Dimensional analysis of blood vessel morphology using image processing techniques

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

- A master's thesis submitted in partial fulfilment of the requirements for the award of Master of Philosophy at Loughborough University.

Metadata Record: https://dspace.lboro.ac.uk/2134/31987

Publisher: © A. Eustaquio-Martin

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 2.5 Generic (CC BY-NC-ND 2.5) licence. Full details of this licence are available at: http://creativecommons.org/licenses/by-nc-nd/2.5/

Please cite the published version.
This item was submitted to Loughborough University as an MPhil thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

For the full text of this licence, please go to:
http://creativecommons.org/licenses/by-nc-nd/2.5/
DIMENSIONAL ANALYSIS OF BLOOD VESSEL MORPHOLOGY USING IMAGE PROCESSING TECHNIQUES

by

Almudena Eustaquio-Martin

A Master's Thesis
Submitted in partial fulfilment of the requirements for the award of Master of Philosophy of the Loughborough University of Technology
November 1995
A mis padres
ABSTRACT

An image processing procedure to perform a quasi-real time dimensional analysis of blood vessels undergoing morphological changes in a perfusion myograph is developed. Images of blood vessels are recorded using perfusion myography and video microscopy. The vessels' edges are detected automatically using an edge-tracking algorithm which combines a tracking method with image processing techniques, including: gradient operators and threshold. Because of the scattering effects that occur when the images are recorded, the edges found by the algorithm are not the actual vessels' edges. An attempt to correct this effect using a low order optical model based on Beer-Lambert's law of light transmission through absorbing media is proposed and tested.
I would like to express my gratitude to the numerous people that have helped me throughout this project, but special thanks are given to my supervisor, P. R. Smith, for his guidance through the course of the research, to all members of the Optical Engineering Group for supporting me at all times and to Bert Thurston and Mike Bennett for providing me with the images and the physiological background needed to carry out this research. I would also like to thank Enrique Lopez-Poveda for his encouragement and unconditional support at all stages in the project.
1. INTRODUCTION

2. BACKGROUND

1. DIGITAL IMAGES
2. GENERAL METHODS TO FIND EDGES IN A DIGITAL IMAGE
   2.1 Techniques based on edge-enhancement by linear and/or nonlinear operators and threshold Gradient operators Laplacian operators
   2.2 Techniques that involve the fitting of an ideal edge model to the image data
3. PARTICULAR METHODS USED TO DETECT EDGES IN BLOOD VESSEL IMAGES
   The need for a new algorithm for estimating blood vessel dimensions
3. THE EDGE-TRACKING ALGORITHM

1. IMAGE PROCESSING TECHNIQUES EMPLOYED IN THE ALGORITHM
   1.1 LookUp-Tabe (LUT) transformation
   1.2 Threshold
   1.3 Contrast
   1.4 Derivative

2. THE ALGORITHM
   2.1 General overview
   2.2 Starting edges
      Approximated starting edges
      Standard four-edges vessels
      Narrow vessels
   2.3 Edge-tracking method
   2.4 Diameter calculation

3. GENERALISATION OF THE ALGORITHM FOR A TIME VARYING IMAGE

4. IMPLEMENTATION OF THE ALGORITHM

4. RESULTS

5. AN OPTICAL MODEL OF LIGHT TRANSMISSION TO CORRECT THE RESULTS OBTAINED WITH THE EDGE-TRACKING ALGORITHM

1. THE MODEL
2. APPLICATION OF THE MODEL TO CORRECT THE RESULTS OBTAINED USING THE EDGE-TRACKING ALGORITHM
3. DISCUSSION

6. GENERAL DISCUSSION AND CONCLUSIONS

FURTHER RESEARCH

REFERENCES

APPENDIX A: Code Listings

APPENDIX B: Dimensional analysis of blood vessel images in real time
Chapter 1

Introduction

The work presented in this thesis was originally motivated by the need of an automatic method for measuring the dimensional changes of dissected blood vessels to vasoconstrictor and vasodilator drugs. The dimensional variations are demonstrated on rat mesenteric resistance arteries.

Until recently, information about the properties of resistance arteries had to be inferred from haemodynamic studies and from visual examination of vessels in vascular beds that were readily accessible. More recently a technique has been developed which allows, for the first time, to study the structure and function of isolated sections of resistance arteries. This technique is referred to as perfusion myography [1-2]. Segments of vessel are secured on two glass cannulae using single strands of a nylon braided suture. The artery is perfused with physiological salt solution and the perfusion pressure maintained constant. The pressurisation of cannulated resistance arteries allows the vessel to assume a conformation that resembles the in vivo condition.

The arteriograph is transferred to the stage of a light microscope with a TV camera attached to the viewing tube. Images of the vessel are formed under diffuse white-light illumination. Contrast is achieved by differential light absorption of the lumen
Chapter 1: Introduction

with respect to the vessel's walls. The changes in vascular diameters to the vasoconstrictor noradrenaline and vasodilator acetycholine are then recorded.

The purpose of this thesis is to develop an automatic method to detect in real time the dimensional variations of blood vessel images recorded using perfusion myography. An algorithm based on image processing techniques has been developed to accomplish this task. The proposed algorithm has been designed and implemented so that the dimensional analysis can be done in quasi real time. Consequently, one of the characteristics of the algorithm must be to minimise the computation time to analyse each image. The algorithm keeps this requirement by examining a very small number of pixels rather than the whole image. This is possible using a tracking method that detects the vessel edges by operating only over a small portion of the image around the vessel. This algorithm will be referred to as the edge-tracking algorithm.

Although the edges detected by the algorithm are those observed in the image, they do not correspond to the actual edges of the vessel. The physical origin of this inconsistency is likely to be attributable to a scattering effect caused by diffuse illumination of the vessel lumen through the optically diffusing tissue of the wall. A low-order optical model of light transmission through a blood vessel with circular cross-section is derived and used to establish a relationship between the edges found by the edge-tracking algorithm and the actual edges of the vessel.

Some of the contents in this thesis were presented in the form of paper at the European Biomedical Optics conference celebrated in Barcelona in September, 1995 (see Appendix B).
Chapter 1: Introduction

A brief guide to the thesis

This thesis is divided into six chapters. The order of the chapters follow a logical process of presentation leading to a final discussion and conclusions.

