation algorithm. The marker is selected based on local variance with a high threshold. c) Mask used in the second segmentation algorithm. The mask is selected based on local variance, but with a lower threshold compared to the threshold used to find the marker. d) Resulting reconstruction by dilation. The result is also dilated with a small structure element to better match the original vessel size. e) The two segmentation results combined.

the relevant parts of the mask image. An example algorithm execution and result is presented in figure 3.11.

Improving segmentation results using morphological operators

The resulting segmented vessels from the previous sections require further processing. Isolated pixels should be regarded as noise, and be removed. Additionally, vessels with squared corners should be rounded. A solution to both of these problems is morphological opening.

Morphological opening is equivalent to a morphological erosion followed by dilation (Soille, 2003):

$$\gamma_B(f) = \delta_{\check{B}}[\varepsilon_B(f)] \tag{3.5}$$

where $\check{B}$ is B reflected. Erosion, $\varepsilon_B(f)$, of a set f by structure element B is defined as:

$$\varepsilon_B(f) = \{x | B_x \subseteq f\} \tag{3.6}$$

where B_x means B translated by a vector x. In other words, erosion results in all the points x where B translated by x is included in f. Likewise, dilation,

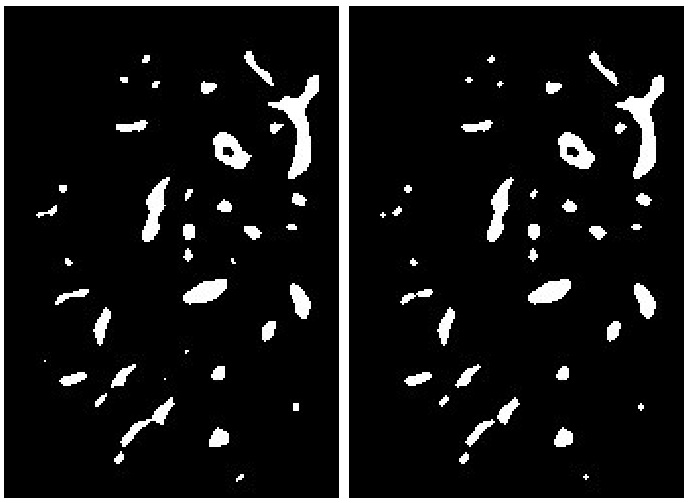

(a) (b)

Figure 3.12: a) A thresholded liver. b) Result of using morphological opening with structure element B on image a).

$\delta_B(f)$, can be expressed as:

$$\delta_B(f) = \{x | B_x \cap f \neq \emptyset\} \tag{3.7}$$

Nearly the same as erosion, the dilation result includes all the points x where B translated by x is *somewhat* included in f.

In our application a circular structure element B of radius 1 seemed adequate in both removing spurious pixels, and rounding squared corners:

$$B = \begin{pmatrix} & 1 & \\ 1 & 1 & 1 \\ & 1 & \end{pmatrix}$$

The segmentation result so far is shown in figure 3.12.

A third problem arises from the use of matched filtering. Holes may appear within the vessels that are clearly undesirable. This phenomena is shown in figure 3.13.

A way to resolve this flaw is through geodesic transformations (Soille, 2003). These transformations are based on the basic morphological operators as well, but a mask image is added to the computations. Before we continue, we will take a closer look at geodesic erosion. The definition of geodesic

(a) (b) (c)

Figure 3.13: a) A vessel in a CT image. b) Matched filtered result of a). c) Thresholded result of b).

erosion is as follows:

$$\varepsilon_g^{(1)}(f) = \varepsilon^{(1)}(f) \vee g \tag{3.8}$$

with mask image g, and marker image f. Geodesic erosion is the result of erosion with pixel values not smaller than mask image g. $\varepsilon^{(1)}(f)$ stand for erosion of f by the structure element,

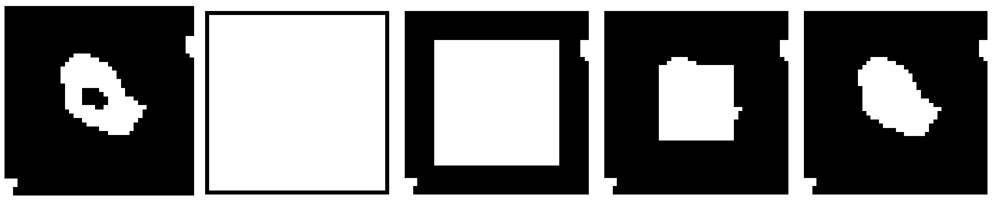
$$\begin{pmatrix} & 1 & \\ 1 & 1 & 1 \\ & 1 & \end{pmatrix}$$

As explained, this basic transformation can be used to fill the holes previously mentioned. We define holes in a binary image as regions of zeros that are not connected with the border of the image. A simple marker f is used that consists of values 1 everywhere but in the image's border. Further, the thresholded vessels are used as mask image g. Subsequent geodesic erosions, such that $f = \varepsilon_g^{(1)}(f)$, will then eventually result in f where all the holes of the binary image g are filled. The algorithm is stopped when there is no further change in f. Also known as reconstruction by erosion, this process is shown in figure 3.14.

A similar transformation, namely reconstruction by dilation, is used to find local maxima in section 3.1.2. Using this transformation, Soille (2003) defines the regional maxima as:

$$RMAX(f) = f - R_f^{\delta}(f-1) \tag{3.9}$$

where $R_f^{\delta}(f-1)$ is reconstruction by dilation with mask image f and marker image $f-1$.



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

Figure 3.14: a) Mask image g. b) Initial marker image f. c), d), e) Reconstruction by erosion ultimately yields a filled version of the thresholded image g.

3.1.3 Classification

Now that we have the segmented vessels, we want to find the points where the blood vessels enter and leave the CT images. To accomplish this, we first need to classify the segmented regions. There are two possible classifications for each segment:

1. A vessel running perpendicular to the CT image.
2. A vessel running in an oblique angle to the CT image.

The following classifier were chosen to classify the regions:

1. If the region is circular, it should be classified as 1.
2. Else, classify the region as 2.

In more detail, the classifier returns class 1 if the vessel's area and its convex area are similar, and the ratio of its minor and major axis do not exceed a predetermined limit.

The first condition has to be true since cross sections should not be classified as 1, and the last condition ensures that primarily elliptical elements receives classification 2.

3.1.4 Vessel centre extraction

The classification of the previously processed segments are used to derive the centres of the vessels. The centre points are needed as node positions in the following blood vessel graph initialisation and search.

n_1	n_2	n_3
n_8	p	n_4
n_7	n_6	n_5

Figure 3.15: The defined neighbourhood of a pixel p.

Simplest are the vessels that runs vertically through the CT image. In these cases, the centres are simply the mass centres of the segmented regions. The centres of cross sections and elliptical segments are more complicated to derive. Here, a skeleton algorithm is applied to the segments and the centre points are located at intervals within the skeleton.

Two types of centre points are used. Some of the centre points will represent potential interconnection points between adjacent slices. The mass centres and endpoints of the skeletons correspond to such points, however, additional interconnection points may be added as described in the following subsections.

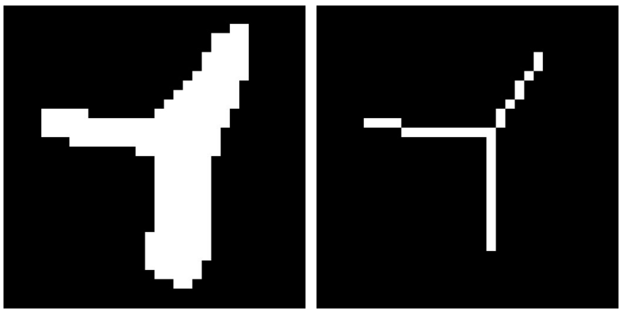
Skeleton

A skeleton is a minimal representation of a region with the same topology as the region itself. In section 2.3.4, we described several 2D skeleton algorithms. From these algorithms, Guo and Hall (1989) was selected because it results in a minimal skeleton that do not need postprocessing such as pruning. It is an iterative algorithm that consists of two processing passes for each iteration. The algorithm results in an 8-connected skeleton, and is based on a set of criteria functions rather than templates.

The neighbourhood of a pixel p is defined as shown in figure 3.15, where n_1 correspond to the upper left neighbour pixel, and n_{2-8} is the remaining 8-neighbour pixels in clockwise direction. In Guo and Hall (1989), a pixel p is deleted if the following conditions are true:

1. $\sum_{i=1}^{4}(\text{NOT}(n_{2i}) \text{ AND } (n_{2i+1} \text{ OR } n_{2i+2})) = 1$
2. $2 \leq min(\sum_{i=1}^{4}(n_{2i} \text{ OR } n_{2i+1}), \sum_{i=1}^{4}(n_{2i-1} \text{ OR } n_{2i})) \leq 3$
3. if pass 1, $((n_2 \text{ OR } n_3 \text{ OR } (\text{NOT}(n_5))) \text{ AND } n_4) = 0$
4. if pass 2, $((n_6 \text{ OR } n_7 \text{ OR } (\text{NOT}(n_1))) \text{ AND } n_8) = 0$

where n_9 and n_{10} corresponds to n_1 and n_2, respectively. The processed image and the operators "AND", "OR", and "NOT" are binary. Integer values



(a) (b)

Figure 3.16: a) A thresholded blood vessel. b) The skeleton of a).

1 and 0 are used instead of *true* and *false*. Using this definition, the sum of binary operator results is a positive integer. Further, two passes are run for each iteration as stated above. Condition 3 is used in the first pass, while condition 4 is used in the second pass. The process is repeated until there are no further pixels to be removed. Figure 3.16 shows an example skeleton result of a segmented blood vessel using this algorithm.

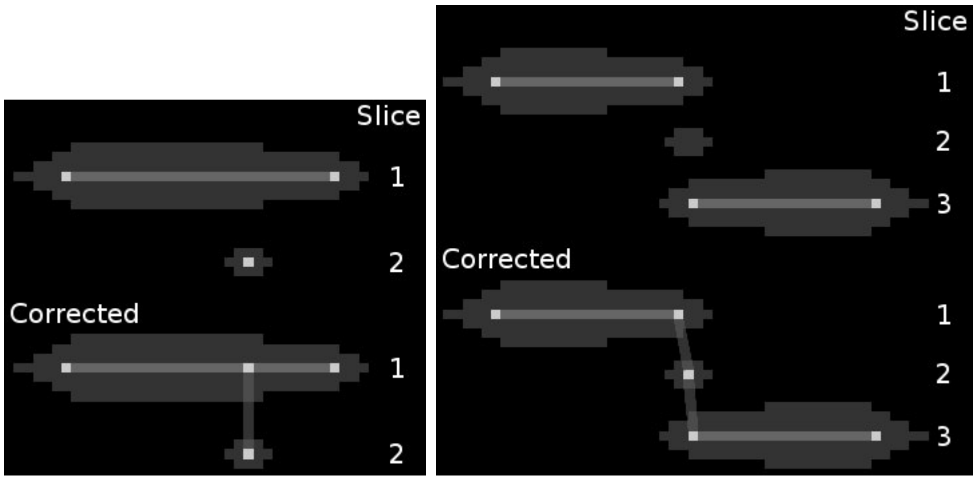
Skeleton postprocessing

Large hepatic vessels running at an oblique angle to a CT slice may still be projected onto more than one slice during a CT scan. This represents a major difficulty when processing each 2D image separately. If no further processing was performed, each of these vessels would be visualised as several vessels running in parallel to the slice planes. Another problem arises when a vessel running at an oblique angle to the CT slice branches into a vessel running perpendicular to the slice as shown in figure 3.17 a). In this case, there is no interconnection point that could interconnect these two segments.

Corrections of the vessel centre and interconnection points are therefore required. First, these corrections consist of removing vessel centres based on the thickness of similar vessel segments lying in corresponding positions in adjacent CT slices. Next, potential interconnection points are added making likely interconnections possible:

1. Follow the skeleton of each segment in each CT slice.

2. Remove the centres whose distance (see subsection 3.1.5) to the background is smaller than that of neighbouring centres in adjacent CT



(a) (b)

Figure 3.17: Vessel centre corrections. White pixels correspond to interconnection points, and opaque gray lines are possible interconnections. a) Add potential interconnection points within the skeletons near endpoints or mass centres if no such points exist. b) Interconnection points are added if interconnections between two segments are made impossible.

slices.

3. Follow the remaining skeletons.
4. Add potential interconnection points within the skeletons near endpoints or mass centres if there exist no such points already. See figure 3.17 a).
5. If endpoints have been removed in step 2 such that potential graph interconnections are made impossible, points are added to make the interconnections possible. See figure 3.17 b).

An example result is presented in figure 3.18.

Endpoint centres

The positions where the vessels enter or leave a CT image is used to interconnect vessels between CT slices. We call these positions endpoint centres, and define them as skeleton pixels that have only one neighbouring pixel. See figure 3.19 for an example.

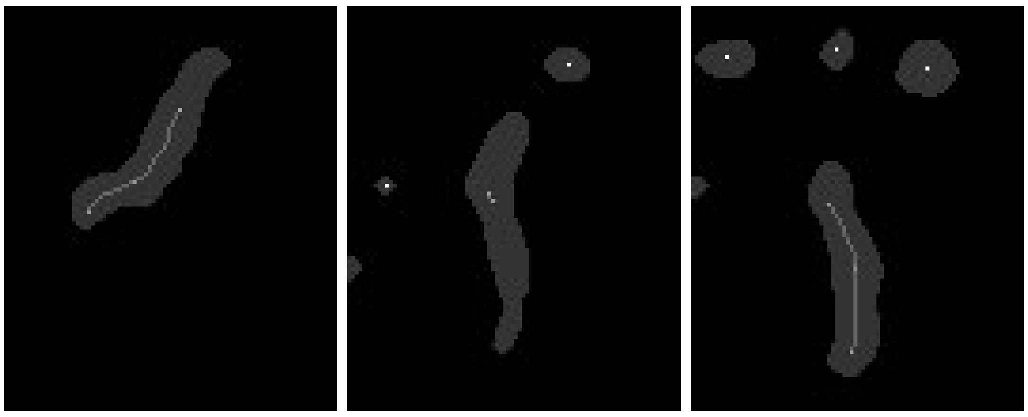

(a) (b) (c)

Figure 3.18: This figure shows the results after applying the centre corrections on three adjacent CT images. A vessel branches from a), and continues into b) and c).

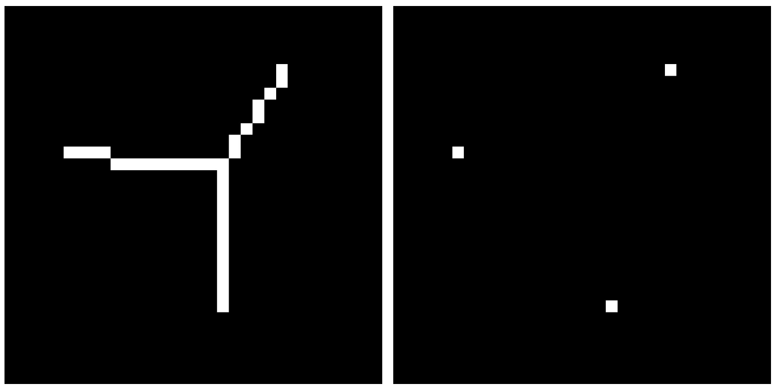
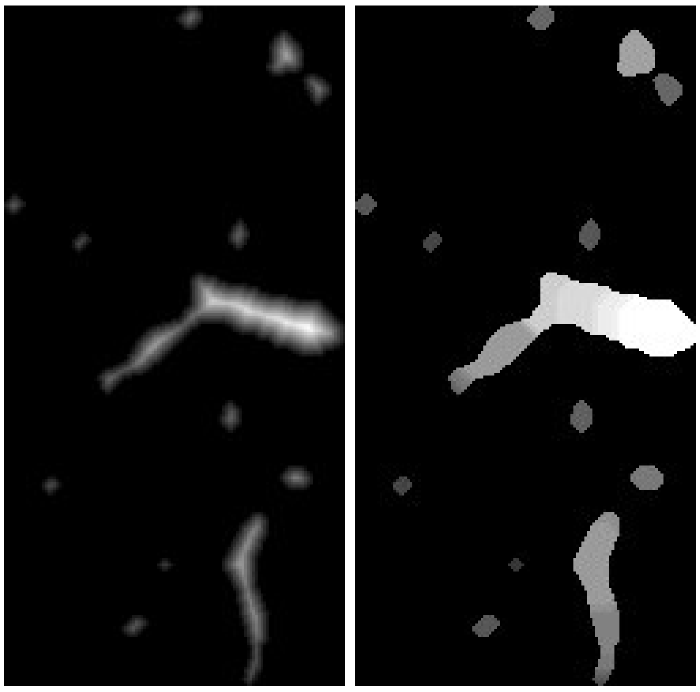

(a) (b)

Figure 3.19: a) The skeleton from figure 3.16. b) Vessel endpoint centre where the vessel enters or leaves the CT image.

3.1.5 Vessel sizes

The vessel sizes are measured using a Euclidean distance map of the segmented vessels. To ensure that the correct distance, that is the distance



(a) (b)

Figure 3.20: a) An Euclidean distance map of a CT image. b) The distance map is morphologically dilated and masked with the original segmented vessels. This is done to ensure that the right distance, the one corresponding to the size of the vessel, is picked.

corresponding to the size of the vessel, is selected, the distance map is morphologically dilated and masked as shown in figure 3.20. Especially in the algorithm described in subsection 3.1.4, when comparing vessel sizes in adjacent CT slices, the positions used for comparison do seldom correspond to a vessel centre in the adjacent CT slices. By dilating the distance map, the correct vessel size is thus selected.

An additional positive effect by dilating the distance map, is that the change in vessel sizes become more continuous in a given direction. This corresponds well with the observed anatomy of hepatic vessels.

3.1.6 Vessel graph structure

By vessel graph, we refer to nodes and node interconnections that represent the blood vessel tree found in the liver. In this brief section we will define the data structures used to represent the vessel graph. Two main classes are suggested: Nodes and interconnections. See figure 3.21 for examples. Each node correspond to a vessel joint, and the interconnections describe how these nodes are linked.

A node data structure contains an unique id to distinguish it, the node's position, and its size. On the other hand, the interconnection class holds two ids corresponding to the interconnected nodes. In order to improve the processing time of the graph algorithms, the distance between the associated nodes are stored, and interconnections in both directions between two nodes are added. Two types of interconnections are used, namely permanent and alterable interconnections. More details can be found in the following sections.

3.1.7 Vessel graph initialisation

The search space for the most likely vessel graph is vast and can not be explored in a exhaustive search. A good initialisation of the vessel graph is therefore needed as a starting point for the following global search algorithm.

We split the vessel interconnections into two groups, namely known and unknown interconnections. Vessels segments corresponding to vessels that run in an oblique angle to a CT slice are derived directly from the vessel skeletons, and can be regarded as correct. We consider these interconnections permanent and they will not be changed during the global search. On the other hand, vessel segments corresponding to vessels running perpendicular to a CT slice should be interconnected with similar vessels in adjacent CT slices. These interconnections are not known beforehand, and an algorithm to make initial interconnections is required. Figure 3.21 shows an example initialisation of both known and unknown vessel connections.

Permanent, known vessel interconnections

To derive the known vessel interconnections, the skeleton is subdivided into segments. Each section represents one small part of a blood vessel. During this stage, branch points and endpoints should be given special consideration since a chain of segments will both start and end in one.