Chapter 2 is a review of the general methods most commonly employed in image processing to detect objects' edges in a digital image. Additionally, an updated review of the particular methods developed to detect blood vessel edges by image processing techniques is presented. The new system for the detection of vessel edges developed in this thesis is placed in the context of the alternative method previously reviewed.

In Chapter 3 the edge-tracking algorithm is explained in detail. In the first section, the image processing techniques employed by the algorithm are presented. Section 2 presents the tracking method to detect vessel edges. The way in which the algorithm is applied to analyse a continuous sequence of images is presented in Section 3. Finally, some general considerations on the implementation of the algorithm are given in Section 5 of this chapter.

In Chapter 4 the results obtained with the edge-tracking algorithm are presented. An explanation of some of the problems encountered by the algorithm is also given.

In Chapter 5 the need to establish a relationship between the position of edges found by the edge-tracking algorithm and the actual dimensions of the vessels is explained. A low-order optical model of light transmission through vessels with circular cross-section is presented as a plausible approach to obtain the required relationship.

A general discussion on the validity, applicability and possible improvements of the presented method to perform the dimensional analysis of blood vessels is given in Chapter 6.
Chapter 2
Background

1 DIGITAL IMAGES

A good definition of digital image is given by Niblack [3]:

"...in digital image processing (...) an image is a two dimensional array of numbers: each number corresponds to one small area of the visual image, and the number gives the level of darkness or lightness of the area. We will assume that the higher the number, the lighter the area, so zero is black, the maximum value is white, and intermediate values are shades of grey. Each small area to which a number is assigned is called a 'pixel'..."

([3], pp. 16-17)

It will always be assumed that an image is a rectangular array of \( N \) columns by \( M \) rows. Each pixel is, therefore, characterised by its value (level of grey), and a set of co-ordinates \((n, m)\) which defines the position of the pixel within the two-dimensional array. In our case, the value of each pixel will always be an integer number between 0 (black) and 255 (white). In computer terms the value of each pixel is stored as a byte (8 bits).

Equivalently, a digital image can be considered as a mathematical function of two variables, \( I(n, m) \), where \( n \) \((1 \leq n \leq N)\) indicates the column and \( m \) \((1 \leq m \leq M)\) the row in the rectangular array; The value of the function \( I \) \((0 \leq I \leq 255)\) specifies the grey-level,
or intensity, of the pixel. Conventional mathematical operations, expressed in their
digital form, (e.g. gradient and Laplacian) can be performed on the image.

A way to know some of the images’ features is to find the edges in the image. Many
edge-detector methods have been developed with this purpose. An edge in an image
can be defined as a local discontinuity in the grey-level function of the image, i.e., a
local ‘jump’ between two well-defined intensity levels. Usually, these changes in
intensity correspond to the objects’ boundaries in the image. The detection and
localisation of intensity changes is problematic due to the presence of noise ([4], p.
193). Noise is a variation in the image intensity which carries no useful information
but may be confused with scene-related intensity changes. Both noise and edge
features tend to be high spatial frequency events, i.e., the intensity changes rapidly
over the image. It is inevitable that the edge-detection process will be affected by the
signal-to-noise ratio.

The general methods for the edge-detection in digital images are reviewed in Section
2 of this chapter. In Section 3, a review of several studies in which these edge-
detection techniques have been applied to the detection of blood vessel edges is
presented.

2 GENERAL METHODS TO FIND EDGES IN A DIGITAL IMAGE

There are two main types of techniques to find object’s edges in a digital image:

2.1- Techniques of edge-detection by applying linear or nonlinear operators to the
image.
2.2-Techniques that involve the fitting of an ideal edge model to the image data.

Each type is explained separately in the following sections.
2.1 Techniques based on edge-enhancement by linear and/or nonlinear operators and threshold

A common approach to edge-detection is to apply an edge-enhancement operation in order to produce an image with accentuated spatial changes in intensity. The enhancement of the edges is usually obtained by convolution of the image with either a linear or a nonlinear mask. There are several types of masks, but the most common ones are those that use the first and second order derivatives, i.e., the gradient and the Laplacian operators respectively. Since the derivative behaves as a high pass filter, it is suitable to enhance high spatial frequency events such as object's edges.

The gradient and the Laplacian operators enhance the edges in a different way (see below). The gradient produces an increase in intensity around the exact position of the edges. Using the Laplacian operator, however, the edges are exactly localised at the zero-crossing points of the Laplacian of the image. When using the gradient, therefore, an extra threshold needs to be applied in order to detect the approximate position of the edges.

The gradient operators

An edge in a image corresponds to large values of the gradient operator applied to the function $I(n, m)$. Moving across an edge the gradient starts at zero, increases to a maximum at the position of the edge, and decreases back to zero (see Figures 2.1(a) and 2.1(b)).
Figure 2.1. (a) Step function that represents an ideal edge. (b) The gradient of the step function. (c) The Laplacian of the step function.
For a continuous function, \( f(x, y) \), the mathematical expression for the gradient in two dimensions is:

\[
\nabla f(x, y) = \frac{\partial f(x, y)}{\partial x} \hat{i} + \frac{\partial f(x, y)}{\partial y} \hat{j}
\]

(1)

However, an image is defined by a digital function \( I(n, m) \) and, therefore, a digital gradient must be computed. A digital gradient may be computed by convolving two windows with the image ([3], pp. 73-74): one window gives the \( x \) component of the gradient, and the other gives the \( y \) component. The \( x \) component of the gradient, \( g_x \), is evaluated as the difference between two consecutive pixels in the horizontal axis. It is mathematically expressed as:

\[
g_x(n, m) = I(n+1, m) - I(n, m)
\]

(2)

The \( y \) component of the gradient, \( g_y \), is evaluated as the difference between two consecutive pixels in the vertical axis. Its mathematical expression is:

\[
g_y(n, m) = I(n, m+1) - I(n, m)
\]

(3)

As a result, the \( x \) component of the gradient, \( g_x(n, m) \), is very sensitive to large sudden changes in the image intensity along the \( x \) direction. Therefore, it detects the vertical edges in the image. On the other hand, the \( y \) component of the gradient, \( g_y(n, m) \), is more sensitive to changes along the \( y \) direction and; hence, it detects the horizontal edges in the image.

Equations (2) and (3) generate values of the gradient centred at the point \((n+1/2, m)\) for the gradient in \( x \)-direction and at the point \((n, m+1/2)\) for the gradient in \( y \)-direction. To obtain values centred at \((n, m)\), a symmetric gradient about \((n, m)\) must be used. It may be mathematically expressed as follows:
\[ g_x(n, m) = I(n+1, m) - I(n-1, m) \]  \hspace{1cm} (4)

\[ g_y(n, m) = I(n, m+1) - I(n, m-1) \]  \hspace{1cm} (5)

To apply the gradient defined in the equations (4) and (5) to an image is equivalent to convolving the image with two masks, namely mask\(_x\) and mask\(_y\):

\[ mask_x = \begin{pmatrix} -1 & 0 & 1 \end{pmatrix} \]  \hspace{1cm} (6)

\[ mask_y = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \]  \hspace{1cm} (7)

These masks only find the edges along the x- and y- directions. To enhance the edges along the diagonal directions there is another set of masks. They are known as the Robert's operators ([3], p. 75) and are represented by the masks:

\[
\begin{pmatrix}
-1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 1
\end{pmatrix}
\]

To enhance edges at 45 degrees

\[
\begin{pmatrix}
0 & 0 & -1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{pmatrix}
\]

To enhance edges at 135 degrees

The immediate drawback of these gradient filters is their sensitivity to noise. An alternative to the above gradient operators to reduce the noise is to use nonlinear derivatives. For example, Rosenfeld [5] defines a derivative which consists in
Chapter 2: Background

smoothing the image before estimating its derivative. The gradient is computed taking the absolute differences of running averages of function values, i.e.:

\[
D = \frac{|I(n+k)+...+I(n+1)|}{k} - \frac{|I(n)+...+I(n-k+1)|}{k}
\]  

(8)

Using this derivative, the larger the \(k\), the more conspicuous becomes the central edge relative to the noise edges. However, the larger the \(k\), the less precisely localised is the detected edge.

Another way of avoiding the noise is by convolving the image with larger masks, such as Prewitt and Sobel's mask ([3], p. 74; [6] p. 487). To apply this mask to the images is equivalent to averaging (which reduces the noise) and performing a derivative, which enhances the edges.

After the gradient is applied (i.e., after emphasising the edges), the pixels of the image which correspond to the actual edges need to be identified. A common approach is to select the pixels of sufficiently large gradient magnitude, i.e., the point where the luminance transition is sharp enough, by applying a suitable threshold. In the thresholded image the edges will be displayed as white features on a black background.

A qualitative evaluation of the design and performance of these enhancement/thresholding methods for edge detection is reported by Abdou and Pratt in [7].
Laplacian operators

The Laplacian operator is equivalent to the second order derivative. The edges in an image correspond to the zero crossing of the Laplacian operator applied to the function \( I(n, m) \) (see Figure 2.1(c)).

For a continuous function, \( f(x, y) \), the mathematical expression of the Laplacian in two dimensions is as follows:

\[
\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}
\]

(9)

In order to apply the Laplacian operator to a digital image, a discrete Laplacian must be defined. In one dimension the digital Laplacian is defined as:

\[
l_x(n) = g_x(n) - g_x(n-1) = I(n) - 2 \times I(n-1) + I(n-2)
\]

(10)

There are various ways in which this expression can be extended to two dimensions (see [6], p. 482).

Both the gradient and the Laplacian operators are significantly different. The gradient tends to produce large values over a broad region, which does not help to accurately localise the edges in the image. The zero crossing of the Laplacian, on the contrary, produces exact localisation of the edges but amplifies any error component. It is, therefore, much noisier than the gradient (see Figure 2.1). Consequently, the gradient is more appropriate in noisy images, although the edges will not be so accurately localised. The Laplacian should be used for clear images.
2.2 Techniques that involve the fitting of an ideal edge model to the image data

In these techniques the actual image data is locally matched to some ideal representation of a one- or two-dimensional edge model. If the fit is sufficiently accurate, and edge is assumed to exist and its assigned parameters are those of the ideal edge model.

![Graph showing one-dimensional edge fitting](image)

**Figure 2.2.** One-dimensional edge fitting.

A simple example in one dimension is to fit the image signal, \( I(n) \), with the ideal step function \( s(n) \) (see Figure 2.2):

\[
s(n) = \begin{cases} 
  b & \text{if } n < n_0 \\
  b + h & \text{if } n \geq n_0 
\end{cases}
\]

An edge is present if the mean square error:
\[ \varepsilon = \int_{n_0}^{n_0+L} [I(n) - s(n)]^2 \, dn \]  \hspace{1cm} (12) 

is below some threshold value.

Although the example given is for a one-dimensional edge, the same principle can be trivially extended for edges in two dimensions (see [6], p. 492).

3 PARTICULAR METHODS USED TO DETECT EDGES IN BLOOD VESSEL IMAGES

All the techniques explained above are useful to find object's edges in any kind of digital images. In the case of blood vessel images, due to the particular characteristics of this type of images, specific techniques have been developed to detect the edges of the vessels. Most of the techniques developed use fitting methods (explained above), or some variety of tracking method. The tracking methods consist in finding the edges of the vessel in a column of the image using the edges previously found in a different column.

Shmueli et al [8], develop a method to determine blood vessel boundaries from X-ray images with low signal-to-noise ratio. The method finds the vessel contours scanning the vessel image line by line across the vessel, assuming that due to the vessel properties (namely, spatial continuity and that the edges' position can not change abruptly from line to line) the edges' values at some scanned line must be somehow related to their values at neighbouring scanned lines. The computation time is approximately 15 minutes for an entire image.

Reiber et al [9], [10], develop a technique to assess coronary arterial dimensions automatically. The method consists in selecting and magnifying a region of the
image. Then, the vessel's edges in this region are found by computing the weighted sum of the first and second derivative functions over perpendicular scanlines to the vessel centreline. This method requires the human specification of a region of interest and a tentative centreline and it is very expensive in computation time (approximately 10 minutes per blood vessel).