We have earlier looked at skeletons, e.g. in figure 3.16, and at vessel endpoints in figure 3.19. In order to find the branching points, we find the

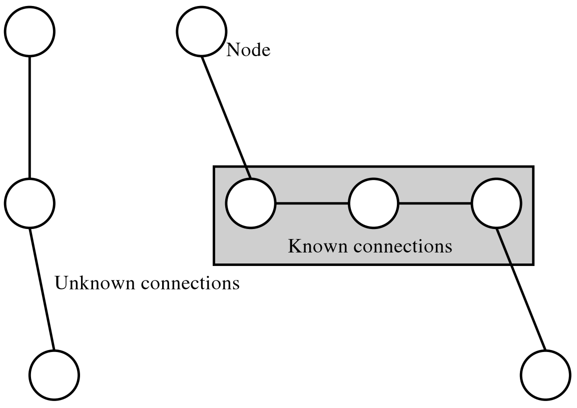


Figure 3.21:

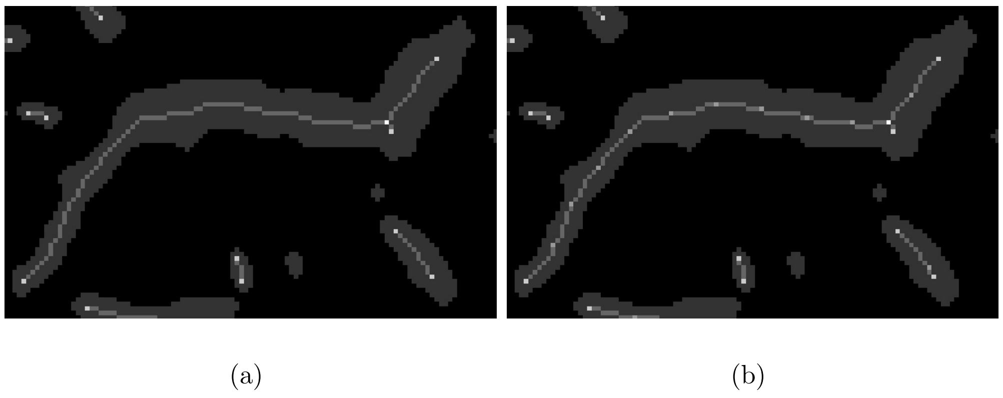


Figure 3.22: a) Thresholded vessel, with skeleton, endpoints, and branch points shown in lighter gray. b) The result after subdividing the skeleton into segments.

pixels in the skeleton that have three or more neighbouring pixels of value 1. This definition may result in more than one pixel per branching point. If this is the case, the median pixel is used as the branching point.

Segments are constructed from endpoints to branching points, and from branching points to other branching points. This process is repeated until all the remaining skeletons are divided into sections. Each segment but the last in a chain have the same length. Figure 3.22 shows the resulting sections of a skeleton separated by light gray points.

The segments are created in six steps:

1. A segment is created with the start pixel selected at an endpoint or a branching point.

2. The segment is expanded along the skeleton to a neighbouring pixel.

3. Stop if an endpoint or a branch is reached.

4. Goto step 2 if not a given length is exceeded.

5. Create a new segment with start pixel set at the current location of the skeleton.

6. Goto step 2.

Euclidean distances are used, i.e. the length of two pixels lying horizontally or vertically together is 1, and the distance of two pixels positioned together diagonally is $\sqrt{2}$. At the end, each segment's start and end points are registered as nodes, and interconnections are added in between.

Alterable, unknown vessel interconnections

Vessel segments corresponding to vessels running perpendicular to a CT slice need to be interconnected to other vessels in adjacent CT slices. These interconnections are unknown, and should be derived based on anatomical knowledge on hepatic vessels. In the initialisation though, we create likely interconnections depending on interconnection lengths and the size of the vessels that are to be interconnected. If two nodes in adjacent CT slices are positioned closely together and are of similar size, an interconnection is added between them to the initialisation graph. However, the initialisation algorithm need further prerequisites:

1. All vessel segments should be interconnected to one or two separate vessels in adjacent CT slices.

2. Closely positioned segments should be preferred as opposed to more distant segments.

3. Points whose distance exceeds a given limit should not be interconnected.

4. Segments having similar sizes should be preferred.

5. Vessel loops should be avoided.

6. If only one possible interconnection exist that satisfy the above prerequisites around two nodes, this interconnection should be made permanent.

   If two segments are the only possible interconnections that satisfy the above prerequisites, they should be permanently interconnected.

Occasionally, the third requirement leads to breaking the first, i.e. a segment will not be interconnected to any other. It might in some cases be difficult to find an appropriate interconnection, thus it would be better to leave this up to the following global search.

The sixth requirement were added during testing of our algorithm. It was apparent that some interconnections should not change during the global search since they were obviously correct. Having our global search algorithm trying to alter these interconnections would only result in a larger and more complex search space. Therefore, by making these interconnections permanent we drastically reduce the processing time, while the quality of our results are preserved.

An algorithm that satisfy the above prerequisites has been derived as follows:

1. Select the first CT slice and call it S1.

2. Name the next slice S2.

3. The interconnections derived in steps 4 and 5 are stored as temporary interconnections.

4. For each segment in S1, find the closest segment in S2 and interconnect them if the distance between the two is within a given length and their sizes are similar.

5. For each segment in S2, find the closest segment in S1 and interconnect them if the distance between the two is within a given length and their sizes are similar.

6. The temporary interconnections are ordered by interconnection length.

7. The temporary interconnections are added one by one while the vessel graph is checked for loops. If a vessel loop is detected, the last interconnection is removed. After this, the process is continued.

8. If S2 does not equal the last CT slice: S1=S2, and goto step 2.

9. If two nodes have only the option to interconnect to each other, under the condition that the distance should be within a predetermined length, make the interconnection between them permanent.

Step 4 and 5 ensures that every segment is tested for interconnection to another segment on the adjacent slices. Special care has to be taken not to create loops in our vessel graph. The interconnections are therefore added to a temporary storage and ordered by interconnection length before they are added. The ordered interconnections are added one by one while the vessel graph is checked for looping structures. By doing this, the most likely interconnections are added before less likely interconnections, which are removed if a looping structure is found.

3.1.8 Genetic algorithms

Our final step in finding a probable vessel graph is to conduct a global search. We want, as efficient as possible, to search through dissimilar graphs and test their validity. The most likely graph will be chosen for visualisation at the end.

Genetic algorithms are an interesting approach to global search (Goldberg, 1998; Banzhaf et al., 1998). The approach is based on Darwinian natural selection, popularly called "survival of the fittest". Fit specimen are selected for reproduction at the expense of less fit specimen through several generations. In this way, the specimen become more and more adapted to a given environment.

Brief introduction

The DNA of a particular organism is called its genome, and an organism's observable properties is named its phenotype. In genetic algorithms this distinction is important. Genomes are merged and changed through crossover and mutation, and from these genomes phenotypes are constructed. Furthermore, the building blocks of the genomes are grouped into genes, which in turn consists of a segment of bases. For every generation, fitness are computed for all the individual phenotypes. Based on the fitness, phenotypes are selected to populate the next generation. The phenotypes of a given generation is called that generation's population.

Crossover Crossover is the process of merging two genomes into offspring. If the genome is a string of codes, then two genomes are usually merged by first cutting each of the strings at a random position. Next, each substring

$\boxed{\text{TACGAATTT}\downarrow\text{CAAACC}}$

crossovered with

$\boxed{\text{CATGCTGTA}\downarrow\text{AGTTGA}}$

produces

$\boxed{\text{TACGAATTTAGTTGA}}$

and

$\boxed{\text{CATGCTGTACAAACC}}$

(a)

$\boxed{\text{TACGAATTTCAAACC}}$

mutates to

$\boxed{\text{TACGCATTTCAATCC}}$

(b)

Figure 3.23: Crossover and mutation. DNA is used as an example, and the letters represent the DNA bases. a) Two genomes are merged through crossover. ↓ is where the genomes are split. The last two strings are the resulting genomes. b) Genomes are shown before and after a genetic mutation. Only a few of the letters are changed.

from one genome will be joined with the fitting substring from the other such that two new genomes are created. For an example crossover, see figure 3.23.

Mutation Mutation is basically random change in a genome. It ensures that the gene pool changes, and thus that the fitness of the phenotypes can be improved. The rate of mutation, that is the probability of change in a gene, should be set low. If the probability of mutation is too high, the offspring will be too genetically different from its parents and the advantages of crossover is diminished. See figure 3.23 for an example mutation.

Definition of our genome

The DNA consists of four bases, A (adenosine), C (cytosine), G (guanosine), and T (thymine) (Purves et al., 2001). When programming genetic algorithms, we are not constricted to this "alphabet". Instead, we define our genome to be practically adaptable to the problem at hand. A gene can for example be an integer, character, float, or even a complex data structure consisting of multiple data types. The only restriction is that it must be possible to define crossover and mutation with the chosen genome structure.

Our definition is best explained by looking at figure 3.24. Each gene consists of a node and its interconnections to nodes having higher node ids. The number of interconnections is not set, that is an infinite number of interconnections are possible for each node.

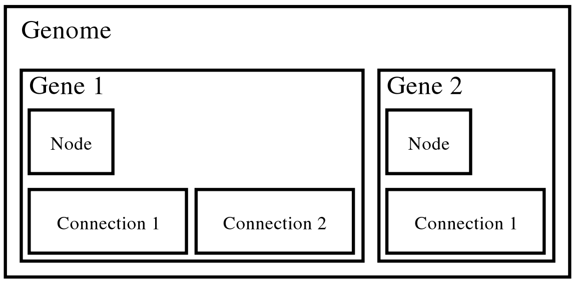


Figure 3.24: A representation of the genome structure used in our application. Each gene consists of a node and its connections to nodes having higher node ids.

Crossover For our definition of the genome structure, we also have to develop a reasonable crossover method. In our case, we simply need to define possible split positions. One solution could be to separate the genome anywhere possible. Although, such a solution could result in a split gene and thus lead to severe mutation in the crossover phase. As explained in the next section, we want to control the mutation rate with a separate parameter. Therefore, our solution is to define possible split locations between whole genes in the genome.

Mutation Using our genome structure, we cannot simply change a value as in the DNA shown in figure 3.23. Mutation in our application is instead addition and removal of interconnections. For each gene, there is a small probability that an interconnection is added, and an equal probability that one is deleted. In this way, there are no major change in the phenotypes from generation to generation, but rather a steady and controlled variation. See figure 3.25 for an example mutation.

As in the initialisation of the graph, we have to be careful not to create any loops in our vessel graph. Therefore, after adding a new interconnection, the graph is checked for loops. If a looping structure is found, the new interconnection is removed.

Adding interconnections The removal of interconnections between nodes are self-explanatory, but the addition of interconnections has to be defined. If a random node were selected for interconnection, we soon would have a highly unlikely vessel graph. We choose therefore to produce a probability distribution for interconnection addition.

The farther away two nodes are, the less likely it is that they should

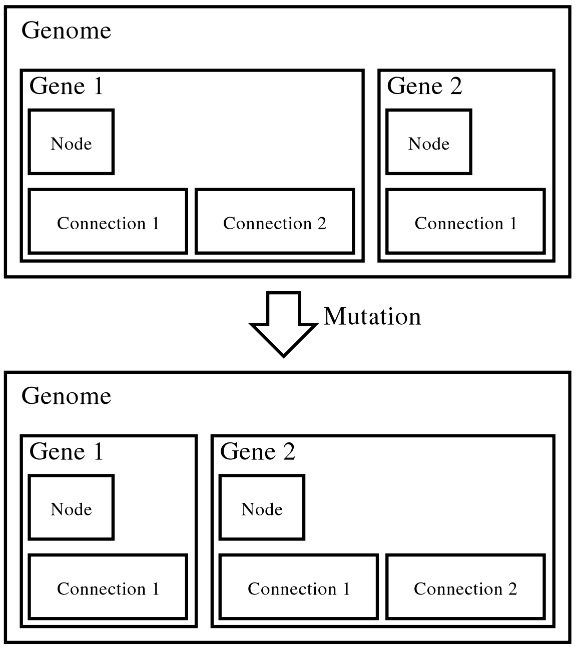


Figure 3.25: An example mutation of our genome structure. A probability is given for addition and/or removal of an interconnection

be interconnected. If distances are normalised between $[0, 1]$, the average distance between two interconnected nodes after initialisation is about 0.034. We can use this mean to formalise a probability function based on the normal distribution:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}} \tag{3.10}$$

where σ is set to 0.034, and x is the length of the suggested interconnection. Probability $p(x)$ is calculated for every possible interconnection within a predetermined distance, and using this probability an interconnection is selected through a roulette wheel method. Figure 3.26 shows the distribution in equation (3.10).

Fitness functions

The fitness represents the ability to adopt to a given environment, and the degree of fitness of a given phenotype is measured by a fitness function. In our application, the fitness function should favour likely vessel structures, while penalising unlikely structures. The selected anatomical criteria of the vessel structures are based on the work in Omholt-Jensen (2002), and extended

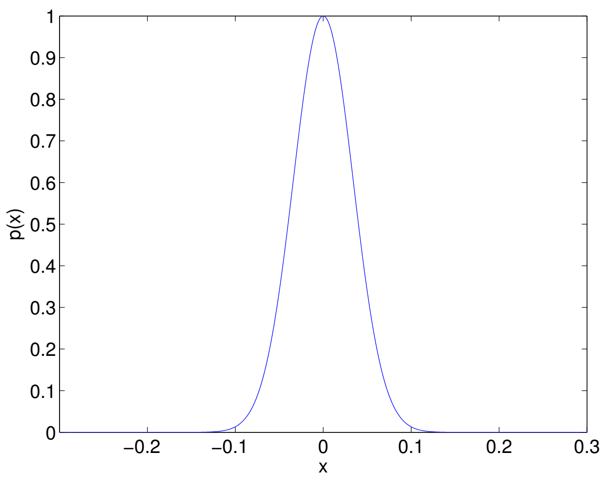


Figure 3.26: Probability distribution of equation (3.10).

further in this project. Five criteria are defined:

1. Distance - The distance between two interconnected nodes should be small.
2. Curvature - The angle between two successive segments should be small.
3. Branching - A segment should not branch into more than two segments.
4. Dimension - The vessel size should be similar between interconnected nodes.
5. Loops - Vessel loops should not exist.

Based on these criteria, we separate the fitness function of the vessel graph into five fitness sub-functions, one for each of the given criteria:

$$F = \alpha F_{distance} + \beta F_{curvature} + \gamma F_{branching} + \delta F_{dimension} + \epsilon F_{loop} \tag{3.11}$$

α, β, γ, δ, and ϵ are constants that can be used to weight the different fitness functions. In the next subsections, we will describe these fitness functions in more detail.

Distance function The distance function should award interconnections that are made between adjacent nodes, and penalise interconnections of distant nodes. We chose a simple linear function to express this as shown in

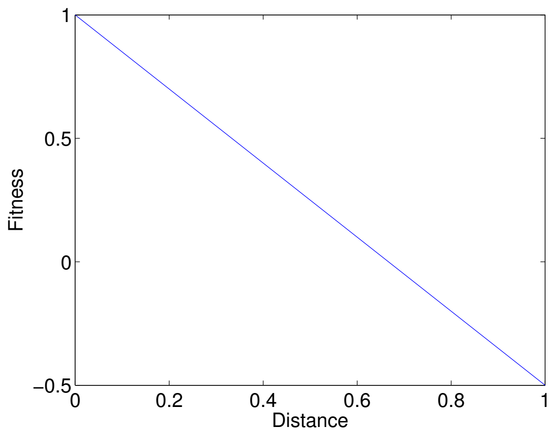


Figure 3.27: Fitness function $f_{distance}$ for interconnection lengths. Two interconnected nodes far away are less fit than those lying close by.

figure 3.27. The distance value that corresponds to zero fitness is set equal to the max interconnection length used in the graph initialisation.

Omholt-Jensen (2002) suggests that the distance should be normalised with respect to the maximum possible distance between two nodes. Although, the distance function would then be affected by the liver size. We propose instead to compute the distance function $F_{distance}$ as the average fitness of all the calculated interconnection lengths:

$$F_{distance} = \frac{1}{N_{conn}} \sum_{i=1}^{N_{conn}} f_{distance}(d(c_i)) \tag{3.12}$$

where $f_{distance}$ is the fitness function in figure 3.27, function d calculates the distance of connection c, and N_{conn} denotes the number of connections.

Curvature function According to Omholt-Jensen (2002), the angle between two successive interconnections usually stays between $[0, \frac{\pi}{2}]$, and is likely to be as small as possible unless the interconnections are part of a vessel branch. The angle θ is derived from the use of the dot product, and is the smallest angle between two vectors:

$$\theta = \arccos(\frac{v_1 \cdot v_2}{|v_1||v_2|}) \tag{3.13}$$

where v_1, and v_2 are the two interconnection vectors. We define the fitness function for curvature as shown in figure 3.28, where a quadratically decreasing function is used:

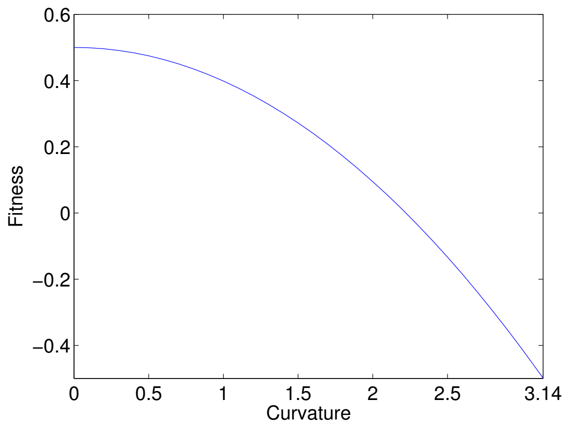


Figure 3.28: Fitness function $f_{curvature}$ for the curvature between connections. Small curvature leads to high fitness.

$$f_{curvature}(\theta) = 0.5 - \frac{\theta^2}{\pi^2} \tag{3.14}$$

The average fitness of all curvatures is calculated in function $F_{curvature}$:

$$F_{curvature} = \frac{1}{N_{curv}} \sum_{i=1}^{N_{curv}} f_{curvature}(a(c_i)) \tag{3.15}$$

where N_{curv} is the number of curvatures tested, and function a calculates the curvature between interconnection c_i and its successive interconnections using equation (3.13).

Branching function Blood vessels sometimes branch into two sub-vessels. However, due to the slice thickness of the CT scans it is possible that a vessel branches more than once between two slices. We should thus allow branching into several sub vessels, but at the cost of fitness. Since it is unlikely that a vessel fork several times between slices, we give interconnections which separate more than three times a fitness of 0. One or zero branches are likely and are granted fitness 1, while three branches are penalised with $\frac{1}{3}$ fitness. The function $f_{branching}$ for an individual node is displayed in figure 3.29.

The total branching function $F_{branching}$ should not be affected by the number of vessel sections. $F_{branching}$ is therefore normalised with respect to the number of interconnections:

$$F_{branching} = \frac{1}{N_{conn}} \sum_{i=1}^{N_{conn}} f_{branching}(c_i) \tag{3.16}$$

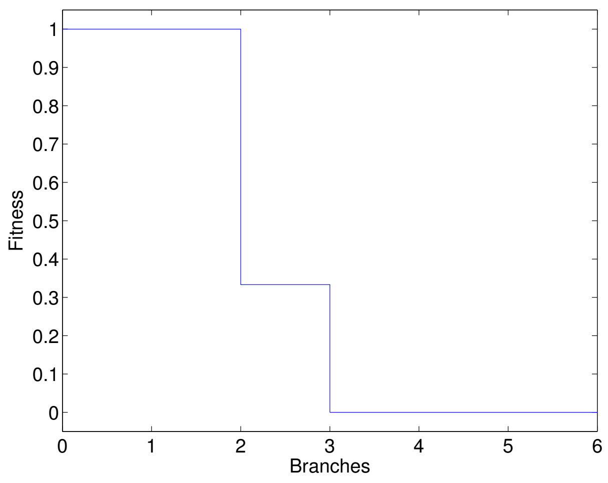


Figure 3.29: Fitness function $f_{branching}$ for branching. More than two sub-vessels lead to less fitness.

where N_{conn} is the number of interconnections, and c_i is interconnection i.

Dimension function Hepatic vessels are either increasing or decreasing in size in a given direction. However, the rate of change is often low. This knowledge is used to formulate the next fitness function, namely dimension fitness. The dimension fitness should ensure that interconnected nodes with similar sizes are rewarded high fitness. As in the distance function, we chose to use a linear function to express this as shown in figure 3.30. The difference value that corresponds to zero fitness is set equal to the max size difference used in the graph initialisation.

To calculate the total dimension fitness $F_{dimension}$ we use the average of all $f_{dimension}$:

$$F_{dimension} = \frac{1}{N_{conn}} \sum_{i=1}^{N_{dim}} f_{dimension}(c_i) \tag{3.17}$$

where, as above, N_{conn} is the number of interconnections, and c_i is interconnection i.

Loop function The last part of the fitness function F is the loop function. Since vessels in the liver generally do not form vessel loops, such loops found in the vessel graph should be penalised. Experimentation showed that the best solution when finding a graph containing a loop was to simply give it a

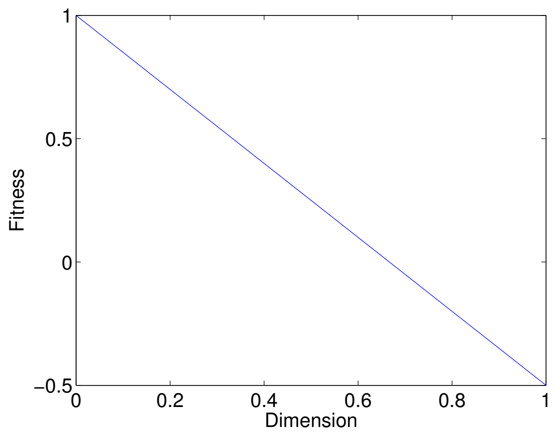


Figure 3.30: Fitness function $f_{dimension}$ for size difference of interconnected nodes. Small size differences are rewarded high fitness.

low fitness:

$$F_{loop} = -4E_{loop} \tag{3.18}$$

where E_{loop} returns true if a loop exists, and false otherwise. It should be noted that in Omholt-Jensen (2002) loops are avoided entirely by disallowing creation of interconnections resulting in a loop. Similar methods are developed as described in previous sections, but looping can still occur. During a crossover, interconnections from two different graphs are merged and there are no mechanisms that ensure that the resulting graph contains no vessel loops.

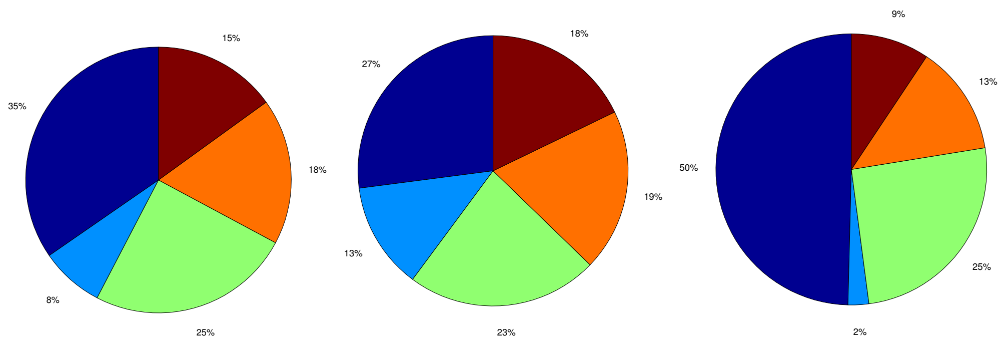
Selection of optimal vessel graph

After the fitness of all the individual phenotypes in a generation are calculated, a selection mechanism should select the most fit for reproduction. We propose the use of Boltzmann selection (Sonka et al., 1999), which calculates the probability of selecting a specific genome by:

$$p(F_i) = e^{\frac{-f_i}{T}} \tag{3.19}$$

where F_i is the fitness of a given genome i, and T is the temperature used in simulated annealing (Sonka et al., 1999). Simulated annealing is a optimisation method that slowly lowers the energy of a search landscape.

In the first generations, T is set to a high value. We call this stage “exploration”, meaning that the value of the fitness is not highly decisive.



(a) Normal (b) Exploration (c) Exploitation

Figure 3.31: Pie charts showing $p(F_i)$ from equation (3.19) of 5 separate genomes in different colours. a) No temperature, that is $T = 1$. b) Temperature T set high. c) Temperature T set low.

At this stage, many different graphs are selected for reproduction. During evolution, the value of T is slowly reduced, and the fitness values become more and more determining. The final stage is called "exploitation", and only the phenotypes with highest fitness are selected. See figure 3.31 for an example.

We wish to constantly improve the fitness of our graphs. The best graph is therefore copied directly to the next generation without any mutation or crossover. A popular term for this is elitism (Goldberg, 1998).

3.2 3D vessel reconstruction

An apparent problem with the 2D vessel reconstruction described in the previous section, is that important information is disregarded in the vessel processing. When the vessels are emphasised by matched filtering for instance, vessel information on adjacent slices are not considered. Likewise, vessel centres are extracted from one CT slice at a time, and the algorithms are not exploiting the CT data to the full extent.

In this section, we propose algorithms that retrieve hepatic vessels using 3D processing techniques that work on a CT volume, and not its individual images one at a time. We base our 3D techniques on the methods used in the previous section. Further changes have also been applied, especially with regards to the search for the most likely vessel graph. An additional weakness with the techniques described in the previous section, is that the algorithms do not result in the same vessel graph when processing a CT scan more than once. This is due to the global search, which attempts to find the global optimum graph through random changes in the processed vessel graphs. Instead, we present here a local search mechanism that give the same result each time the algorithm is executed for a given CT scan.

3.2.1 Preprocessing

As in the previous section, we apply algorithms to emphasise the hepatic vessels before segmentation. The major difference, however, is that the CT images are not histogram equalised individually. This may alter the intensity relationship between adjacent slices. We instead perform a histogram equalisation on the whole CT scan in order to increase the visibility of the hepatic vessels.

Histogram equalisation

As mentioned above, we perform histogram equalisation on the whole CT scan rather than on the individual CT images. The hepatic vessels are thus easier to distinguish for the human eye, while the intensity relationship between the CT slices is preserved.

Matched filtering

Matched filtering is applied in order to emphasise the hepatic vessels. Instead of matching blood vessels in 2D, we implement 3D filters that matches blood vessels in all directions through a 3D convolution. The basic 3D blood vessel template is based on the Gaussian distribution as follows:

$$template(x, y, z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-((x-\mu_x)+(y-\mu_y))^2}{2\sigma^2}} \tag{3.20}$$

where μ_x and μ_y are the centres of the template in the xy-plane, and σ corresponds to the variance where $\sigma = \sigma_x = \sigma_y$. Furthermore, the template is the same for all z. Figure 3.32 show an example template filter.

The 3D template is scaled and rotated in all directions in order to match the hepatic vessels with different sizes and headings. The templates are convolved with the CT scan, and the result from each convolution is summarised at the end.

Since the CT scans may have different voxel size ratios, it is important to scale the templates according to this ratio. This scaling is executed after the rotation and scaling previously described.

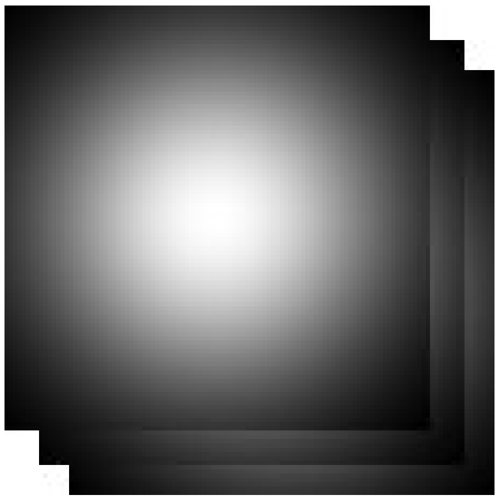


Figure 3.32: Three xy-planes of an example 3D template that is used to match hepatic vessels in a CT scan. The template is scaled and rotated in order to match the different vessels.

Multiple 3D convolutions require a great deal of computational resources, and the above calculations may take hours to execute. A simple and inexpensive solution is to implement these 3D convolutions on the graphics processing units (GPUs) on a modern hardware accelerated graphics card (GPGPU-group, 2005). Expected run-time improvements is discussed in Eidheim et al. (2005).

Anisotropic diffusion

Anisotropic diffusion was briefly introduced in subsection 2.1.5. It represents a method that is very similar to heat conduction, but instead of diffusing all image intensities, strong edges are kept.

Anisotropic diffusion was used in Soler et al. (2001) in a preprocessing stage before segmenting various tissues in the liver. The article presents techniques that are processed on each image individually. Extending this preprocessing method to 3D makes it more powerful due to the increase in data that can be exploited.

We implemented anisotropic diffusion for use prior to a second segmentation algorithm, which will be described in more detail in section 3.2.2. Before applying diffusion, however, we decided to execute a morphological opening on the CT data. This was done in order to make the pixel intensity of the hepatic vessels and the remaining liver tissue more distinct.

The anisotropic diffusion equation is as follows:

$$\frac{\partial I}{\partial t} = \nabla(c\nabla I) \tag{3.21}$$

where the diffusion coefficient c varies in space but not time. We define ∇I with finite differences as follows:

$$\partial_E I = I_{x+1,y,z} - I_{x,y,z} \tag{3.22}$$
$$\partial_W I = I_{x,y,z} - I_{x-1,y,z} \tag{3.23}$$
$$\partial_N I = I_{x,y+1,z} - I_{x,y,z} \tag{3.24}$$
$$\partial_S I = I_{x,y,z} - I_{x,y-1,z} \tag{3.25}$$
$$\partial_U I = I_{x,y,z+1} - I_{x,y,z} \tag{3.26}$$
$$\partial_D I = I_{x,y,z} - I_{x,y,z-1} \tag{3.27}$$

Further, ∇^2 is defined as:

$$\partial^2 I = \frac{1}{6}(\partial_E I - \partial_W I + \partial_N I - \partial_S I + \partial_U I - \partial_D I) \tag{3.28}$$

We next define c to be small near edges of I, and large in homogeneous regions:

$$c = 1 - (\partial_E I_{orig} + \partial_W I_{orig} + \partial_N I_{orig} + \partial_S I_{orig} + \partial_U I_{orig} + \partial_D I_{orig}) \tag{3.29}$$

where I_{orig} is the original volume to be anisotropically diffused, and c is set to 0 if $c < 0$. Figure 3.33 shows an example result.

3.2.2 Segmentation

From the preprocessing result computed in the previous section, we now apply segmentation algorithms to locate the hepatic vessels. First, we apply the thresholding algorithm used in 2D vessel reconstruction on the matched filtered result.

A second segmentation algorithm is applied to improve the segmentation result. This algorithm is based on intensity values and local variance of the CT scan. The automatic thresholding algorithm is also applied here at the end.

Automatic thresholding using entropy of the histogram

We have extended the thresholding algorithm outlined in section 3.1.2 to work on volumes instead of images. First, a volume histogram has to be

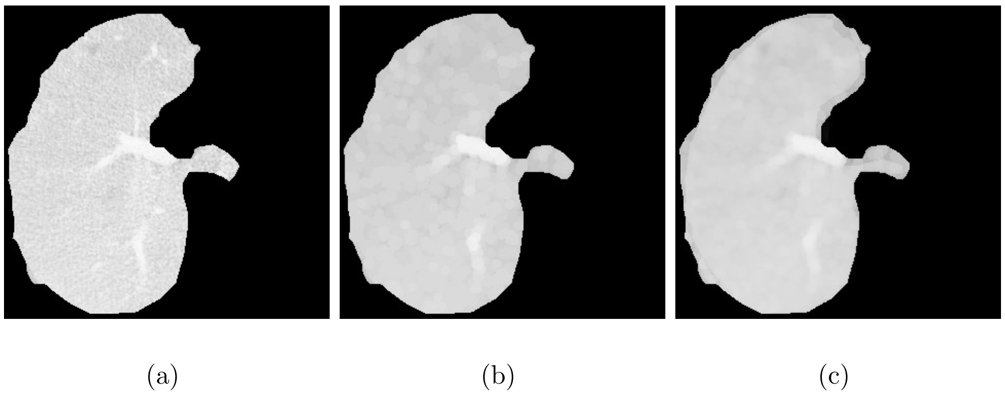


Figure 3.33: Example 3D anisotropic diffusion of a CT volume. a) An original CT slice. b) Morphological opening result of a). c) Resulting anisotropic diffusion of b). The vessels can now be more easily distinguished from the liver tissue.

found, but the calculation of a volume histogram is similar to that of an image histogram. The only difference is that pixels are counted in three dimensions instead of two. When the volume histogram has been computed, the entropy is measured and local minima are found as described in section 3.1.2.

The next step is to select one of these thresholds to apply to the matched filtered CT scan. In the 2D reconstruction algorithm, we used the knowledge based fuzzy functions presented in Glombitza et al. (1999). This technique has been implemented for 3D volumes as well. Glombitza et al. (1999) proposed to compare the area and bounding box of the thresholded hepatic vessels and the liver. The simple 3D extension consists of comparing the volumes instead of areas, and use a 3D bounding box. All the different threshold results are compared to the liver, and the most appropriate is selected.

Segmentation based on anisotropic diffusion and local variance

The projections of hepatic vessels in CT scans are usually of bright pixel intensity with low local variance. We have used this to define a second segmentation algorithm to locate vessel regions that the matched filtering fail to emphasise. These are typically regions of larger vessels that lie close to the boundary of the liver. This was also an issue in the 2D reconstruction algorithm, but another solution is presented here.

We base our segmentation algorithm on the anisotropic diffusion procedure outlined in subsection 3.2.1. The result of this method is first thresh-

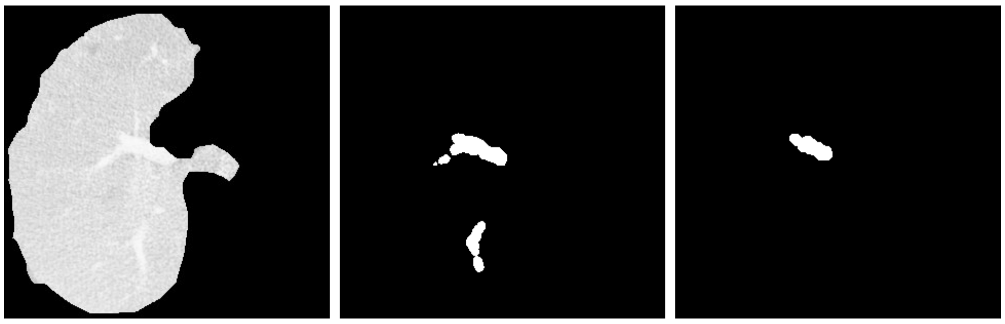

(a) (b) (c)

Figure 3.34: An example result using the second segmentation algorithm. a) Original CT slice. b) Thresholded diffusion result. c) Segmentation result consisting of pixels from b) having low local intensity variance.

olded using the automatic thresholding algorithm from subsection 3.2.2. Next, for each segmented pixel in the previous step, the local variance is calculated. If the local variance surrounding a voxel is below a predetermined value, the voxel is set true. The result is dilated slightly to compensate for the fact that boundary pixels have high local variance. An example is given in figure 3.34.

The segmentation results from both the segmentation algorithms are added together, and holes are filled through morphological reconstruction described in 3.1.2. Finally, a morphological closing with a small spherical structure element was applied to the segmentation results in order to attach separated vessels and to close small concave regions. These postprocessing methods were applied to decrease the possibility of vessel loops, and to make the reconstruction algorithm less dependant on the heavily parameter controlled graph search.

3.2.3 Vessel graph extraction

The next step is to locate the vessel centres and to derive the vessel widths. We use this information first to initialise a vessel graph, and thereafter to improve the vessel graph in a local search for the most likely vessel graph.

Vessel centres

In section 3.1, we used 2D skeletons or mass centre points to locate vessel centres depending on classifications of the 2D vessel segments. Instead, we

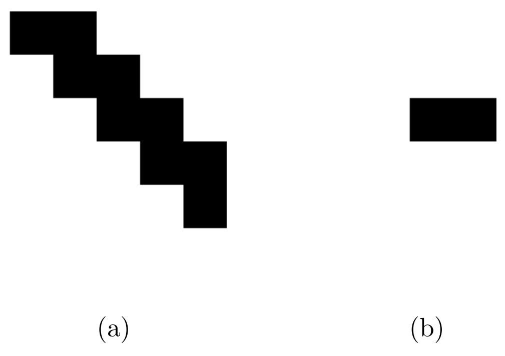


Figure 3.35: The algorithm presented in Palagyi et al. (2001) fails to preserve the 6-connected endpoints shown in this figure. a) The original object. b) The skeleton of the object.

will apply a 3D skeleton algorithm on the segmented vessels computed in the previous section.

The derivation of 3D skeletons have been the subject of many studies in the field of image analysis. A short overview was given in section 2.3.4. Many articles propose methods to find and delete simple points, that is pixels in a volume that can be removed without affecting the topology of the volume. An additional challenge is to select an appropriate order in which simple points are to be removed.

A mathematically sound classification of simple points was given in Malandain and Bertrand (1992). This simple point classifier has been used in many other articles such as in Palagyi et al. (2001) to create object skeletons in medical applications. During testing of this algorithm, it was discovered that it had a fundamental weakness under certain conditions. Figure 3.35 shows an example flaw where diagonal 6-connected pixels are mistakenly removed even though they are endpoints in a 6-connected skeleton. This is due to the endpoint check that is defined, where a pixel is marked as an endpoint only if the number of 26-connected neighbours are equal to 1. The number of 26-connected neighbours in figure 3.35 is 2, and the pixels are therefore removed one by one. If we modify the endpoint classification algorithm to count the number of 6-connected neighbours instead of the 26-connected neighbours, new endpoints would be introduced depending on the order in which pixels are removed. This is also undesirable, since we want the skeleton computation to be independent of an object's rotation.

A new 3D skeleton algorithm was therefore derived based on the simple point classifier presented in Malandain and Bertrand (1992). Here, a pixel p is defined as a simple point if the following two criteria are fulfilled:

1. The number of 26-connected foreground pixels in the 26-neighbourhood of p is 1.

2. The number of 6-connected background pixels in the 18-neighbourhood of p is 1.

The 26-neighbourhood of a pixel p corresponds to all the pixels that are 26-connected to p, but not including pixel p. The 6-neighbourhood is defined in the same manner.

6-connectivity is ensured for all objects in the data volume using this simple point classifier to remove pixels. A complete proof is given in Bertrand and Malandain (1994).

After the pixels are classified, a scheme to delete the simple pixels is needed. We have derived the following procedure:

1. Keep endpoints, that is true pixels that have only one 6-connected true neighbour.

2. For all other true pixels, store simple points in an array.

3. Remove the pixels in the array one by one, but keep pixels that are no longer simple points.

4. Repeat step 3 until no further change.

5. If a pixel has been deleted in step 3, goto step 1.

The first solution that comes to mind when using the simple point classifier, is to execute step 1 to 3 until a skeleton is produced. This solution might, however, remove an object altogether if all the pixels constituting the object are marked simple. Step 3 is repeated to remove unnecessary endpoints that might result after one or few executions of this step.

This algorithm will result in a minimal representation of objects in a 3D volume. The resulting skeleton will lie within the medial axis since object boundaries are removed iteratively. At the same time, due to the simple point definition, the topology of the objects are preserved. Figure 3.36 shows an example execution of this algorithm.

Smaller skeleton branches are unimportant in the analysis of the hepatic vessels, and are therefore pruned. At the end, the skeletons are subdivided into nodes as described in section 3.1.7.

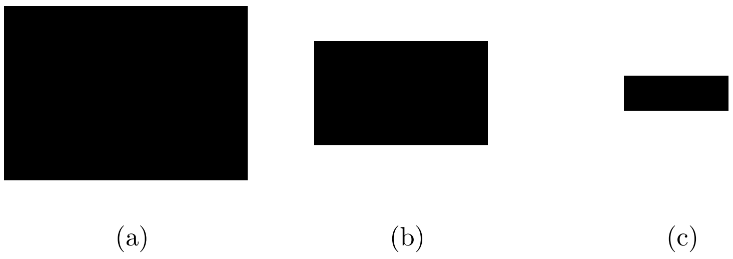


Figure 3.36: Successive thinning of a 2D square using our new algorithm. 3D thinning is similarly computed. a) a 2D square. b) Resulting thinning after one iteration. c) Resulting skeleton after two iterations.