Many of the methods used to find blood vessel edges are based on fitting techniques. These fitting algorithms make use of perpendicular vessel profile intensity models, and find the edges by minimising the mismatch between the measured intensity profile and the model. Pappas [11], uses a fitting technique to find the edges of coronary angiograms (X-ray picture of arteries taken after a contrast agent has been injected into the vessels). Additionally, he compares the results of this method with the results obtained with methods based on derivative techniques, concluding that the results of the fitting technique are more accurate but require more computation time.

The defined vessel-profile model is based on the study of a three-dimensional projection of a vessel model. In addition, a model for the background is defined as a low order polynomial. Once a model is defined, the user interaction is needed to begin the process; the user must specify a starting point and the approximate direction of the vessel. Then, the analysis is done by examining the densitometric profiles along lines crossing the vessel, assuming each profile is independent of the others.

Sun [12], combines a fitting method with a tracking algorithm. The proposed method finds the edges (and, therefore, the diameter) of digital coronary arteriograms beginning from a start-search-point given by the user, which makes the method not to be totally automatic. This method finds the vessels' boundaries in a point along the length of the vessel by analysing an interval of pixels around the vessel perpendicular to its direction. To determine this interval of pixels a point inside the vessel and the vessel direction must be known. The internal point is calculated by tracking the centreline along the vessel, and for each new column it is actualised by fitting the
new intensity profile with an ideal rectangular model of the vessel's profile. The new
direction of the vessel will be the direction of the line between two consecutive
centre points. A threshold parameter is used to automatically determine the end of an
artery. The computation time is about 1.87 seconds per vessel.

Miles and Nuttall [13] also combine fitting and tracking methods to estimate the
diameter of cochlear blood vessels in digital images. The method develops a model
of the vessel cross-section based on the vessel geometry and the physics of the
imaging process. The algorithm fits the model profile with the real profile of the
vessel. To determine the point where the calculations must be performed a tracking
method is used. The tracking method tracks the vessel pathway beginning in a
starting point given by the user. The user must also specify a stop point that
determines the end of the vessel. The computation time is approximately 15 seconds
for each image.

Chaudhuri et al [14], use a two-dimensional matched filter for the detection of blood
vessel edges in retinal images. By studying the optical and spatial properties of an
image, the grey-level profile of the cross-section of a blood vessel is approximated by
a Gaussian-shaped curve plus a Gaussian white noise. In order to improve the results
the matching operation is performed over a number of cross-sections, thus generating
a two-dimensional filter. To localise an edge in the image a threshold is applied: if
the grey-level of a pixel is above the threshold value the pixel belongs to the vessel.

Michoud et al [15], determine the vessel diameter by fitting the data obtained from a
densitometric segment perpendicular to the vessel with a model of the light
attenuation caused by the blood vessel, which is calculated using Lambert-Beer's law
of light absorption [16]. Two models are proposed. The first model assumes that red
blood cells are uniformly distributed in the vessel, whereas the second one takes into
account the presence of a plasma layer at the vessel's wall. The data is fitted using
nonlinear algorithms. The computation time for the first method is about two
seconds for every measurement, while the second model needs one minute per measurement.

To compare all these methods is a difficult task because each one of them has been developed for a different kind of blood vessel and a different type of image. Methods [9] to [12] are developed for the study of coronary arteries with a contrast agent inside. Method [13] is developed for cochlear blood vessels with fluorescent dyes inside. Finally, method [14] is suitable for retinal images. Additionally, all the methods mentioned above ([8] to [14]) are to be applied on images where a vessel tree is displayed or the vessel is not the only object in the image (for instance, for images taken \textit{in vivo} where bones and various types of tissues may also appear in the image). In the later case some contrast agent is necessary to help distinguish the vessel boundaries from the background objects. A threshold is generally applied to these images in order to define the vessel boundaries before further analysis.

\textbf{The need for a new algorithm for estimating blood vessel dimensions}

It has been always of great interest to the medical community to investigate the blood vessels' dimensions and their temporal and spatial variations to vasodilator and vasoconstrictor drugs. However, the knowledge in this matter has been always limited by the existing technology and software to measure such dimensions. Generally, common image processing techniques have been applied to blood vessel images in order to enhance and detect the vessels' boundaries (see above). None of the above reviewed algorithms for edge-detection in blood vessel images is suitable when the purpose is to measure the vessels' dimensions automatically. Most importantly, none of those algorithms was designed to measure both the spatial and temporal variations of the vessels' dimensions in quasi-real time.
The aim of this project is to develop an algorithm that automatically measures the internal and external diameters of a single blood vessel in a digital image and its spatial and temporal variations in real time. In order to find the internal and external diameters and their spatial variations, the algorithm must be able to detect the internal and external vessel's edges not only at a single cross-section of the vessel, but along the whole of its length. Additionally, the aimed algorithm must be fast enough to permit the analysis of the diameter variations in real time. Therefore, each sampled image should be analysed in the shortest possible time. An extra specification is that the algorithm must avoid the problems caused by the presence of tissues, bubbles, and any other impurities in the image. Finally, the edge-detection process must be fully automatic, i.e., without any interaction with the user.

The way in which the algorithm is designed depends very much on the way in which the blood vessel image is presented and recorded. The aimed algorithm is to be applied to blood vessel images recorded using perfusion myography. Therefore, a segment of blood vessel (placed on a light background) is, ideally, the only object in the type of images with which the algorithm will be dealing. In the real images, however, some impurities in the background or inside the vessel will be also present. These impurities are, in most cases, traces of tissue attached to the vessels' walls, air bubbles in the lumen and noise in the image. The algorithm must be able to deal with these types of impurities.

The profile-fitting methods reviewed above, [11] to [15], do not study the possibility that the vessels have bubbles inside. In the real images, the blood vessels' profiles are modified by impurities in the lumen, making impossible the fitting with the model profile. Since the bubbles are placed at different positions and have different dimensions for different vessels, it is impossible to define a profile model which includes possible air bubbles in the lumen. Moreover, profile-fitting techniques can not be generalised for narrow and wide vessels because narrow blood vessels have profiles which are very different from those of a normal vessel. In normal blood
vessels there is a good contrast between the intensity level of the vessels' wall and the lumen. This is not the case for narrow vessels. Therefore, it would be necessary to define different profile models for different internal diameters. Finally, the profile-fitting methods may be more accurate but they are computationally more expensive which makes them inadequate when a quasi-real time analysis is required.

The general methods (see above) to find the edges in a digital image are appropriate to find all the edges in the image. Since the ultimate objective, in our case, is to calculate the vessels' dimensions, i.e., their internal and external diameters, only the edges of the blood vessel need to be found. If the algorithm was such that all the edges in the image were detected, the impurities' edges could be confused with those of the vessel, leading to erroneous values of the vessels' dimensions.

It is possible indeed to remove all the impurities around the vessels' edges using general image processing methods. This could lead us to think that the general methods of edge-detection could be used for our purpose. However, to remove all the impurities costs a great deal of computation time. Besides, since the impurities are placed at random and have different characteristics for different images, a particular method should be designed to eliminate them in every image. This, again, requires long computation times and prevents the process of edge-detection from being totally automatic.

The last resource to solve the proposed problem is to design the algorithm as an edge-tracking one (i.e., to use a similar approach as those reported in [8], [12] and [13]). The algorithm could be designed to avoid all impurities as long as they do not touch the vessel's walls. Additionally, this approach requires only a small number of pixels around the vessels' wall to be analysed. Hence, it reduces the computation time enormously, making quasi-real time analysis possible. However, none of the edge-tracking methods reviewed above are appropriate in our case for two reasons. Firstly, they are designed to detect the external edge only. We, however, are
interested in an algorithm which detects both the internal and external edges. Secondly, they need the user's interaction to indicate the starting point for the analysis, the vessel's direction or to select a segment of the vessel. In other words, the analysis process is not fully automatic, which makes impossible their application to detect the temporal variations of blood vessels' dimensions.

All methods proposed in the literature and reviewed above have aspects which are useful in our case. However, it has been shown that none of them is fully appropriate for our purpose. This justifies the effort dedicated to develop the new edge-tracking algorithm presented in this report and the model for correcting the approximated vessel's dimensions obtained with it.
Chapter 3

The edge-tracking algorithm

In this chapter an algorithm to detect the edges of a blood vessel displayed in an image is proposed and explained in detail. The first section describes the image processing techniques employed by the algorithm and how these techniques can be applied to the task of detecting blood vessel edges. In Section 2 the algorithm is developed. Firstly, an overview of the way in which the algorithm operates is presented. Later in this section, a detailed description of every step followed by the algorithm to detect and track the vessel's edges is given. In Section 3 it is explained how the proposed algorithm can be generalised to detect the vessel edges in a sequence of images varying in time. Finally, Section 4 describes how the algorithm has been implemented and executed in practice.

1 IMAGE PROCESSING TECHNIQUES EMPLOYED IN THE ALGORITHM

1.1 LookUp-Table (LUT) transformation

An LUT transformation is an operation performed on the grey-level values of an image to enhance the image without adding new information [17, p. 25].
Chapter 3: The edge-tracking algorithm

The grey-level interval, \((\text{min}, \text{max})\), in which all the pixels of an image are contained, is usually smaller than the maximum interval allowed, \((0,255)\). The LUT transformation used by the edge-tracking algorithm is a linear transformation that, when applied to an image, modifies the grey-level of all pixels in the image extending it from the interval \((\text{min}, \text{max})\) to the interval \((0, 255)\). It is done by applying the following operation to each pixel:

\[
O(n,m) = \frac{255 \times (I(n,m) - \text{min})}{\text{max} - \text{min}}
\]  

(1)

where the pair \((n, m)\) indicates the co-ordinates of a generic pixel in the image; \(O(n,m)\) is the new intensity value that the LUT transformation assigns to the pixel; and \(I(n,m)\) is the grey-level of the pixel before the transformation is applied.

This transformation enhances the contrast and the brightness of the image. All the images to which this transformation is applied, will end up with the same grey-level interval, \((0, 255)\).

This transformation is computationally expensive when it is applied to the whole image. It will be explained below that the proposed algorithm requires to examine only a few columns in the image and, therefore, the LUT transformation needs to be applied only to those columns. This reduces significantly the time of computation. For a single column the LUT transformation consists in calculating the maximum and the minimum grey-level of the pixels in the column, and applying expression (1) to all of them. Figure 3.1 shows a typical intensity profile of a blood vessel image before and after the LUT transformation is applied. It is seen that the grey-level interval of the column with the LUT transformation is larger, \((0,255)\), than that of the column without the LUT transformation.
1.2 Threshold

Sometimes it is useful to separate the pixels of an image into different categories attending to some common characteristics. This process is called segmentation.

The threshold technique is a segmentation tool commonly used in image processing analysis. The threshold divides the image in two categories, 'zero' and 'one', transforming an ordinary image in a binary image, making possible the separation between the background and the objects. This is achieved by assigning the value 'zero' to those pixels whose grey-level is above a predetermined threshold value (these pixels usually correspond to the background), and the value 'one' to those pixels whose grey-level is below the threshold (these pixels usually correspond to the objects in the image). The assignment may be done the other way round.
For instance, when the object in the image is a blood vessel, after the threshold is applied the background and the lumen of the vessel take value 'zero', and the vessel's walls take value 'one' (see Figure 3.2 (b)). Therefore, the walls of the vessel, which are the interesting part of the image, become easily distinguishable from the background. Figure 3.2 (a) shows a blood vessel image, and Figure 3.2 (b) shows the same image after a threshold has been applied.

![Figure 3.2](image)

(a) (b)

Figure 3.2 (a) A blood vessel image. (b) The same image as in (a) after a threshold has been applied (threshold value=125).

Because the transition between the background and the vessel's walls is not a sudden one, the selection of the threshold is a difficult task. If the threshold is set too low some pixels of the vessel's walls may be considered as part of the background; on the other hand, if it is set too high some pixels of the background may be considered as part of the vessel's edges. The appropriate threshold for a given image can be found by manual selection; that is to apply different thresholds to the image selecting the
best one according to the user's opinion. Alternatively, an automatic method may be used to obtain an optimum threshold ([3], pp. 113-117; [4], p. 70; [6], pp. 534-539).

The threshold is usually applied to whole images. In the proposed algorithm, however, the threshold needs to be applied only to some of the interesting columns (see below). As a result the computation time is considerably reduced, yet obtaining the desired effect in the interesting part of the image. The appropriate threshold is chosen by manual selection (see below).