Vessel sizes

As in subsection 3.1.5, the vessel sizes are computed from a dilated distance transform. 3D Euclidean distance is used, and the distances are scaled with respect to the voxel size ratio of the CT scan.

3.2.4 Vessel graph initialisation

We now have the node positions and their sizes that we need to initialise the vessel graph. Additionally, we have most of the interconnections required through the 3D skeleton algorithm. Some interconnections may be lacking, however, due to false vessel segmentation. We solve this by interconnecting adjacent nodes that are similar in size.

The initialisation algorithm is as follows. For each interconnected vessel structure, the most fitting interconnection to another vessel structure is chosen. This process is repeated until there are no more fitting interconnections. In this way, looping structures are avoided and more likely interconnections are added before less likely ones.

The initialisation performed in the 3D vessel reconstruction is less complex than in the 2D vessel reconstruction, and more interconnections are derived directly from the vessel skeletons. The 3D technique is thus less vulnerable to error, and it should in theory produce better results.

Smoothing the vessel graph

An additional step is performed in the initialisation of the vessel graph, namely averaging the node positions with their interconnected neighbours. The step size of the original nodes may be large in the z-direction due to the CT slice thickness. By applying this average filter, the locations of the hepatic vessels are thus more naturally positioned.

3.2.5 Local search

A local search is finally conducted to improve the initialised vessel graph. This method is chosen before a global search for two reasons. First, an exhaustive search for a global optimum can not be executed in reasonable time, and thus the solution graphs would differ in separate executions of the global search. Second, a local search results generally in a better solution in less time than a global search.

In the following subsections we will describe the local search methods we chose to apply to the vessel graph. The vessel graph corrections are applied in the order they are listed.

Size difference correction

If two interconnected nodes differ significantly in size, the interconnection between them is removed. This will further remove spurs from from the vessel skeletons, as well as separate vessel structures that should not be joined.

Angle correction

Starting at the trunk of each vessel structure, the interconnections are followed recursively. Each interconnection that have a larger angle than $\frac{\pi}{2}$ to the previous interconnection is removed.

Multiple branch correction

If a vessel branches into more than two sub-vessels after the above corrections, we apply no further change to the vessel structure. The size difference correction and the angle correction should remove the most unlikely vessel structures. Additionally, branching into more than two sub-vessels may occur in the CT data, for instance if a vessel subdivides more than once between two CT slices.

Vessel loop corrections

Postprocessing algorithms were applied to the segmentation results in order to remove a number of possible looping structures. However, not all vessel loops are guaranteed to be removed using these methods alone. Testing showed that the skeleton might contain smaller vessel loops even after the postprocessing algorithms. These loops are removed by simply deleting one of the looping arcs that consists of pixels with two 6-connected neighbours (pixels not corresponding to skeleton branches).

The above solution is very simple and good results are not guaranteed. A better loop correction procedure needs to be developed if this technique is to be used in clinical applications.

3.3 Visualisation

The finished graph needs to be visualised for inspection and analysis. We have all the necessary data, that is locations and sizes of the nodes and node interconnections, to create a realistic visualisation of the hepatic vessels. A new visualisation algorithm is presented here to create a natural looking visualisation of a vessel graph. This work is also published in Skjermo and Eidheim (2005).

In order to take advantage of modern graphics hardware, we need to extract a polygon mesh model from the graph. A number of techniques exist to create polygon mesh models from voxel data, such as contour tracking, opaque cubes, marching cubes, dividing cubes, and marching tetrahedra (Watt, 1999; SIGGRAPH-group, 2005; Lorensen and Cline, 1987). In our case, however, we need to create the mesh model from a vessel graph, and not voxel data.

The most related work to computing mesh models of graphs is mesh generation of trees. The major difficulty here is to create natural looking branches. Several methods have been proposed, for instance by parametric surfaces in Bloomenthal (1985), key-point interpolation in Oppenheimer (1986), "branching ramiforms" in Bloomenthal and Wyvill (1990), and "refinement by intervals" in Lluch et al. (2001).

Our algorithm is based on the work of Felkel et al. (2002a) and Felkel et al. (2002b). Basically, it consists of creating connected cubes, which are subdivided through Catmull-Clark subdivision (Catmull and Clark, 1978). The result is a smooth, detailed vessel visualisation with natural looking branches. The algorithm will be described in more detail in the following subsections.

3.3.1 Vessel root

The root node, which correspond to the largest node having a single interconnection, is first found for each of the separate vessel trees[1]. From this node, the mesh is created iteratively for each node in the tree processed. The first part of the mesh is created at the location of the root node; a square polygon that is scaled depending on the size of the root node. The square polygon is rotated such that its normal vector points toward the next node.

The mesh is continued depending on whether the next node represents a simple vessel continuation or a vessel branch. These computations are described in the next two sections.

3.3.2 Simple continuation

If the next node has only one further interconnection, we call this a simple continuation of the vessel. A new scaled square polygon is then added at the position of this new node, and it is rotated so that its normal vector points in the direction d_1^s:

$$d_1^s = node_{position}(n-1) - node_{position}(n+1) \tag{3.30}$$

where $node(n)$ corresponds to the current node that involves the simple continuation.

Between the last and the newly made square polygons, new quadrilaterals are added such that a hexahedron is formed as shown in figure 3.37

3.3.3 Branch

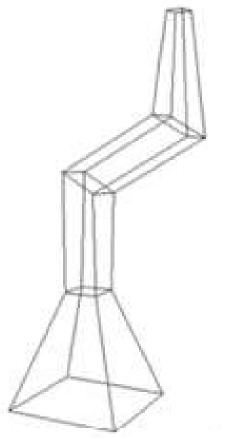
A vessel branch exists where a node has several further interconnections to other nodes. We call these nodes child nodes and denote them by $node(n+1)$.

In order to create a natural looking branch, we use one or more hexahedrons to connect the sub-vessel hexahedrons together in the branch. First of all, a new square polygon, named square 1, is created at position p_1^b:

$$\begin{aligned} p_1^b &= node_{position}(n) - branchSize(node_{size}(n)) \\ &\quad (node_{position}(n) - node_{position}(n-1)) \end{aligned} \tag{3.31}$$

where $branchSize$ is a function depending on the size of $node(n)$, and is used to control the size of the branching structure. The position is shifted slightly

[1]The vessel graph generally consists of several vessel trees, that is a collection of nodes that are directly or indirectly interconnected.



(a)

Figure 3.37: This figure illustrates how the mesh is generated if each node has only one further interconnection. Image courtesy of Skjermo and Eidheim (2005)

towards the last node position to leave more space for the branch. The square is rotated such that its normal vector points in the direction d_1^b:

$$d_1^b = node_{position}(n) - node_{position}(n-1) \tag{3.32}$$

Next, a node interconnection is selected to form a branch structure where additional sub-vessels can be added. The selection of the next node is derived from the *Da Vinci* rule presented in Richter (1970), and the results presented in Murray (1927). The *Da Vinci* rule states that the cross section area of a segment is equal to the combined cross section areas of its child segments after a branch. Richter (1970) derived further that the child segment with the largest cross section area has the smallest angle to the parent segment. We base our next step in the algorithm on this, and order the child nodes by their sizes. The node with the largest size is selected to form the structure of the branch. A new square is thus added at the following position p_2^b:

$$\begin{aligned} p_2^b \;=\; & node_{position}(n) + \\ & branchSize(node_{size}(n)) \\ & (node_{position}(sortedNodes(n+1),1) - node_{position}(n)) \end{aligned} \tag{3.33}$$

where $sortedNodes(n+1,1)$ returns the child node with largest size. The normal vector of this square should point in the direction d_2^b:

$$d_2^b = |(node_{position}(sortedNodes(n+1,1)) - node_{position}(n))| + |d_1^b| \tag{3.34}$$

where the vector norm of d_1^b is denoted $|d_1^b|$. This square is rotated only slightly in the direction of the child node. As in the simple continuation mesh, new quadrilaterals are added between this square and the previous one constituting a hexahedron. After this, a new square is created at position d_3^b:

$$d_3^b = node_{position}(sortedNodes(n+1,1)) \tag{3.35}$$

The normal vector of this square should point in direction d_3^b:

$$d_3^b = node_{position}(sortedNodes(n+1,1) - node_{position}(n) \tag{3.36}$$

Quadrilaterals are added between square 2 and 3, and an additional hexahedron is formed.

The second largest child node is considered next. A new square is created at position p_4^b:

$$p_4^b = averageQuadPosition + branchSize(node_{size}(n)) \tag{3.37}$$

where *averageQuadPosition* is derived from the quadrilaterals added between square 2 and 3; the average position of the quadrilateral that is closest to the second largest child node. The new square is rotated such that its normal vector points in the direction d_4^b:

$$\begin{aligned} d_4^b &= |p_4^b - averageQuadPosition| + \\ &\quad |node_{position}(sortedNodes(n+1,2) - node_{position}(n)| \end{aligned} \tag{3.38}$$

A hexahedron is created between the previous hexahedron and quad 4. Finishing the branch, an additional quad is added at the position of the second largest child node, having normal vector pointing to d_5^b:

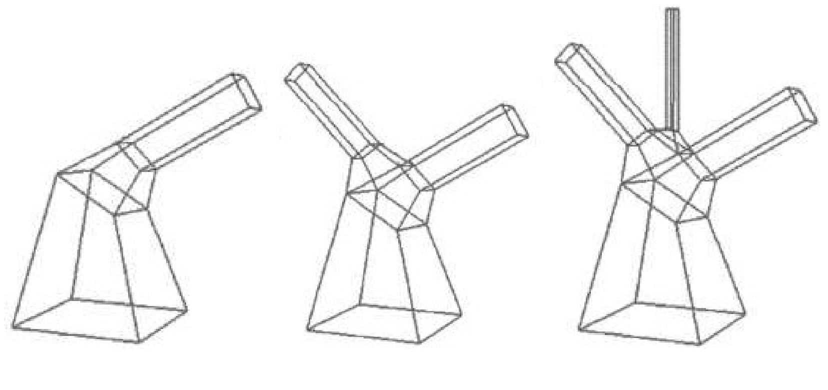
$$d_5^b = node_{position}(sortedNodes(n+1,2)) - node_{position}(n) \tag{3.39}$$

A hexahedron is formed between quad 4 and 5 at the end.

Hepatic vessels seldom branch into more than two sub-vessels, but our algorithm should still be able to handle such branches. If this is the case, these branches will be added as previously described. If the branching structure needs to be extended, this is done iteratively as shown in figure 3.38.

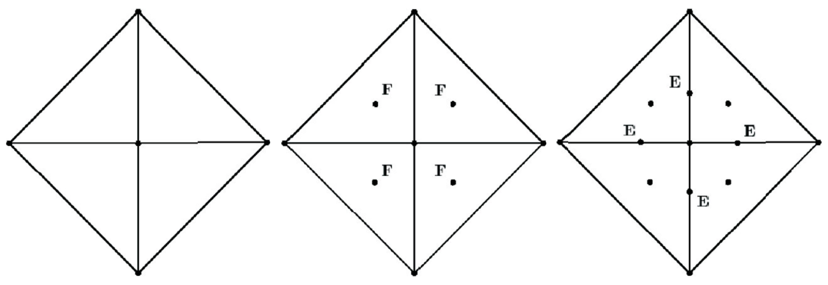
3.3.4 Subdivision

The polygon mesh derived in the previous section is smoothed by subdivision to create natural looking hepatic vessels. The Catmull-Clark subdivision



(a) (b) (c)

Figure 3.38: Branches are created by extra hexahedrons to connect the different sub-vessels. a) The largest sub-vessel is added first. b) The second largest sub-vessel is added. c) A third sub-vessel is added. Images courtesy of Skjermo and Eidheim (2005).



(a) (b) (c)

Figure 3.39: This figure shows the variables and the resulting vertices of one Catmull-Clark subdivision. a) The polygons to be subdivided. c) New vertices, face points. d) New vertices, edge points. Images courtesy of Jo Skjermo.

method (Catmull and Clark, 1978) was chosen, and an outline of this technique is given here.

In polygon subdivision, a finer polygon mesh is created and the vertices are filtered such that the second derivative of each vertex is small. The following steps are performed in Catmull-Clark subdivision (see figure 3.39):

1. New vertices, face points, are introduced at the centre of each face.

2. New vertices, edge points, are added at the centre of each edge.

3. New edges are added to connect the new edge points to adjacent face points.

The old vertices are manipulated afterwards as follows:

$$v = \frac{Q}{n} + \frac{2R}{n} + \frac{v(n-3)}{n} \tag{3.40}$$

where Q corresponds to the average of the new face points surrounding the vertex v, R is the average of the new edge points that is connected to v, and n is the number of edges that share v. Figure 3.40 shows an example subdivisions of a tree produced by the algorithm.

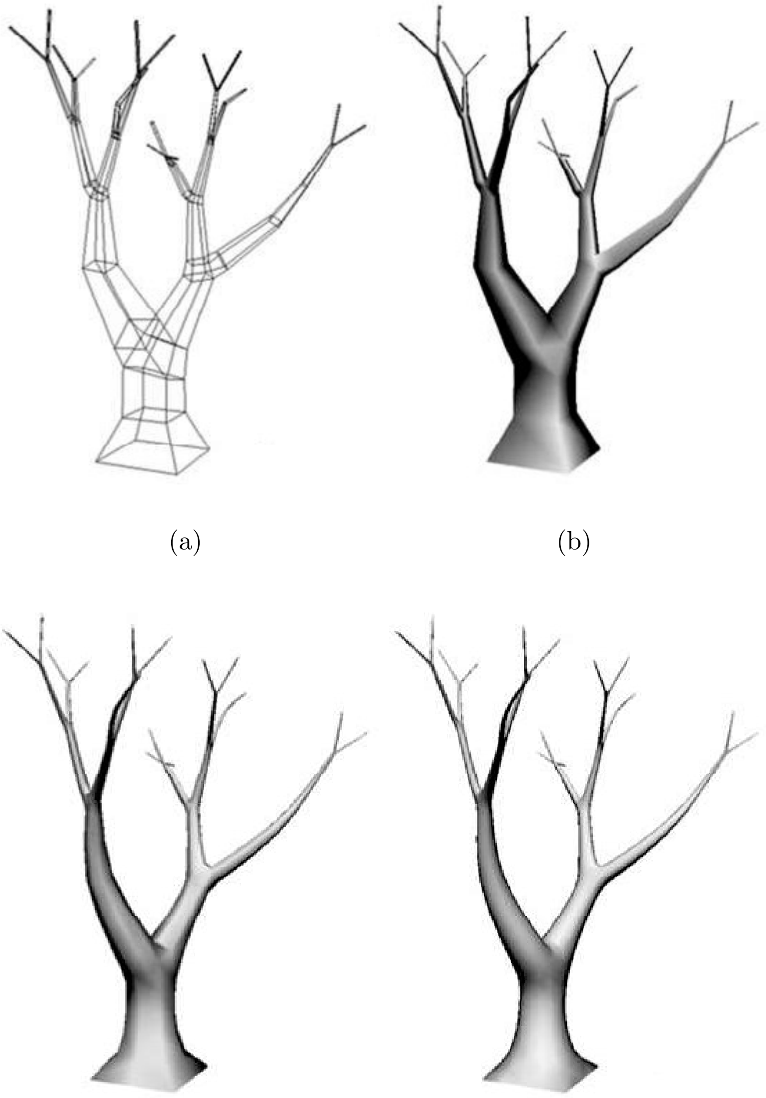

(a) (b) (c) (d)

Figure 3.40: Example subdivision of a tree. a) The polygon mesh produced by the algorithm described in 3.3. b) Shaded mesh. c) The result after one subdivision. d) The result after two subdivisions. Images courtesy of Skjermo and Eidheim (2005).

Chapter 4

Results and discussion

In this chapter we will present the results of the derived algorithms. The results from the 2D vessel reconstruction will be given first, and thereafter the results from the 3D vessel reconstruction. For each method, an example run of the algorithm will be given, including the results after each image analysis task.

4.1 2D vessel reconstruction

The processing algorithms will be demonstrated on four adjacent CT images shown in figure 4.1. At the end, a visualisation of the reconstructed vessels will be presented. We will also show the effects of the global search parameters, and present a likely vessel graph using a tuned set of parameters.

4.1.1 Preprocessing

First, we applied histogram equalisation as described in the previous chapter. The results are shown in figure 4.2.

4.1.2 Segmentation of the liver

Segmentation of the liver was performed prior to matched filtering to reduce edge effects near the boundary of the liver. Additionally, we needed a segmented liver in the following processing stages in order to extract the vessels properly.

There currently exists no satisfactory automatic solution to segment the liver, and this task is beyond the scope of this thesis. To segment the liver,

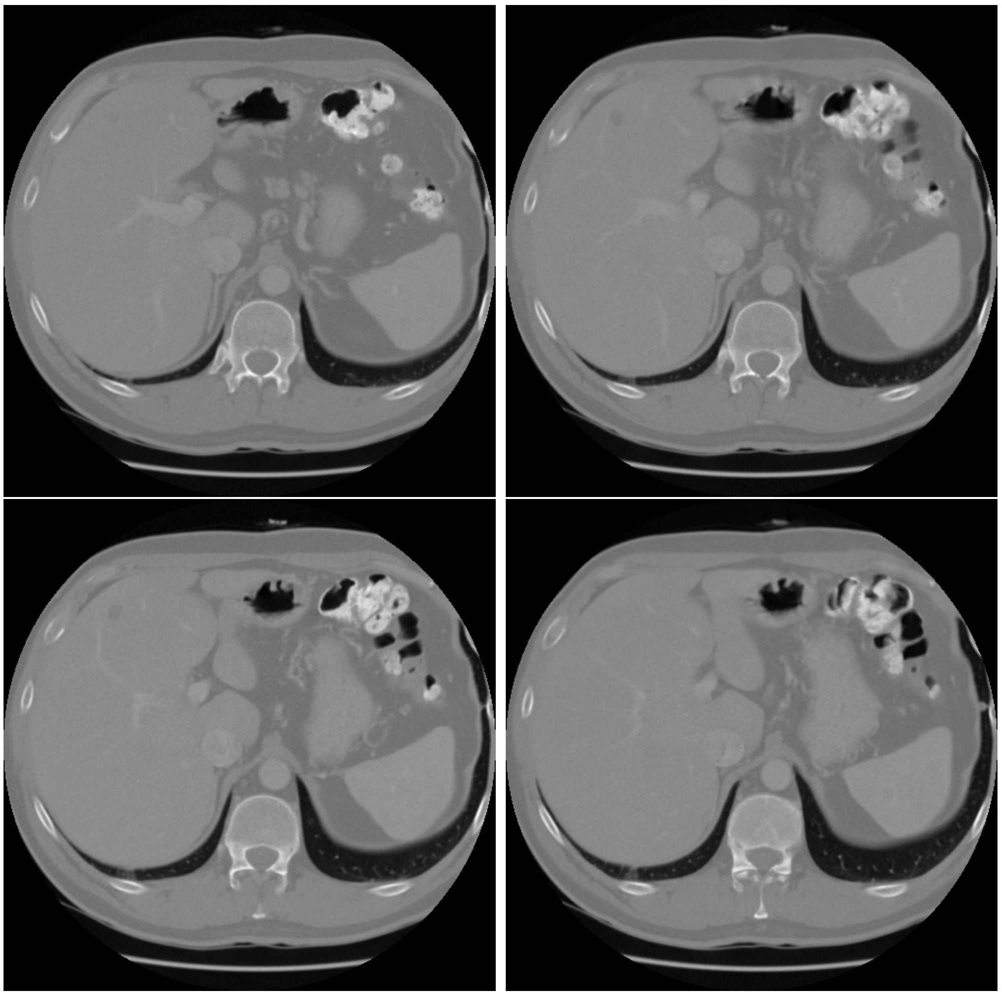


Figure 4.1: We follow the processing of these four CT images.