1.3 Contrast

Sometimes the image needs to be enhanced in order to facilitate the analysis. A contrast enhances the image seeking to improve its appearance to a human viewer or to make easier the machine processing operations.

There are many ways to contrast an image as, for example, by redistributing its grey-levels ([18], p. 136) or by using Fourier transformations ([18], p. 82). The contrast applied in our particular algorithm is defined by the MCONTRAST function of Visilog ([19], p. II-2-49).

The MCONTRAST function is based on the erosion and dilation of an image. Erosion is a transformation over the image that removes isolated points and the small particles, shrinks the other particles, discards peaks on the boundary of the objects, and disconnects some particles ([17], pp. 52, 53). Dilation fills small holes inside particles and gulfs on the boundary of objects, enlarges the size of the particles and may connect neighbouring particles ([17], pp. 54, 55).
Chapter 3: The edge-tracking algorithm

The function MCONTRAST is based on the fact that the function that defines the image $I(n, m)$ is always lying between the erosion and the dilation of $I(n, m)$. Mathematically this can be expressed as:

$$E(I(n,m)) \leq I(n,m) \leq D(I(n,m))$$

(2)

where: $I(n, m)$ is a function that defines the intensity of the pixels inside the image; $E(I(n,m))$ is the erosion applied over $I(n, m)$ ([19], p. II-4-5); and $D(I(n,m))$ is the dilation applied over $I(n, m)$ ([19], p. II-4-9).

For each pixel, $(n, m)$, the output grey-level value, $O(n, m)$, will be the closest value of $I(n, m)$ between the erosion and the dilation, i.e.:

$$O(n,m) = 
\begin{cases} 
D(I(n,m)) & \text{if } [D(I(n,m)) - I(n,m)] < [E(I(n,m)) - I(n,m)] \\
E(I(n,m)) & \text{otherwise} 
\end{cases}$$

(3)

While in Visilog the function MCONTRAST is applied to the whole image, in this algorithm it is applied only to a few pixels (see below) in the image. Again this reduces tremendously the time of computation and yet the results obtained are as expected.

1.4 Derivative

The derivative is a common technique used to enhance the edges present in an image. This is because the derivative acts like a high frequency filter and the edges are high frequency evens in the image.
Chapter 3: The edge-tracking algorithm

The derivative used by this algorithm consists in carrying out two operations on the image:

1.- Average in the x-direction
2.- Derivative in the y-direction

1.- Average in the x-direction.
By averaging in the x-direction the noise in the image is reduced and, therefore, the edges are more easily found. If $I(n, m)$ is the function that defines the image, the average of two pixels in x-direction is expressed as:

$$A(n,m) = \frac{I(n-1,m) + I(n,m)}{2}$$

The greater the number of pixels over which the average is calculated, the larger the noise reduction. On the other hand, if the average is calculated over a very large number of pixels, the image will be have low resolution. As a compromise solution, in this algorithm the average is done over three pixels, i.e.:

$$A(n,m) = \frac{I(n-1,m) + I(n,m) + I(n+1,m)}{3}$$

2.- Derivative in the y-direction.
This algorithm has been developed for images in which the blood vessels are horizontally (or approximately horizontally) displayed. Therefore, the detected edges, will also be horizontally placed in the image. In this case, for the derivative to enhance the edges it must be performed cross-sectional to the vessel's orientation, i.e., in the vertical direction or y-direction (see Chapter 2, Section 2.1). For a continuous function the derivative in y-direction is defined as $\frac{dI(n,m)}{dy}$. Its digital approximation may be computed as the difference between the grey-levels of two
pixels placed in vertical direction, ([3], pp. 73-74); Mathematically this is expressed as:

\[
\frac{dl(n,m)}{dy} = I(n,m) - I(n,m-1)
\]  

(6)

This derivative generates output values centred at the point \((n, m+1/2)\). For the derivative to give values centred at \((n, m)\) it may be defined as:

\[
\frac{dl(n,m)}{dy} = I(n,m+1) - I(n,m-1)
\]  

(7)

By combining equations (5) and (7) the general expression of the derivative implemented in this algorithm is obtained. It is mathematically expressed as:

\[
\frac{dA(n,m)}{dy} = \frac{[I(n+1,m+1) + I(n,m+1) + I(n-1,m+1)] - [I(n+1,m-1) + I(n,m-1) + I(n-1,m-1)]}{3}
\]  

(8)

### 2 THE ALGORITHM

#### 2.1 General Overview

The proposed algorithm measures the inner and outer diameters of blood vessels. It can be applied to a single frame in order to measure the diameter of a single blood vessel; and also it can be applied to a successive series of frames in order to investigate the variations of the vessel’s diameter in time.
The algorithm uses general image processing techniques including as LUT transformation threshold, derivative and contrast. The main difference between this algorithm and others used for edge detection is that other algorithms are interested in finding the edges of all the objects in the image, while this algorithm is only interested in the edges corresponding to the blood vessels' walls. Other edges, such as those of bubbles inside the vessels or impurities in the image, are not interesting because they impede to find properly the blood vessel’s diameters.

By observing a blood vessel image, it can be seen that most of the image is background; only a small part of the image is occupied by the vessel. Yet another smaller portion of the image is occupied by impurities such as bubbles inside the vessel, tissues around the vessel and noise. As we are only interested in describing the blood vessel's profile, it is not necessary to analyse the whole image. Indeed, the edges can be found by studying only a small number of pixels around the blood vessel. This reduces enormously the number of pixels which need to be analysed and, therefore, the computation time is reduced. Moreover, if only the pixels around the vessel are analysed, the impurities in the image are more easily avoided. This prevents the impurities' edges to be regarded as part of the vessel’s edges.

The proposed algorithm is based upon an edge-tracking technique. More specifically, for an ideal blood vessel image with four edges (two internal and two external), oriented in the $x$-direction (see Figure 3.3), it tracks the inner and outer edges from an initial starting column ($x=x_{ini}$), whose profile shows clearly only the four expected edges.
Figure 3.3 An illustration on how the upper external edge is tracked from the real edge $y_{\text{init}}$ in the edge-tracking algorithm. The initial column $x_{\text{init}}$ is chosen as the first column from the left at which only four changes in intensity occur.
The algorithm, therefore, has two well defined parts. Firstly, the starting column $x_{\text{init}}$, where only four remarkable changes in the intensity occur, must be found. The coordinates of the pixels at which the intensity changes occur, are taken as the coordinates of the approximate starting edges at that column ($y'_1\text{init}, y'_2\text{init}, y'_3\text{init}, y'_4\text{init}$). Secondly, the edges of the vessel are tracked from the starting edges along the length of the vessel. The tracking process is done simply by incrementing the $x$-co-ordinate of the starting edges by the quantity $\Delta x$ and taking the resulting values as the coordinates of the approximate edges in the new column ($x_{\text{init}}+\Delta x$). The real edges are found by analysing the derivative around the approximated edges. This process is repeated along the whole length of the vessel (see Figure 3.3).

Figure 3.4 shows a general overview of the way in which the algorithm operates.

![Flow diagram](image)

**Figure 3.4** A flow diagram which illustrates the main general stages in the edge-tracking algorithm.
2.2 Starting Edges

Approximated starting edges

As explained above, the first step in the algorithm is to find the starting edges. These starting edges are found using a threshold technique. Two different cases are considered at this stage depending on the type of vessel being considered: a) standard vessel, and b) narrow vessels.

Standard blood vessels (see Figure 3.5 (a)) are those in which the lumen of the vessel is perfectly distinguishable from the vessel's edges: the grey-level of the pixels of the lumen is similar to the grey-level of the background of the image. This feature is characteristic of blood vessels which are little contracted or not contracted at all. In this case the starting edges are easily found by the usual threshold technique.

Narrow blood vessel (see Figure 3.5 (b)) are those with a small lumen diameter because they are very contracted. The lumen is not distinguishable or hardly distinguishable from the vessel's edges because its grey-level is near the grey-level of the edges. In this case, the internal edges are difficult to find by the usual threshold technique.

Standard four-edges vessels

In a thresholded column the number of changes in the intensity level along the column, from 'zero' to 'one' and vice versa, is easily counted. These changes correspond to the boundaries of the objects in the image.

In an ideal image, the vessel is horizontally positioned in the image and goes from the left to the right side of the image without any cut or discontinuity. In this ideal...
image, the intensity profile along any column, \( x=x_0 \), (i.e., along a perpendicular line to the vessel orientation) must show only four well-marked discontinuities after a suitable threshold is applied to the column. Each one of the four changes in intensity corresponds to each one of the vessels' edges; namely, the external upper edge \( (y'_1) \), the internal upper edge \( (y'_2) \), the internal lower edge \( (y'_3) \) and the external lower edge \( (y'_4) \). The co-ordinates of those discontinuities \( (x_0, y'_1) \), \( (x_0, y'_2) \), \( (x_0, y'_3) \) and \( (x_0, y'_4) \) constitute the co-ordinates of the approximate starting edges of the vessel.

![Figure 3.5](image)

(a) Standard blood vessel. (b) A narrow blood vessel. While the lumen of the standard vessel is perfectly distinguishable from the vessel's wall, it is hardly distinguishable in the narrow vessel.

If the number of changes found is different than four, it means that in the column \( x=x_0 \) there is something else than background and vessel edges (e.g., impurities, such as bubbles inside the blood vessel, nodes, etc.). Therefore, it is difficult to specify which of these changes correspond to the edges of the blood vessel and which to the
impurities' edges. Indeed, the edges of the vessel can not be distinguished from the impurities' edges at all.

Narrow vessels

If the vessel is very narrow it is difficult to distinguish the lumen from the edges. Therefore, by applying a threshold only the external edges can be found. Moreover, the minimum number of changes that can be found are two, corresponding to the upper (\(y'_1\)) and the lower (\(y'_4\)) external edges. To find the internal edges a contrast must be applied (see above). A contrast enhances the image improving the appearance of the pixels between the external and internal edges allowing to find the internal edges of the vessel. Since the contrast is useful only to find the internal edges it is applied only to the pixels between the external edges found previously with the threshold technique.

In order to determine the co-ordinate of the internal edges, the maximum and the minimum intensity levels of the pixels to which the contrast was applied are found, and the average between them is calculated. The first pixel whose intensity is greater than the average is taken as the approximate upper internal edge, and the last pixel whose intensity is greater than the average is taken as the lower internal edge. This is equivalent to applying a threshold to those pixels of the column to which the contrast was applied. The threshold value would be the average intensity calculated as stated above. So the co-ordinates of the four approximate edges are found.

In this algorithm a threshold is applied column by column, from the first column, until a column with only four or two changes in the intensity is found. If a column with four changes is found, the vessel is then regarded as a standard four-edge vessel. The approximate starting edges are found with the technique explained above for
standard four-edge vessels. On the other hand, if only two changes in intensity are found, the vessel is treated as a narrow one, and the approximate starting edges are located using the narrow vessel technique.

It is important to notice that these edges are *approximate* because they depend strongly on the threshold chosen. The main problem to obtain the starting edges is to find an adequate threshold for all images. There are images darker than others and, therefore, the grey-level interval varies from one image to another depending on the illumination conditions under which the images are taken. Consequently, what is a good threshold for an image may not distinguish the background from the objects in another image. A logical solution to this problem is to apply a different threshold to every image. The most suitable threshold for each image can be determined using an automatic threshold-searching technique. An alternative solution is to adapt the image to the threshold. This requires to operate some transformation on the image making possible the use of the same threshold for all the images. In our case the latter solution has been adopted. The general transformation which makes possible that the same threshold can be applied to all the images is the LUT transformation (see Section 1.1). All the columns over which this transformation is applied result with the same grey interval between 0 and 255. Therefore, the same threshold can be applied to all images. Since this algorithm applies the threshold only to columns, the LUT transformation is equally applied individually to each column considered and not to the whole image, which would take a longer computation time.

After the LUT transformation is performed on the column, the threshold is applied. The threshold must be small enough to allow to distinguish the edges from the background. Because the technique only aims to obtain the co-ordinate of the approximate edges, the exact value of the threshold is not really critical. Here, the threshold was set to 85 which is exactly a third of the maximum grey-level, 255.
Real starting edges

Once the approximate edges are located, the co-ordinates of the real edges are found using the derivative technique. It is plausible to assume that the real starting edges are close to the approximate ones. Consequently, to find each real starting edges, $y_i$ (where $i=1$ indicates the upper external edge; $i=2$ the upper internal; $i=3$ the lower internal; and $i=4$ the lower external edge), it is enough to study the value of the derivative of the grey-level function within an interval $(y_i' - \Delta y, y_i' + \Delta y)$ around the corresponding approximate starting edge $y_i'$.