we instead used a semi-automatic segmentation method developed by Heuch (2003). An active contour model was used to segment the liver slice by slice. The contour needed to be guided by the user from time to time, but the contour model did ease the segmentation task. Regions outside of the segmented liver was set equal to 1 in order to avoid edge effects in the following matched filtering. Segmentation results from this application are presented in figure 4.3

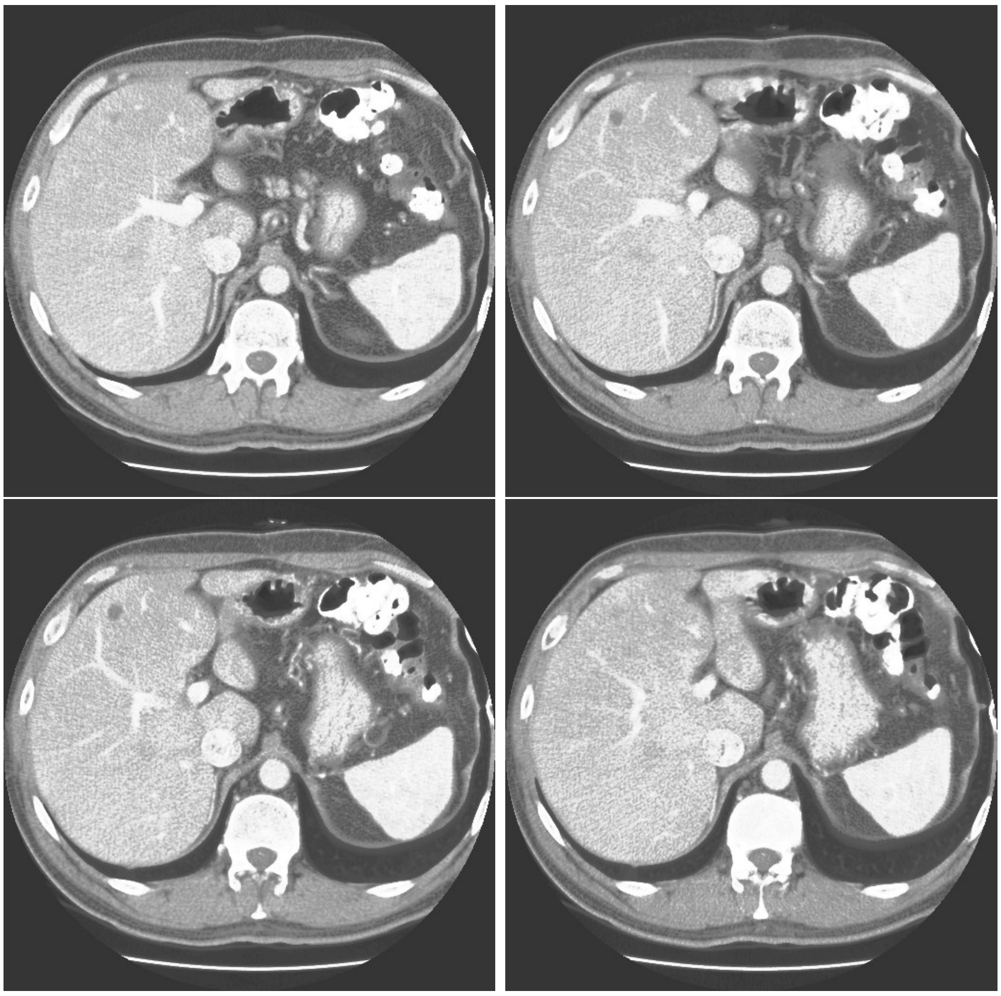


Figure 4.2: Histogram equalised images using eq. (3.1).

4.1.3 Matched filtering

Matched filtering is performed next. The choice of templates are crucial for the method to give good results. A Gaussian curve was used to produce the selection of Gaussian hill templates shown in figure 4.4. In our tests we rotated the template 6 degrees after each convolution, and the template was scaled 7 times. The results from filtering with each rotated and scaled template were summed.

Few additional details would arise by adding more templates to the algorithm, and by doing so the processing time would increase further. Using the templates described above, it took about 40 seconds to filter one CT image.

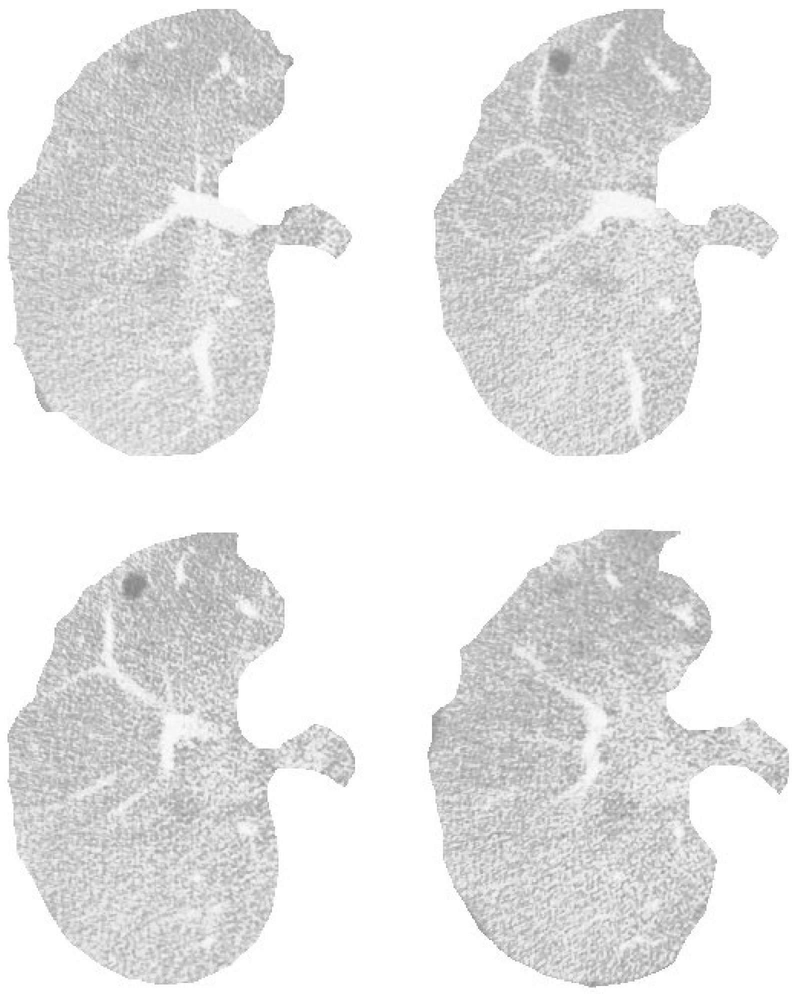
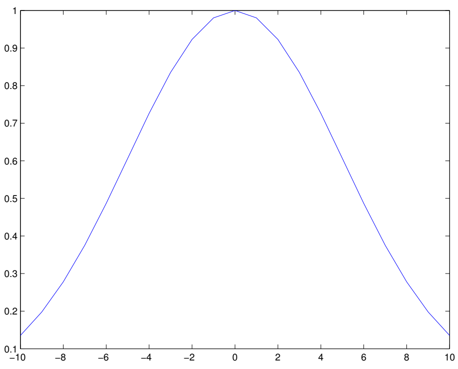


Figure 4.3: Masked liver results before matched filtering. The segmentation was done semi-automatically using the application designed in Heuch (2003).

Figure 4.5 shows the resulting matched filtered CT images.

4.1.4 Segmentation

Segmentation of the emphasised liver vessels is done by our slightly modified entropy thresholding algorithm combined with a second segmentation algorithm based on local mean and variance. After applying the knowledge based selection of the thresholds from Glombitza et al. (1999), the method


(a)

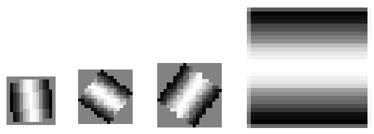

(b)

Figure 4.4: a) Gaussian curve used to design the filters in b). b) A selection of the templates used in our matched filter procedure. 203 templates were used, created by rotating and scaling a Gaussian hill template.

executes fully automatically. From the segmented images in figure 4.5, the algorithms produce the segmentation results shown in figure 4.6.

Further, results from filling, removal of isolated pixels, and rounding of square corners through morphological operators, can be seen in figure 4.7.

4.1.5 Classification

The next phase is the classification of the segmented vessels. There are two possible classifications, either the segment corresponds to a vessel running perpendicular to the CT slice, or it is a vessel running in an oblique angle to the slice. We call the former classification 1, and the latter 2. Figure 4.8 shows the resulting classification, where blue vessels are classified as 1 and red vessels have classification 2.

In our tests we received acceptable results using the following parameters:

1. If the rate between the minor and major axes of a segment is above 0.6, and the rate between the convex hull of the segment and the segment's area is higher than 0.85, classify the segment as 1.

2. Else, classify as 2.

Extensive testing on multiple CT slices were done before we found suitable values. Of course, the work was done manually and it is likely that the parameters can be better adjusted.

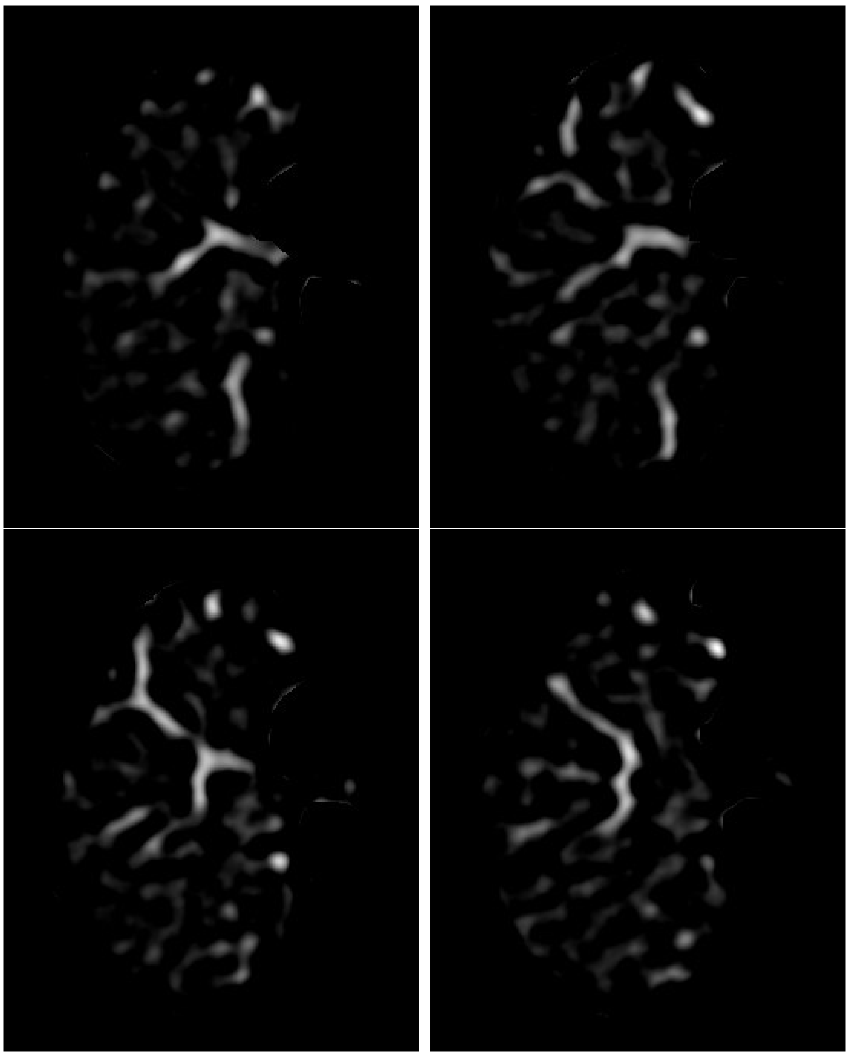


Figure 4.5: Results of using matched filtering on the images in figure 4.2.

4.1.6 Vessel centre extraction

The vessel centres are used to derive nodes and node positions for the vessel graph search. The centres are computed differently depending on the classification of the segments. As described in chapter 3, we use the mass centres for the segments classified as vessels running perpendicular to the CT slices, and the segments' skeletons for the remaining segments. Corrections are made to the centre points, and nodes are eventually created at intervals within the centres. Figure 4.9 shows the resulting node locations where they are represented by bright white pixels.

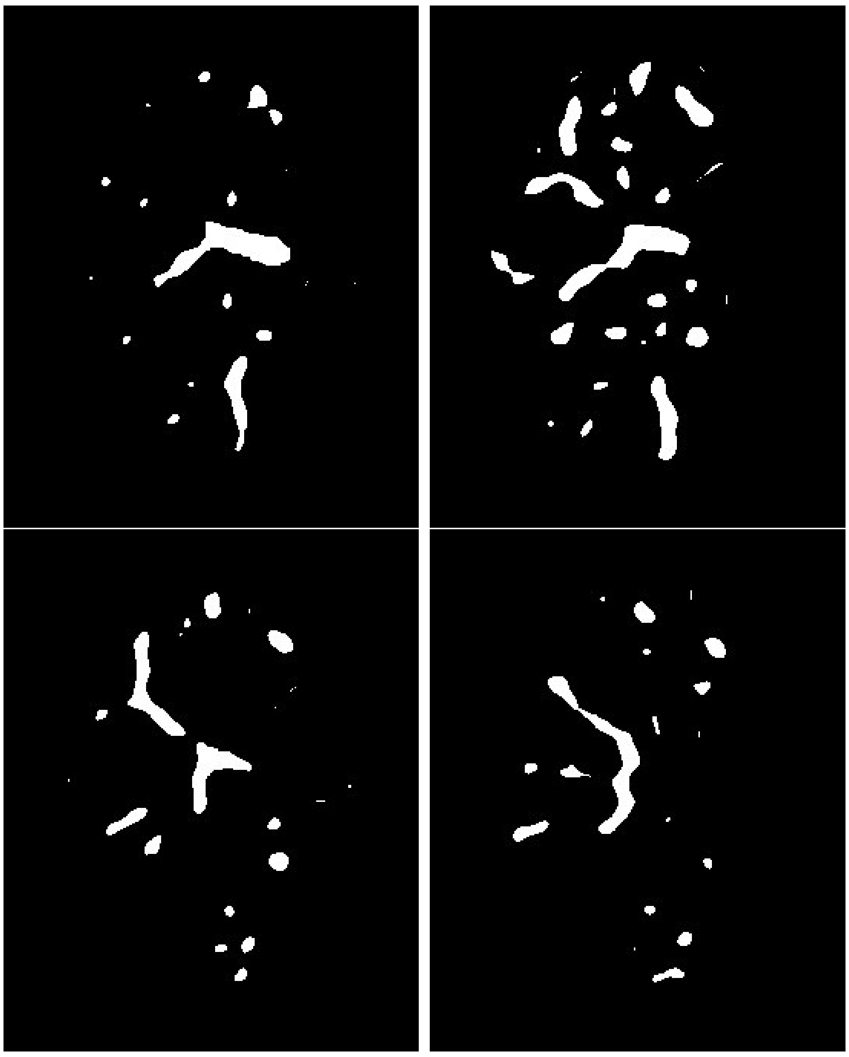


Figure 4.6: The resulting segmentation of the two segmentation algorithms added together. Segmentation based on local mean and variance was used to improve the segmentation of large hepatic vessels close to the boundary of the liver.

4.1.7 Vessel graph initialisation

The vessel graph is initialised based on the size of adjacent nodes between slices. The most probable interconnections are created first, and if a vessel loop is found, the last interconnection is removed before the remaining interconnections are added. Figure 4.10 shows a part of the initialised vessel graph visualised by the algorithm described in section 3.3. All vessels but

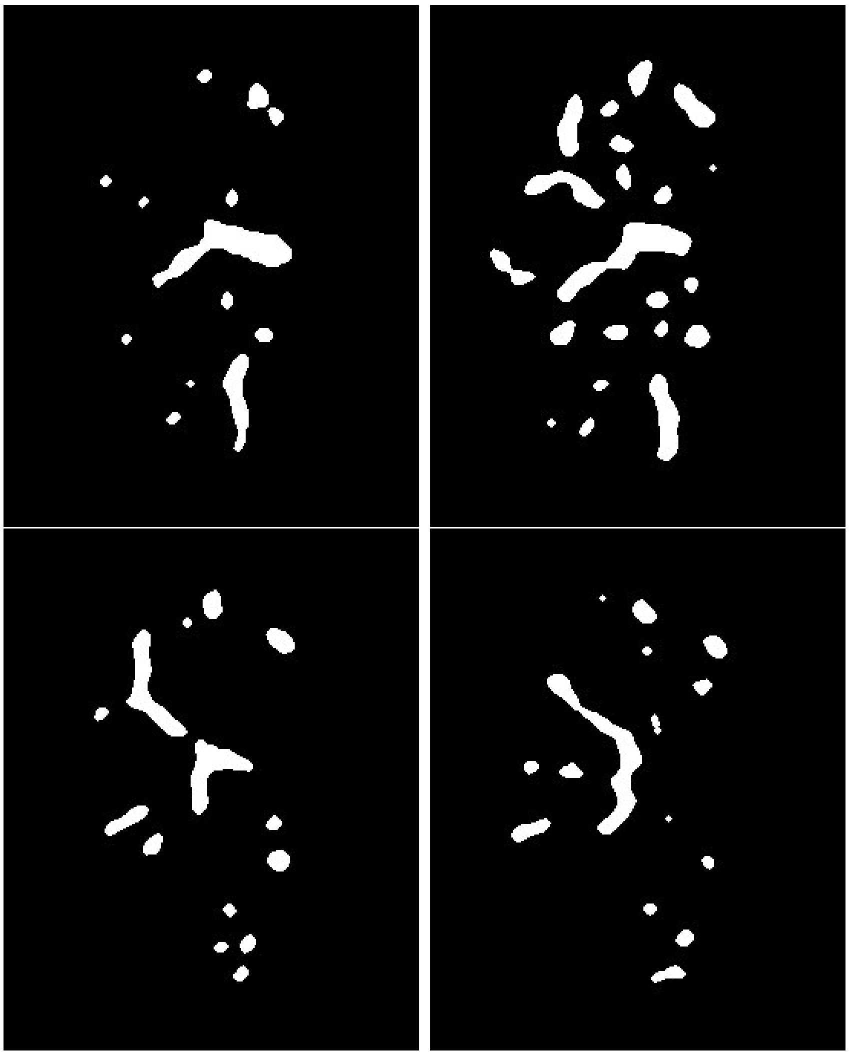


Figure 4.7: Blood vessels after being improved by morphological operators.

the portal vein were removed to increase the visibility of the result.

The application we have developed to visualise and verify our results is shown in figure 4.11. The outline of the liver is showned as well as the CT slices to compare the reconstructed vessels to the original data volume.

4.1.8 Global search

After the vessel graph initialisation, we applied algorithms to improve the graph through a global search mechanism. In the first executions, we used

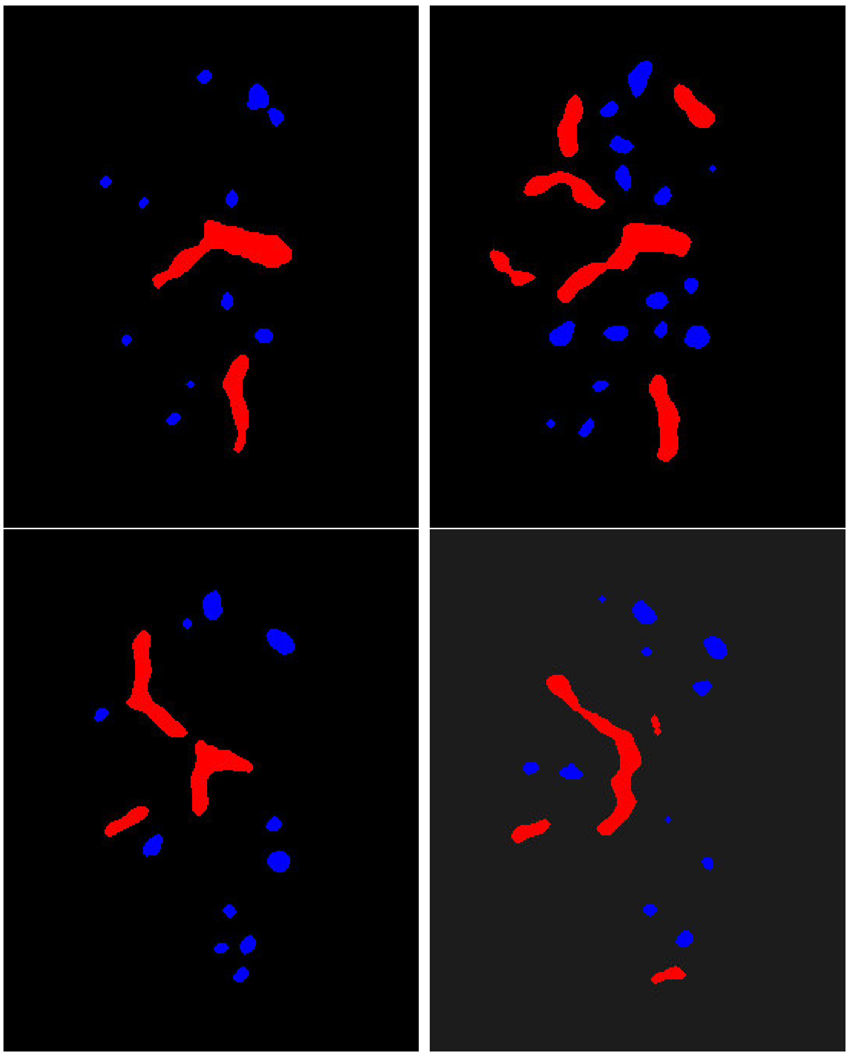


Figure 4.8: Vessel classification. Blue segments are vessels that runs perpendicular to the CT slice, while red segments corresponds to vessels running in an oblique angle to the slice.

three fitness functions only, namely the distance, dimension, and loop functions. The reason for this was that the initialisation was based on the same parameters that were used in these three functions. The resulting graph after applying the global search was equal to the initialisation graph.

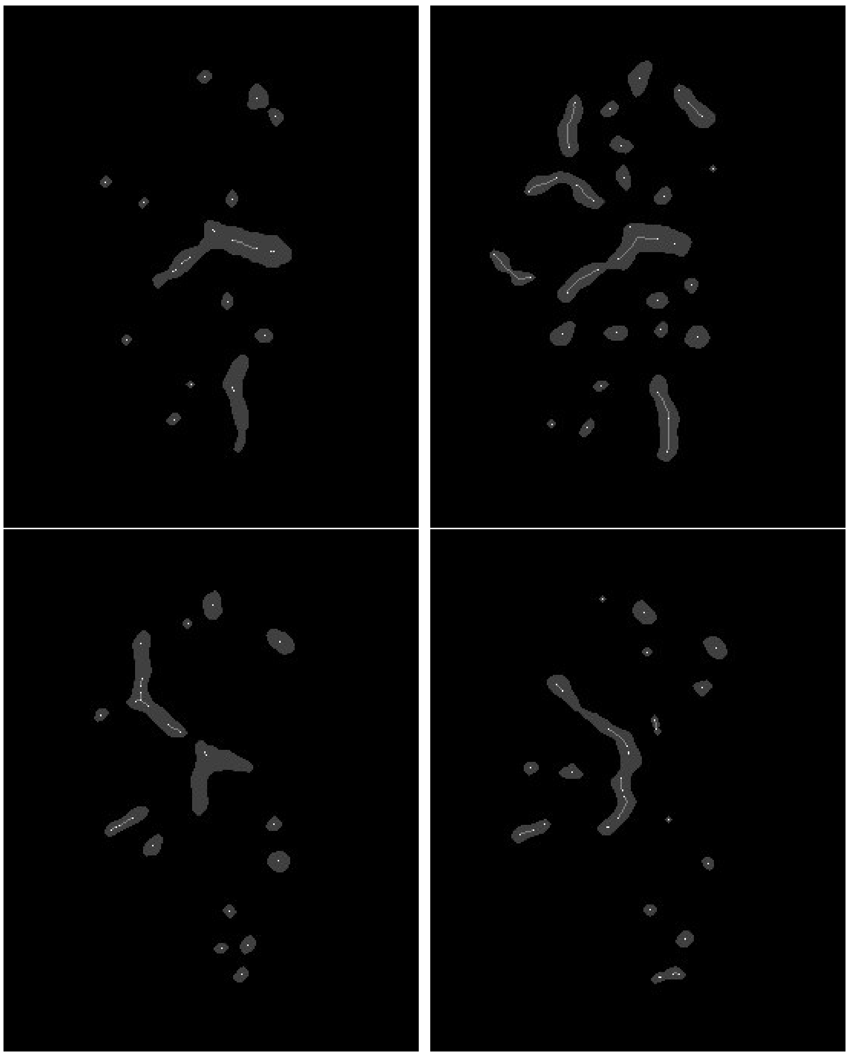


Figure 4.9: The results of the vessel centre calculations. Bright white pixels indicate node locations that will be used to create the vessel graph.

4.1.9 Result analysis

The segmentation algorithms result in a good approximation of the hepatic vessels within the liver. Both small and large vessels are found, and they have a continuous, circular shape. The method is also scalable as the CT imaging equipment becomes more advanced and produces CT images in higher resolutions. To segment larger vessels more templates are simply added in the matched filtering process. The thresholding algorithm remain the same.

The results from the global search using the distance, dimension, and loop

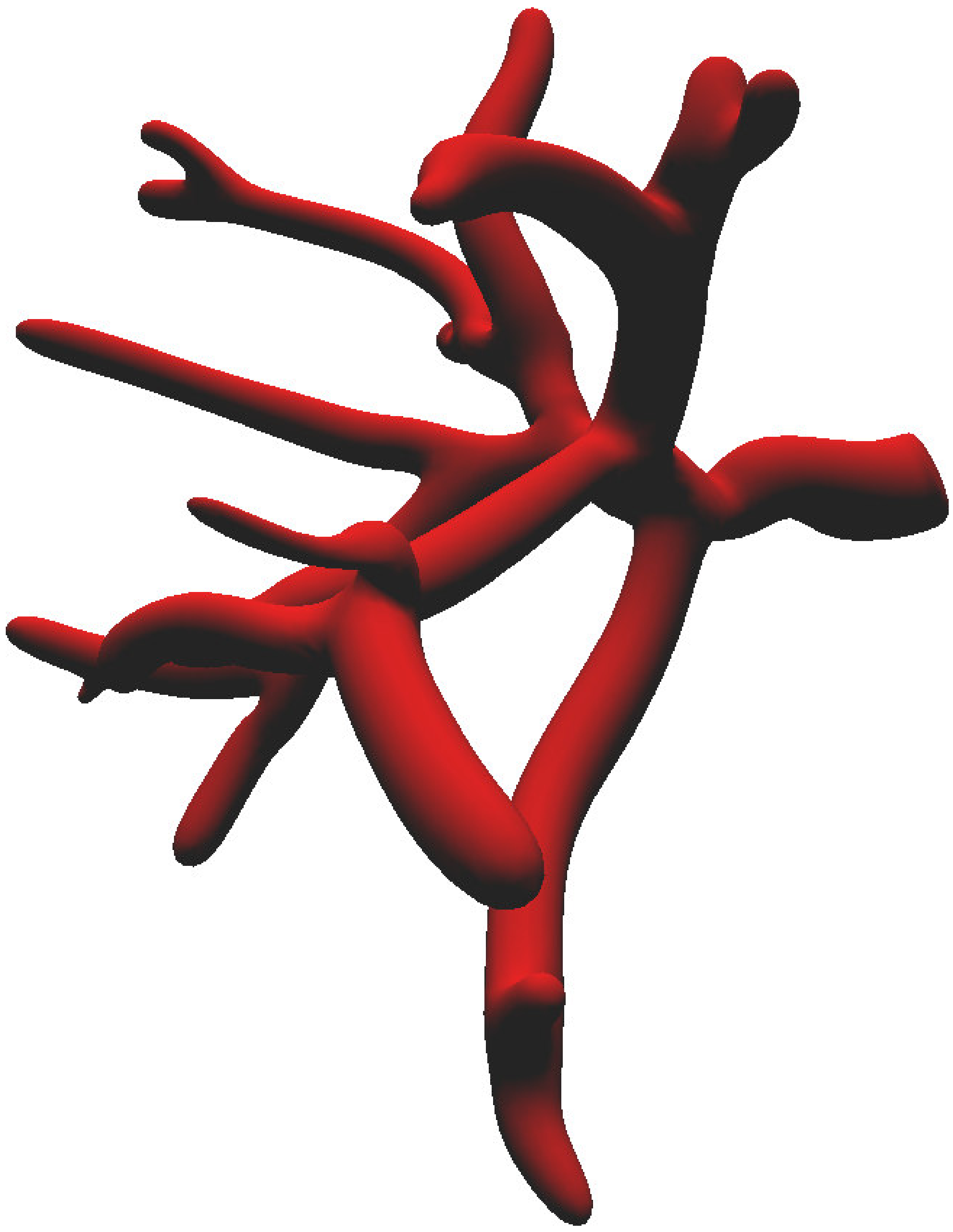


Figure 4.10: This figure shows a visualisation of the portal vein from the initialised vessel graph.

fitness functions are identical to the initialisation of the vessel graph. This proves that our initialisation algorithm is sound with respect to these vessel

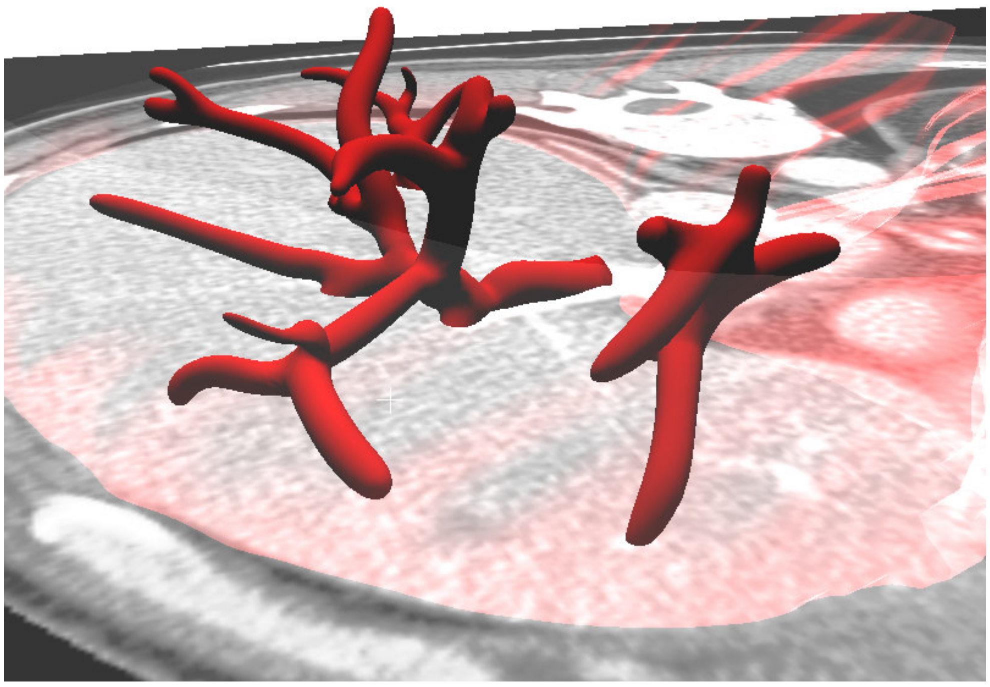


Figure 4.11: The application we have developed for visualisation and verification of the reconstructed vessels is shown here. The outline of the liver and the CT slices are visualised together with the vessels.

criteria.

Since our initial graphs did not contain any likely branches resulting in more than two child vessels, adding the branching fitness did not have any effect in the global search execution. On the other hand, when adding the last fitness function, curvature fitness, the initial graph changed. The results were not neccessary better, however, since weighting the different fitness functions proved difficult. Our conclusion was that the initial graphs were good approximations of the hepatic vessels, and that applying a global search to the algorithm only complicated the method more without necessarily giving a better result.

This method is dependant on numerous parameters. Extensive testing and adjustment of the parameters were done in order to achieve the results we have given in this section. Based on this observation, we must conclude that the method lacks the robustness and adaptability needed in clinical use.

4.2 3D vessel reconstruction

In this section we will present the results from the 3D vessel reconstruction methods. As in the previous section, the four CT images in figure 4.1 will be used as examples throughout the processing.

4.2.1 Preprocessing

The CT scan was first histogram equalised using our 3D equalisation algorithm. Second, matched filtering was applied in order to emphasise the hepatic vessels. The result from the matched filtering was next used in the first segmentation algorithm as described in section 3.2.2. Figure 4.12 shows the matched filtering result. 1800 templates were used in the filtering process. Due to the large number of templates, the algorithm took several hours to execute.

A second segmentation algorithm was applied to better segment large vessels positioned close to the boundary of the liver. Prior to this second algorithm, the CT scan was morphologically opened and filtered using anisotropic diffusion. The resulting diffused images after 10 iterations can be seen in figure 4.13.

4.2.2 Segmentation

Automatic segmentation was executed by the entropy thresholding method and the knowledge based threshold selection described in subsection 3.2.2. This segmentation algorithm was applied on both preprocessing results.

After the anisotropic diffusion result had been thresholded, each segmented pixel was kept if the local variance was below 0.01. The final segmentation was added with the thresholded matched filtering result as shown in figure 4.14.

4.2.3 Vessel centre extraction

Vessel centre points were extracted through our new 3D skeleton algorithm presented in section 3.2.3. Next, smaller branches were pruned since they are not interesting in the analysis of the hepatic vessels. Skeleton results before and after pruning can be seen in figure 4.15. Branches smaller than 10 pixels were pruned.

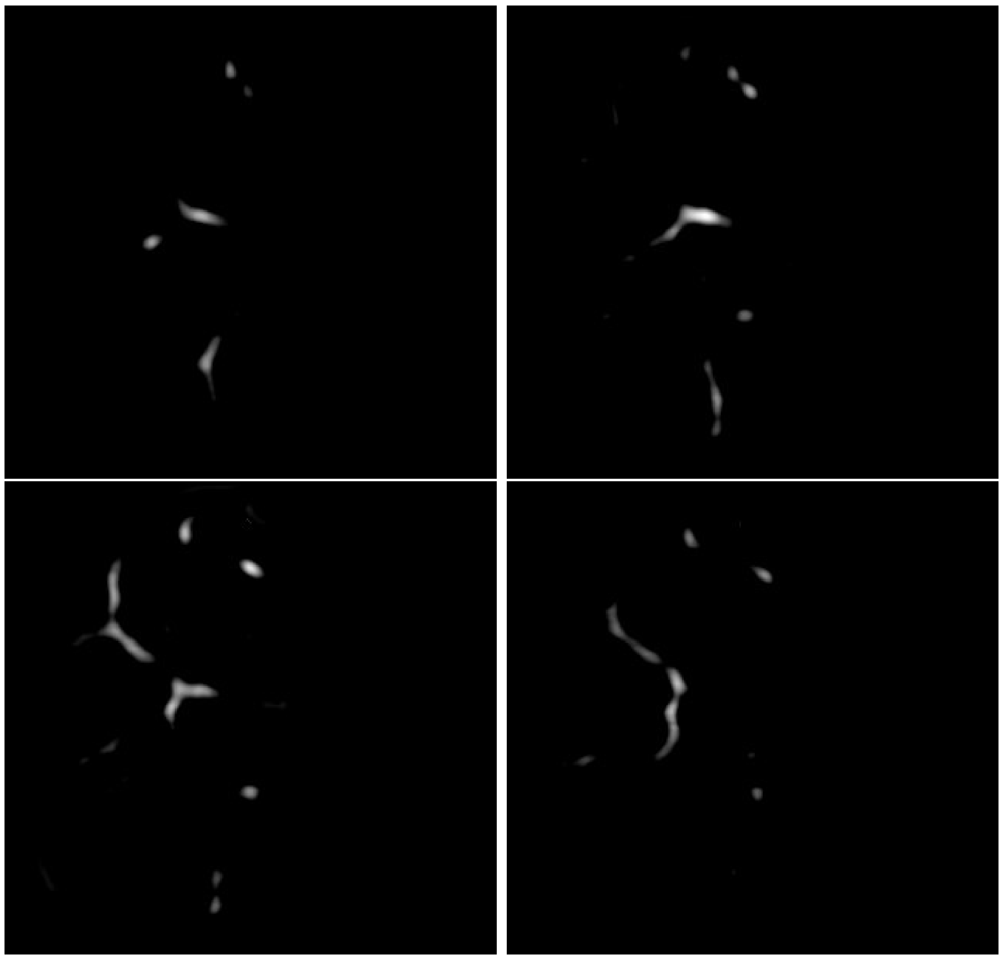


Figure 4.12: The resulting 3D matched filtering, where the slices shown in figure 4.1 are presented.

4.2.4 Vessel sizes

The vessel sizes are extracted from a distance transform as in the 2D vessel reconstruction. A dilated 3D distance transform is used that is scaled depending on the voxel ratio of the CT scan. Figure 4.16 shows example results at the slice positions previously used.

4.2.5 Vessel graph initialisation

As described in section 3.2.4, interconnections are mostly derived from the vessel skeletons. The vessel graph initialisation in this application consists

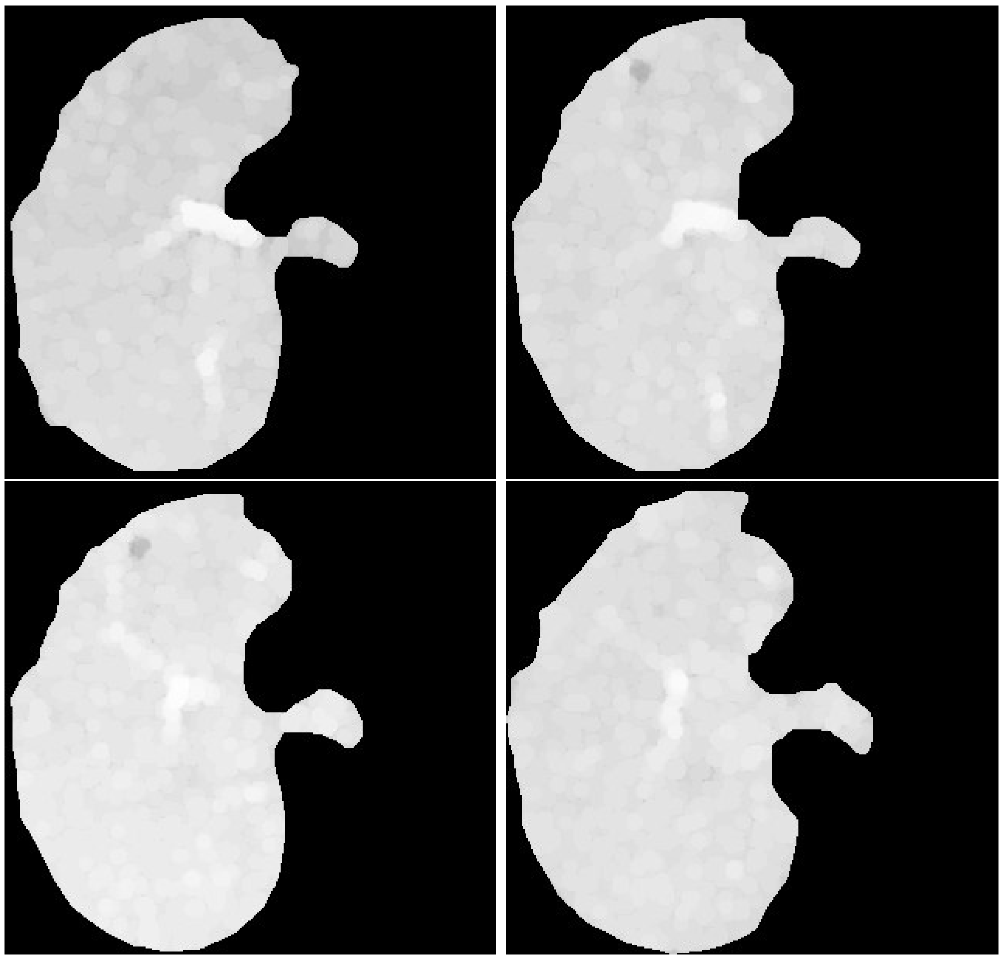


Figure 4.13: The results of the 3D anisotropic diffusion executed 10 iterations. The slices used in figure 4.1 are also presented here.

of interconnecting vessel structures that are likely to be parts of a larger structure. This algorithm is applied iteratively until no more vessel structures are likely to be joined. Figure 4.17 shows the initialisation of the previously presented dataset. Interconnections from the skeleton alone is shown first, and the derived additional interconnections are shown for comparison.

4.2.6 Local search

The local search method process each interconnection from the initial graph, and removes unlikely interconnections based on node size and the angle between the interconnections. The results from the local search can be seen in

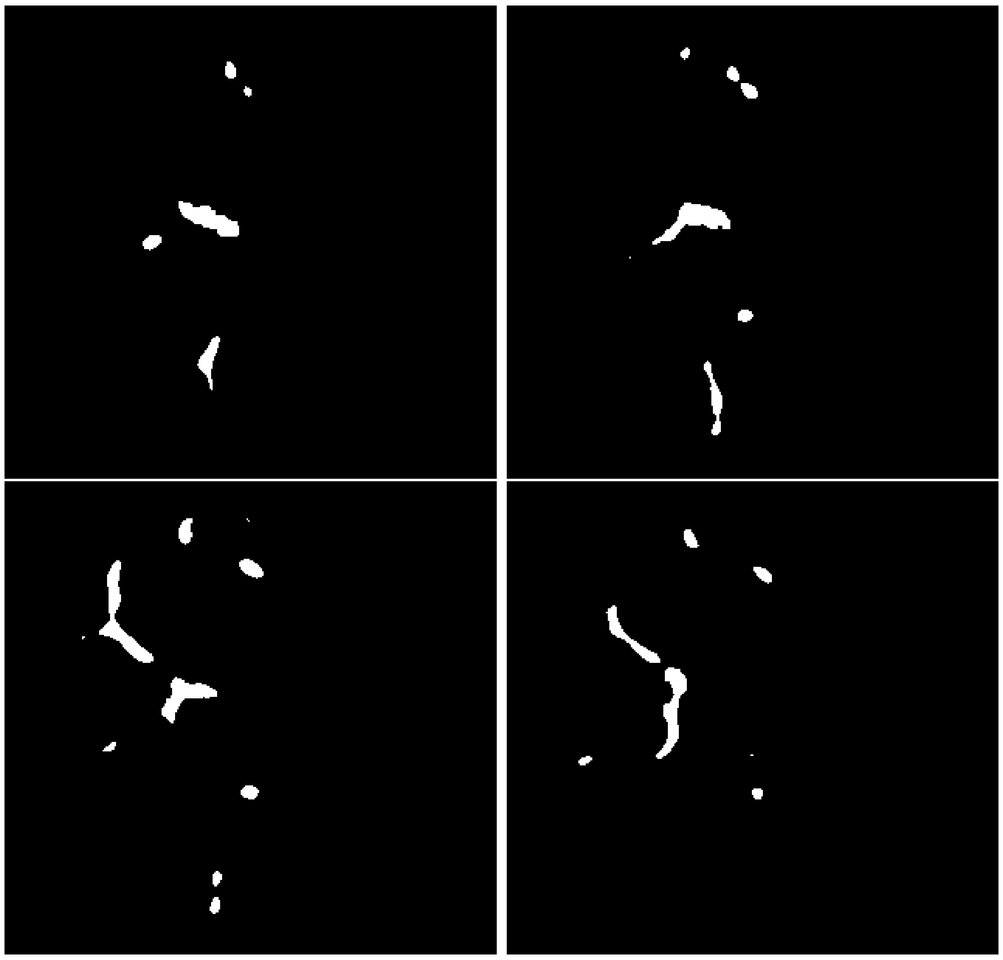


Figure 4.14: The final segmentation after adding both segmentation results together.

figure 4.18, and it can be seen that some of the interconnections have been removed. The portal vein from the same result is shown from a different angle in figure 4.21.

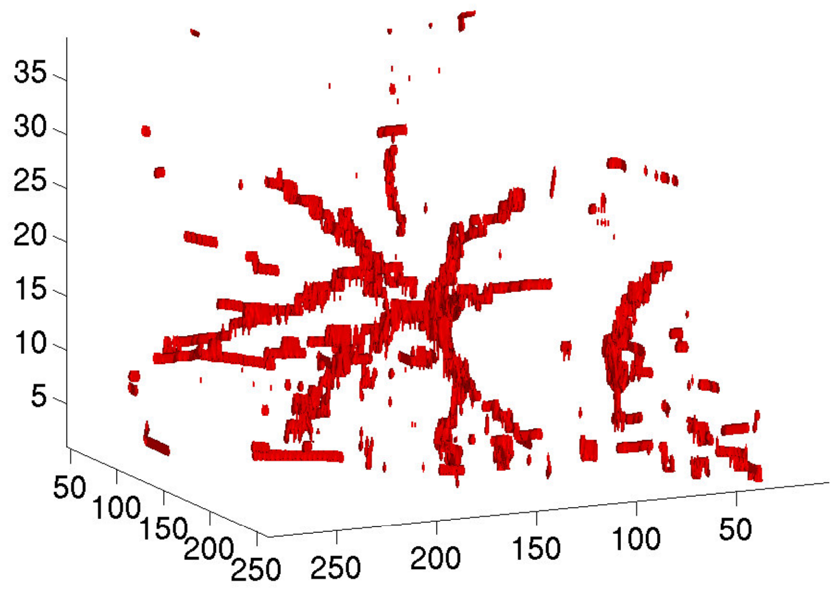
Results from two other datasets can be seen in figure ?? and ??.

4.2.7 Result analysis

The number of templates used in the matched filtering procedure is large and the results can take hours to compute. The runtime of the procedure may be reduced, however, by implementing the filtering on the GPU as previously mentioned.

The skeleton produced by the new algorithm results in a sound minimal representation of the segmented vessels. Almost all interconnections were derived from this skeleton. This gives a more solid initialisation of the vessel graph than that of the 2D vessel reconstruction method.

After applying the vessel graph corrections, a likely vessel reconstruction is made. The number of interconnected nodes are larger than in the results from the previous method, which means that more of the vessel structure is reconstructed. Moreover, the results seem more likely when compared to the CT slices by hand.



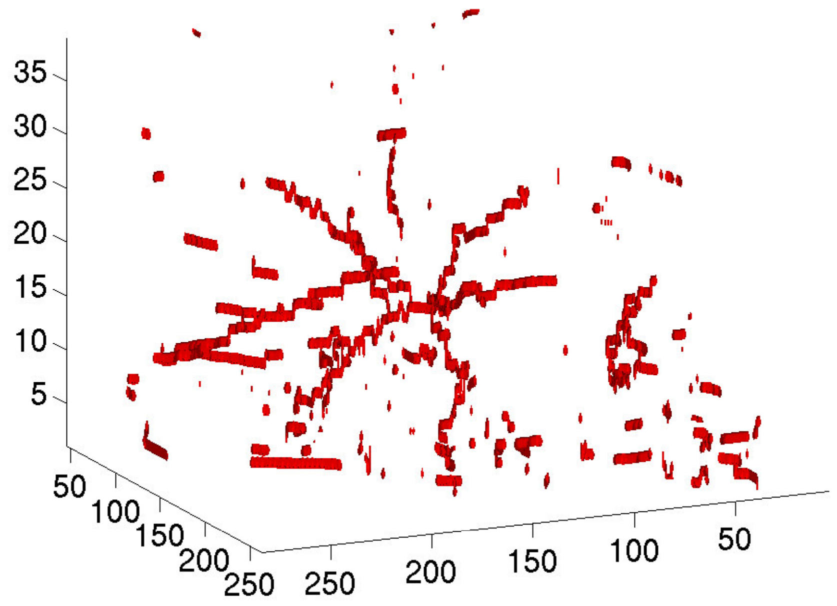


Figure 4.15: The vessel centre points were extracted from a 3D skeleton. a) The resulting 3D skeleton. b) The result after pruning smaller branches from b).

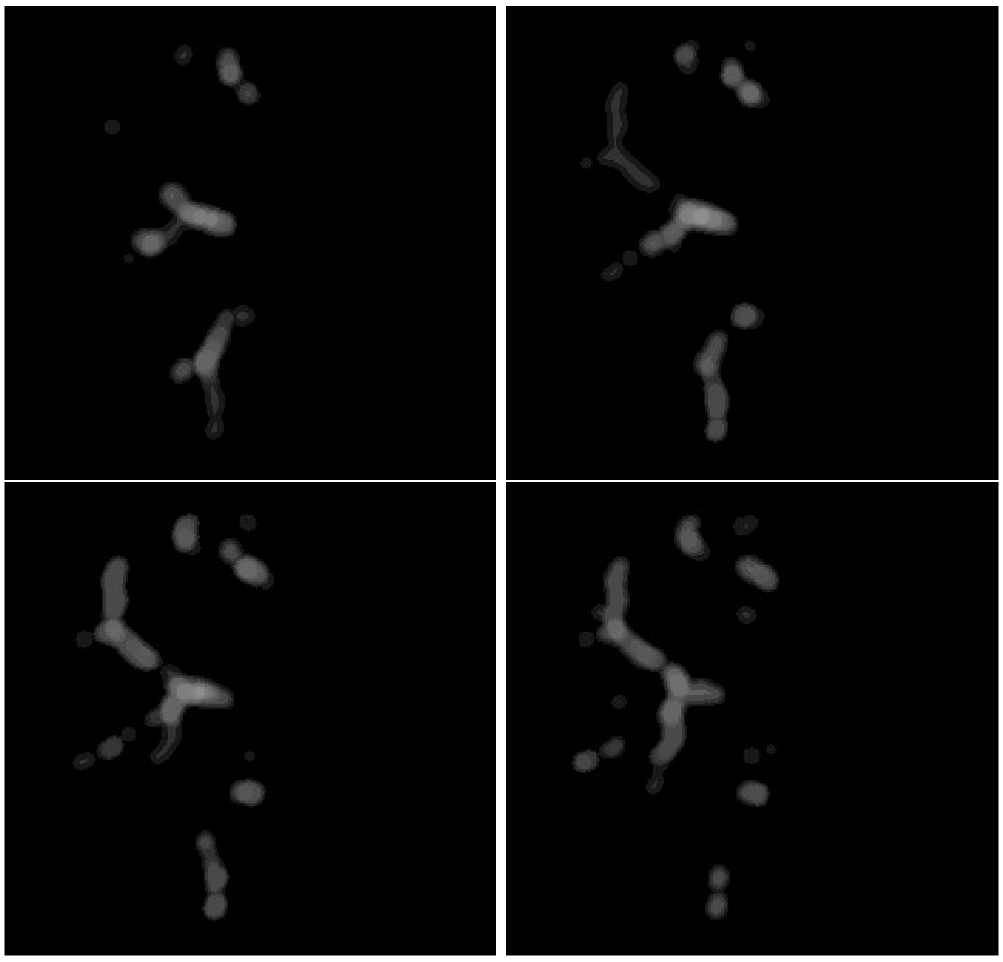


Figure 4.16: A dilated 3D distance transform were computed to find the vessel sizes.

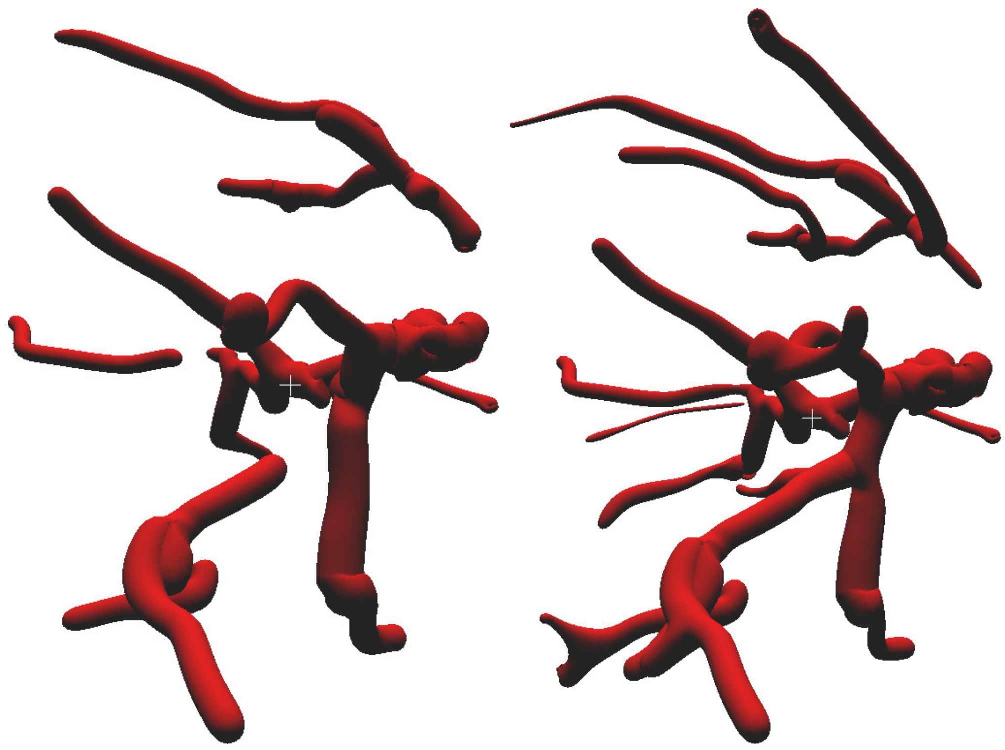


Figure 4.17: Initialisation of the vessel graph resulting from 3D processing of the CT data. a) The vessel interconnections derived directly from the 3D skeleton algorithm. b) The resulting initialisation where interconnections are added between nearby nodes that are similar in size.

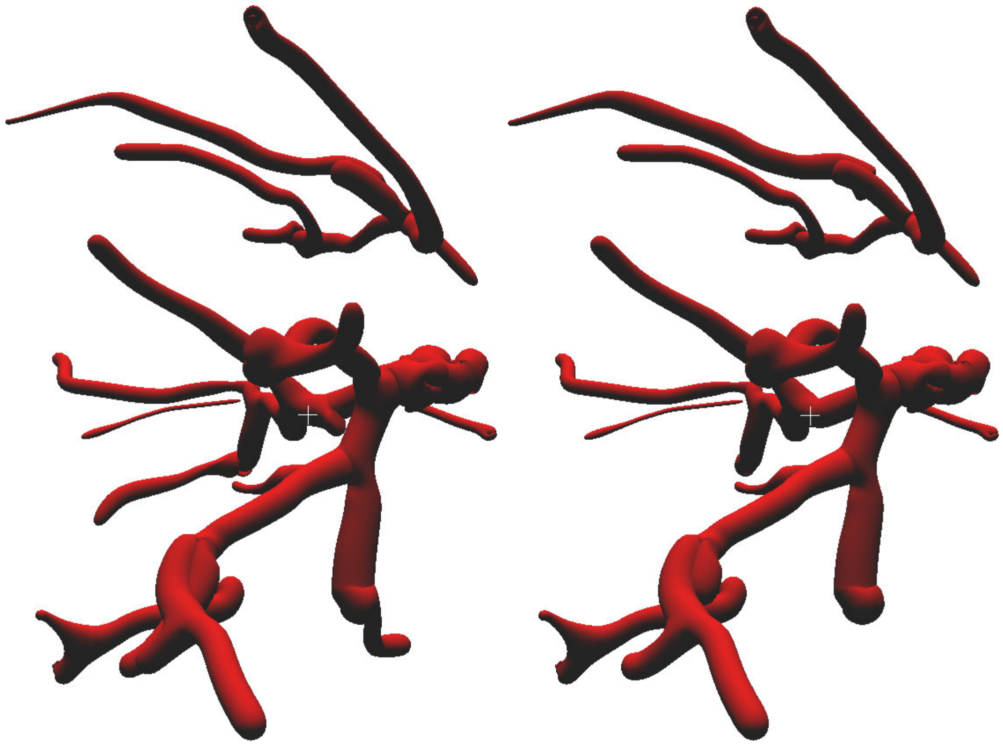


Figure 4.18: Resulting vessel graph after applying a local search to remove unlikely interconnections. a) Initial vessel graph. b) Final vessel graph after applying local search.

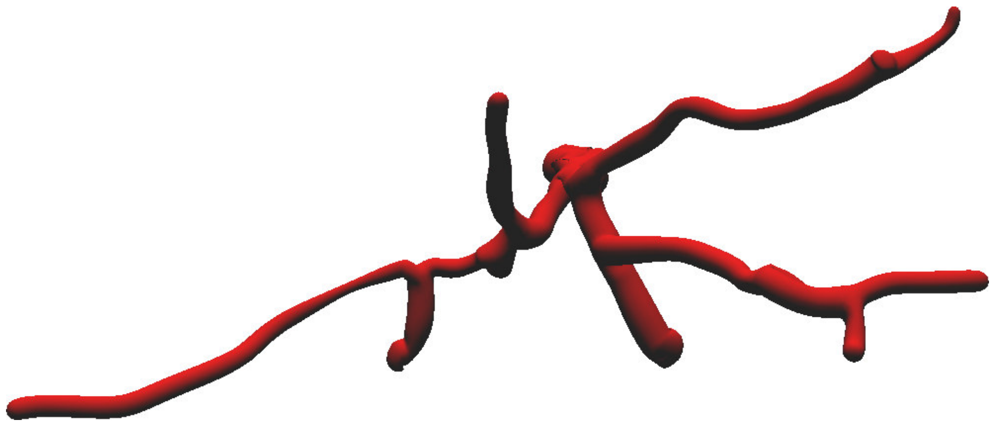


Figure 4.19: The portal vein from the results also shown in figure 4.18 is shown here from a different angle.

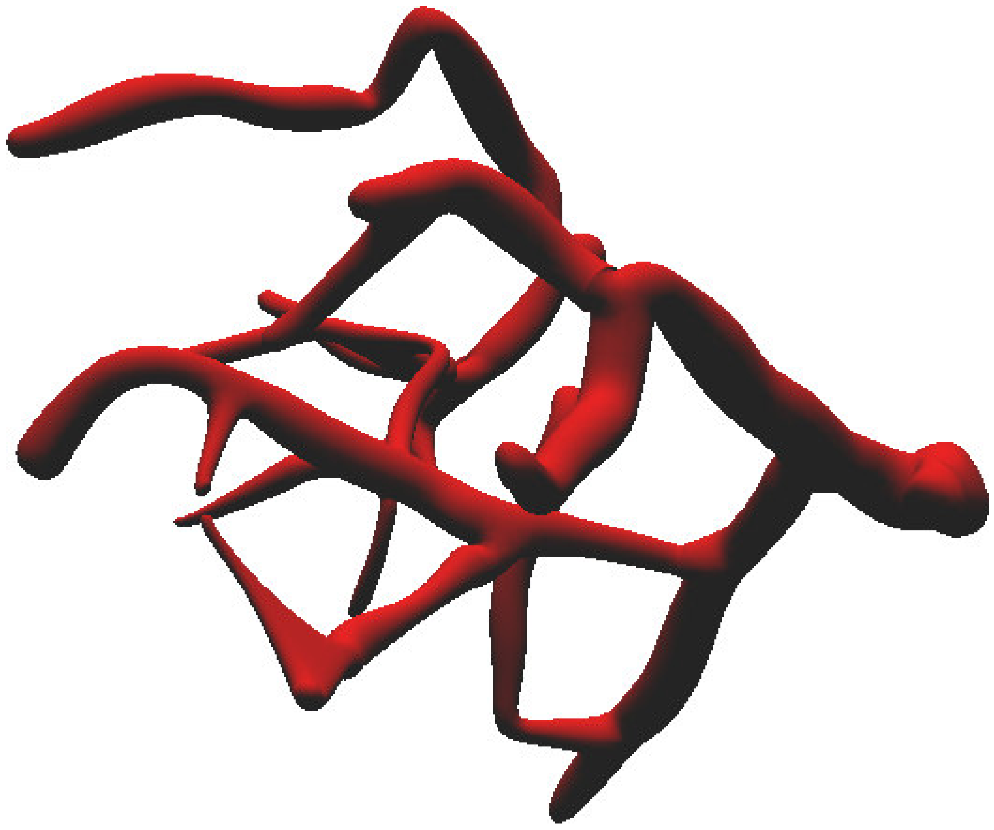


Figure 4.20: Resulting reconstruction of the portal vein from CT scan 2.

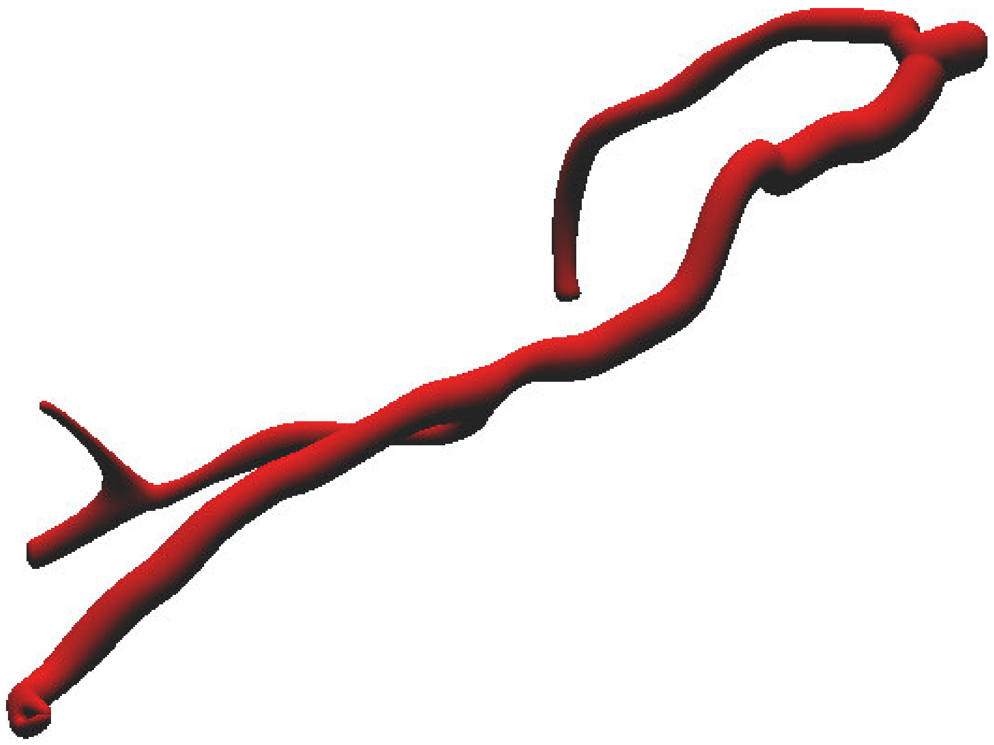


Figure 4.21: Resulting reconstruction of a part of the hepatic vein from CT scan 3.

Chapter 5

Conclusion and future work

The main motivation behind this thesis was to ease the planning phase prior to a liver surgery, and to improve the current presentations of the hepatic vessels from CT scans. In more detail, an interactive representation of the hepatic vessels within the liver was to be developed. Through image processing algorithms the vessels should be emphasised, segmented, and a likely vessel graph should be derived. A polygon mesh should be constructed from this graph and finally visualised for inspection.

5.1 Conclusion

We have developed two distinct methods to create a realistic vessel graph from a CT scan. The first method works on each CT slice individually before the graph is built. Additional algorithms have been implemented compared with the work done in Omholt-Jensen (2002). Most important are segmentation based on local variance to better segment larger vessels near the boundary of the liver, classification of the segments combined with shape skeletons to better extract the vessel centre points, and corrections to the derived centre points based on information in adjacent CT slices. The initialisation of the graph and the cost functions used in the global search also differs from Omholt-Jensen (2002). Unfortunately, this method rely on numerous parameters, which makes the method highly vulnerable to bad parameter choices. Because of this, the method lacks the robustness and adaptability needed in clinical use.

The second method was derived to reduce this dependability on parameters, and additionally take advantage of the information between neighbouring CT slices throughout the processing. The algorithms were extended to 3D and applied to the whole CT scan at once. In addition to the thresholding of the 3D matched filtered result, 3D anisotropic diffusion was applied and thresholded using the same thresholding algorithm. The two segmentation results were added together to improve the overall segmentation. A new 3D skeleton algorithm was derived to extract the vessel centres based on a simple point definition previously derived. The algorithm produces more usable skeletons for our application than other skeleton algorithms studied. Moreover, a local search was implemented that removed some of the unlikely vessel interconnections in a matter of seconds. This represent a major improvement to the global search implemented in the first method, which could take hours to execute.

A realistic polygon mesh algorithm has been implemented that resulted in natural looking vessel branches. The visualisation of the hepatic vessels has been implemented in an interactive application where the vessels can be studied from every angle and position. The outline of the liver can be shown in relation to the vessels, and the CT images can be viewed to compare the results to the original dataset.

The results from both methods have been compared, and we conclude that the second method results in a more likely reconstruction of hepatic vessels from CT scans. The number of interconnected nodes were larger using this method, and the graph corresponded better to the inspected CT slices when studied by hand.

The results have been visually inspected by a radiologist and the response was positive, but an empirical study on the correctness of the vessel graph has not been performed. Before this application is ready for production use, this study needs to be completed, but the technique shows great promise and may eventually be used clinically.

5.2 Future work

While our results are promising, there is still work that needs to be done. We propose the following improvements:

- Our results need to be tested and verified by experienced radiologists.
- Improved cost functions can be derived, as well as more appropriate function weights.
- Our preprocessing phase could be improved to extract even more information from the CT images.
- In the 3D reconstruction algorithm, we have a poor mechanism for resolving vessel loops. A good result is not guaranteed, and a better

solution needs to be derived before this technique is used in clinical applications.

- Without significant change, our algorithm can be adopted to similar problems, for instance vessel segmentation in the eye fundus.
- As medical modalities improve, higher resolution CT images can be obtained. This will enable us to find even smaller blood vessels using our algorithm. As previously mentioned, we simply need to add more templates to our matched filtering algorithms.
- The preprocessing algorithms can be more efficiently executed on the GPU of modern graphics cards.
- Tumours should be segmented and visualised in relation to the liver and the liver vessels.

Bibliography

Aykac, D., Hoffman, E. A., McLennan, G., and Reinhardt, J. M. (2003). Segmentation and analysis of the human airway tree from three-dimensional x-ray ct images. *IEEE Transactions on Medical Imaging*, 22(8):940–950.

Banzhaf, W., Nordin, P., Keller, R. E., and Francone, F. D. (1998). *Genetic Programming: An Introduction*. Morgan Kaufmann Publishers, Inc.

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Bertrand, G. and Malandain, G. (1994). A new characterization of three-dimensional simple points. *Pattern Recognition Letters*, 15(3):196–175.

Bloomenthal, J. (1985). Modeling the mighty maple. *Computer Graphics*, 19(3):305–311.

Bloomenthal, J. and Wyvill, B. (1990). Interactive techniques for implicit modeling. *Computer Graphics*, 24(2):109–116.

Blum, H. (1967). A transformation for extracting new descriptors of shape. In Wathen-Dunn, W., editor, *Models for the Perception of Speech and Visual Form*. MIT Press, Cambridge, Mass.

Borgefors, G., Nystrom, I., and Baja, G. (1999). Computing skeletons in three dimensions. *Pattern Recognition*, 32:1225–1236.

Catmull, E. and Clark, J. (1978). Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355.

Chaudhuri, S., Chatterjee, S., Katz, N., Nelson, M., and Goldbaum, M. (1989). Detection of blood vessels in retinal images using two-dimensional matched filters. *IEEE Transactions on Medical Imageing*, 8(3):263–269.

Choi, H. S., Haynor, D. R., and Kim, Y. (1991). Partial volume tissue classification of multichannel magnetic resonance images - a mixel model. *IEEE Transactions on Medical Imaging*, 10(3):395–407.

Chung, D. H. and Sapiro, G. (2000). Segmenting skin lesions with partial-differential-equations-based image processing algorithms. *IEEE Transactions on Medical Imaging*, 19(7):763–767.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603 – 619.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. John Wiley & Sons.

Eidheim, O. C., Aurdal, L., Omholt-Jensen, T., Mala, T., and Edwin, B. (2004a). Interconnecting segmented hepatic vessels in adjacent ct slices. *NOBIM*, pages 91–97.

Eidheim, O. C., Aurdal, L., Omholt-Jensen, T., Mala, T., and Edwin, B. (2004b). Segmentation of liver vessels as seen in mr and ct images. *Computer Assisted Radiology and Surgery*, pages 201–206.

Eidheim, O. C., Skjermo, J., and Aurdal, L. (2005). Real-time analysis of ultrasound images using gpu. *Computer Assisted Radiology and Surgery*.

Felkel, P., Fuhrmann, A., Kanitsar, A., and Wegenkittl, R. (2002a). Surface reconstruction of the branching vessels for augmented reality aided surgery. In *Analysis of Biomedical Signals and Images*, volume 16, pages 252–254. BIOSIGNAL 2002.

Felkel, P., Kanitsar, A., and Fuhrmann, A. L. (2002b). Surface models of tube trees. *Tech. Rep. TR VRVis 2002 008, VRVis*.

Fernández, G., Bischof, H., and Beichel, R. (2003). Nonlinear filters on 3d ct imaging - bilateral filter and mean shift filter. *Computer Vision - CVWW'03*.

Fok, Y.-L., Chan, J. C. K., and Chin, R. T. (1996). Automated analysis of nerve-cell images using active contour models. *IEEE Transactions on Medical Imaging*, 15(3):353–368.

Frangi, A., Rueckert, D., Schnabel, J., and Niessen, W. (2002). Automatic construction of multiple-object three-dimensional statistical shape models:

application to cardiac modeling. *IEEE Transactions on Medical Imaging*, 21(9):1151–1166.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distribution, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAM1-6, No. 6:721–741.

Glombitza, G., Lamadé, W., Demiris, A. M., Göpfert, M.-R., Mayer, A., Bahner, M. L., Meinzer, H.-P., Richter, G., Lehnert, T., and Herfarth, C. (1999). Virtual planning of liver resections: image processing, visualization and volumetric evaluation. *International Journal of Medical Informatics*, 53:225–237.

Goldberg, D. E. (1998). *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley.

Gomberg, B., Saha, P., Song, H. K., Hwang, S., and Wehrli, F. (2000). Topological analysis of trabecular bone mr images. *IEEE Transactions on Medical Imaging*, 19(3):166–174.

Gonzalez, R. C. and Woods, R. E. (2002). *Digital Image Processing.* Prentice Hall, second edition.

GPGPU-group (2005). General-purpose computations using graphics hardware. `http://www.gpgpu.org`.

Guo, Z. and Hall, R. W. (1989). Parallel thinning with two-subiteration algorithms. *Communications of the ACM*, 32(3):359–373.

Haris, K., Efstratiadis, S. N., Maglaveras, N., Pappas, C., Gourassas, J., and Louridas, G. (1999). Model-based morphological segmentation and labeling of coronary angiograms. *IEEE Transactions on Medical Imaging*, 18(10):1003–1015.

Heuch, H. (2003). Segmentation of the liver from mr and ct images. Master's thesis, Norwegian University of Science and Technology.

Hokland, J. (2002). Introduksjon til bayesiansk bildeanalyse. *Kompendium i fag SIF8068 - Statistisk Bildeanalyse og Læring*, pages 1–9.

Hough, P. V. C. (1959). Machine analysis of bubble chamber pictures. *International Conference on High Energy Accelerators and Instrumentation, CERN.*

Inaoka, N., Suzuki, H., and Fekuda, M. (1992). Hepatic blood vessel recognition using anatomical knowledge. In Loew, M. H., editor, *Medical Imaging VI: image processing*, pages 509–513. SPIE.

Kapur, J. N., Sahoo, P. K., and Wong, A. K. C. (1985). A new method for gray-level picture thresholding using the entropy of the histogram. *Computer Vision, Graphics, and Image Processing*, 29:273–285.

Kass, M., Witkin, A., and Terzoploulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, pages 321–331.

Kelemen, A. and Székely, G. (1999). Elastic model-based segmentation of 3-d neuroradiological data sets. *IEEE Transactions on Medical Imaging*, 18(10):828–839.

Kelemen, A., Székely, G., and Gerig, G. (1998). Three-dimensional model-based segmentation of brain mri. In *Proceedings of the Workshop on Biomedical Image Analysis*, pages 4–13.

Krivanek, A. and Sonka, M. (1998). Ovarian ultrasound image analysis: follicle segmentation. *IEEE Transactions on Medical Imaging*, 17(6):935–944.

Lam, L., Lee, S.-W., and Suen, C. (1992). Thinning methodologies-a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885.

Lluch, J., Vicent, M., Fernandez, S., Monserrat, C., and Vivo, R. (2001). Modelling of branched structures using a single polygonal mesh. In *IASTED International Conference on Visualization, Imaging, and Image Processing.*

Lobreget, S., Verbeek, P., and Groen, F. (1980). Three-dimensional skeletonization: principle and algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:75–77.

Lobregt, S. and Viergever, M. A. (1995). A discrete dynamic contour model. *IEEE Transactions on Medical Imaging*, 14(1):12–24.

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. In *Computer Graphics (Proceedings of SIGGRAPH '87)*, volume 21, pages 163–169.

Ma, C. and Sonka, M. (1996). A fully parallel 3d thinning algorithm and its applications. *Computer Vision and Image Understanding*, 64:420–433.

Malandain, G. and Bertrand, G. (1992). Fast characterization of 3d simple points. In *11th IEEE International Conference on Pattern Recognition*, pages 232–235.

Malandain, G., Bertrand, G., and Ayache, N. (1993). Topological segmentation of discrete surfaces. *International Journal of Computer Vision*, 10(2):183–197.

Malladi, R. and Sethian, J. A. (1996). A unified approach to noise removal, image enhancement, and shape recovery. *IEEE Transactions on Image Processing*, 5(11):1554–1568.

Malladi, R., Sethian, J. A., and Vemuri, B. C. (1995). Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175.

Martinez-Perez, M., Hughes, A., Stanton, A., Thom, S., Bharath, A., and Parker, K. (1999). Segmentation of retinal blood vessels based on the second directional derivative and region growing. *Proceedings of the International Conference on Image Processing*, 2:173–176.

McInerney, T. and Terzopoulos, D. (2000). Deformable models. In Bankman, I., editor, *Handbook of Medical Imaging*, pages 127–145. Academic Press.

Murray, C. D. (1927). A relationship between circumference and weight in trees and its bearing in branching angles. *Journal of General Phyiol.*, 9:725–729.

Nyström, I. and Smedby, Ö. (2001). Skeletonization of volumetric vascular images – distance information utilized for visualization. *Journal of Combinatorial Optimization*, 5(1):27–41. Special Issue on Optimization Problems in Medical Applications.

Omholt-Jensen, T. (2002). Segmentation of the hepatic vessels as seen in mr or ct images. Master's thesis, Norwegian University of Science and Technology.

Oppenheimer, P. E. (1986). Real time design and animation of fractal plants and trees. *Computer Graphics*, 20(4):55–64.

Palagyi, K., Sorantin, E. Balogh, E., and Kuba, A. (2001). A sequential 3d thinning algorithm and its medical applications. In Insana, M. and Leahy, R., editors, *IPMI 2001, LNCS 2082*, pages 409–415. Springer-Verlag Berlin Heidelberg.

Papoulis, A. (1991). *Probability, Random Variables, and Stochastic Proceses.* McGraw-Hill, New York, 3rd edition.

Perona, P. and Malik, J. (1990). Scalar-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639.

Purves, W. K., Sadava, D., Orians, G. H., and Heller, H. C. (2001). *Life: The science of Biology.* Sinauer Associates, Inc and W.H. Freeman and Coompany.

Richter, J. P. (1970). *The notebooks of Leonardo da Vinc Vol. 1.* Dover Pubns.

Ripley, B. D. (1987). *Stochastic Simulation.* John Wiley & Sons.

Saha, P., Chauduri, B., and Majumder, D. (1997). A new shape preserving parallel thinning algorithm for 3d digital images. *Pattern Recognition*, 30:1939–1955.

Sclaroff, S. and Liu, L. (2001). Deformable shape detection and description via model-based region grouping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(5):475–489.

Sethian, J. A. (1996). *Level Set Methods and Fast Marching Methods: Evolving interfaces in Computatinal Geometry, Fluid Mechanics, Computer Vision and Materials Science.* Cambridge University Press.

Sethian, J. A. (1997). Tracking interfaces with level sets. *American Scientist.*

Sethian, J. A. (2004a). Evolving interface approach to shape recovery. `http://math.berkeley.edu/~sethian/Movies/Movieartery.html`.

Sethian, J. A. (2004b). Noise removal from images. `http://math.berkeley.edu/~sethian/Movies/Movienoiseremoval.html`.

SIGGRAPH-group (2005). Acm siggraph. `http://www.siggraph.org`.

Skjermo, J. and Eidheim, O. C. (2005). Polygon mesh generation of branching structures. *14th Scandinavian Conference on Image Analysis.*

Soille, P. (2003). *Morphological Image Analysis.* Springer-Verlag.

Soler, L., Delingette, H., Malandain, G., Montagnat, J., Ayache, N., Koehl, C., Dourthe, O., Malassagne, B., Smith, M., Mutter, D., and Marescaux, J. (2001). Fully automatic anatomical, pathological, and functional segmentation from ct scans for hepatic surgery. *Computer Aided Surgery*, 6:131–142.

Sonka, M., Hlavac, V., and Boyle, R. (1999). *Image Processing, Analysis and Machine Vision*. PWS Publishing, second edition.

Székely, G. and Gerig, G. (2000). Model-based segmentation of radiological images. *Künstliche Intelligenz*, 3:18–23.

Thomas, J. G., Peters II, R. A., and Jeanty, P. (1991). Automatic segmentation of ultrasound images using morphological operators. *IEEE Transactions on Medical Imaging*, 10(2):180–186.

Tom, B., Efstratiadis, S., and Katsaggelos, A. (1994). Motion estimation of skeletonized angiographic images using elastic registration. *IEEE Transactions on Medical Imaging*, 13(3):450–460.

Tuduki, Y., Murase, K., Izumida, M., Miki, H., Kikuchi, K., Murakami, K., and Ikezoe, J. (2000). Automated seeded region growing algorithm for extraction of cerebral blood vessels from magnetic resonance angiographic data. *Proceedings of the 22nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 3:1756–1759.

Volkau, I., Zheng, W., Baimouratov, R., Aziz, A., and Nowinski, W. L. (2005). Geometric modeling of the human normal cerebral arterial system. *IEEE Transactions on Medical Imaging*, 24(4):529–539.

Watt, A. (1999). *3D Computer Graphics*. Addison Wesley.

Winkler, G. (1995). *Image Analysis, Random Fields and Dynamic Monte Carlo Methods*. Springer-Verlag.

Xie, W., Thompson, R., and Perucchio, R. (2003). A topology-preserving parallel 3d thinning algorithm for extracting the curve skeleton. *Pattern Recognition*, 36:1529–1544.

Xu, C. and Prince, J. L. (1998). Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369.

Xu, C. and Prince, J. L. (2000). Gradient vector flow deformable models. In Bankman, I., editor, *Handbook of Medical Imaging*, pages 159–169. Academic Press.

Yim, P., Choyke, P., and Summers, R. (2000). Gray-scale skeletonization of small vessels in magnetic resonance angiography. *IEEE Transactions on Medical Imaging*, 19(6):568–576.

Zahlten, C., Jürgens, H., Evertsz, C. J. G., Leppek, R., Peitgen, H.-O., and Klose, K. J. (1995). Portal vein reconstruction based on topology. *European Journal of Radiology*, 19(2):96–100.