In the interval around each approximate starting edge, the position of the real starting edge coincides with the co-ordinates at which the derivative of the grey-level function takes its extreme values (maxima and minima) in that interval. The upper-external ($y_1$) and the lower-internal ($y_3$) real edges correspond to the co-ordinates at which the derivative function takes its minimum value within their corresponding intervals, $(y_1' - \Delta y, y_1' + \Delta y)$ and $(y_3' - \Delta y, y_3' + \Delta y)$ respectively. This is so, because these two edges define a change in the intensity from the background to the vessel's walls. Therefore, the derivative is calculated as the difference between pixels with a low grey-level value (i.e., these pixels correspond to the vessel's walls) minus pixels with larger grey-level values (which correspond to the background), resulting in negative values of the derivative.

On the other hand, the upper internal ($y_2$) and the lower internal ($y_4$) real edges correspond to the co-ordinates at which the derivative function takes its maximum value within their corresponding intervals, $(y_2' - \Delta y, y_2' + \Delta y)$ and $(y_4' - \Delta y, y_4' + \Delta y)$ respectively. In this case the derivative is calculated as a difference between pixels corresponding to the background (high grey-level) minus those corresponding to the vessel's walls (low grey-level), resulting in positive values of the derivative.
Chapter 3: The edge-tracking algorithm

The value of $\Delta y$ must keep certain requirements. It must be large enough to indicate the real edges within the intervals $(y'_i-\Delta y, y'_i+\Delta y) \ (i=1,...,4)$, but small enough so that the edges corresponding to impurities or bubbles around the vessel's walls are not included in those intervals. Therefore, the value of $\Delta y$ must be chosen as a compromise between both conditions. In any case, $\Delta y$ depends on the average value of the thickness of the vessel's walls, which itself depends on the way in which the images are taken. Unless otherwise stated, $\Delta y$ was initially assigned a default value of 10 pixels for the four edges in all of our images. This value was accepted as a suitable one after trying several others. However, the final value of $\Delta y$ may be different for each one of the four edges as it is explained as follows.

If the vessel is narrow enough, it may happen that both the upper external ($y_1$) and the lower internal ($y_3$) real edges are contained within the interval which correspond to one of them only, let us say $y_1$, for instance. Since the same criteria (the real starting edge is defined as the co-ordinate at which the derivative function takes its minimum value within the studied interval) are used to find $y_1$ and $y_3$, both edges may be confused. To avoid this problem, a smaller interval around the approximate starting edges must be considered in these case. The ultimate interval around each approximate edge is defined as follows in the algorithm:

1.- The interval around the approximate edge $y'_1$, in which the derivative is analysed is:

$$((y'_1-10), \min(y'_1+10, y'_2))$$

where $y'_1$ is the approximate upper external edge, $y'_2$ is the approximate upper internal edge, and $\min(y'_1+10, y'_2)$ means the minimum value between $(y'_1+10)$ and $y'_2$. Since $y'_2<y'_3$, this interval ensures that the minimum value of the derivative in this interval can only correspond to the real edge $y_1$, and never to $y_3$. Moreover, $y_1$
can never be confused with $y_2$ (even when both of them are within the studied interval) because the criterion is different for both of them: while for $y_1$ the minimum value of the derivative is required, $y_2$ is defined at the co-ordinate at which the derivative takes its maximum value.

2.- The interval around the approximate edge $y'_2$, in which the derivative is analysed is:

$$\left( (y'_2 - 10), \min(y'_2 + 10, y'_3) \right)$$

where $y'_2$ is the approximate upper internal edge, $y'_3$ is the approximate lower internal edge, and $\min(y'_2 + 10, y'_3)$ means the minimum value between $(y'_2 + 10)$ and $y'_3$. Since $y'_3 < y'_4$, this interval ensures that the maximum value of the derivative in this interval can only correspond to the real edge $y_2$, and never to $y_4$. Moreover, $y_2$ can never be confused either with $y_1$ or $y_3$ (even when the three of them are within the studied interval) because the criterion is different for the three of them: while for $y_2$ the maximum value of the derivative is required, $y_1$ and $y_3$ are defined at the co-ordinate at which the derivative takes its minimum value.

3.- The interval around the approximate edge $y'_3$, in which the derivative is analysed is:

$$\left( \max(y'_3 - 10, y'_2), (y'_3 + 10) \right)$$

where $y'_3$ is the approximate lower internal edge, $y'_2$ is the approximate upper internal edge, and $\max(y'_3 - 10, y'_2)$ means the maximum value between $(y'_3 - 10)$ and $y'_2$. This interval ensures that the minimum value of the derivative in this interval can only correspond to the real edge $y_3$, and never to $y_1$. Moreover, $y_3$ can never be confused with either $y_4$ or $y_2$ (even when the three of them are within the studied interval).
interval) because the criterion is different for the three of them: while for $y_3$ the minimum value of the derivative is required, $y_4$ and $y_2$ are defined at the co-ordinate at which the derivative takes its maximum value.

4.- The interval around the approximate edge $y'_4$, in which the derivative is analysed is:

$$ \left( \max(y'_4-10, y'_3), (y'_4+10) \right) $$

where $y'_4$ is the approximate lower external edge, $y'_3$ is the approximate lower internal edge, and $\max(y'_4-10, y'_3)$ means the maximum value between ($y'_4-10$) and $y'_3$. This interval ensures that the maximum value of the derivative in this interval can only correspond to the real edge $y_4$, and never to $y_2$. Moreover, $y_4$ can never be confused with $y_3$ (even when both of them are within the studied interval) because the criterion is different for both of them: while for $y_4$ the maximum value of the derivative is required, $y_3$ is defined at the co-ordinate at which the derivative takes its minimum value.

Figure 3.6 shows a flow diagram which illustrates the main steps to find the real starting edges in all possible situations.

### 2.3 Edge-tracking method

Once the real starting edges have been found, the next task is to find the edges along the rest of the vessel. An edge-tracking method is used to accomplish the task. The tracking method is based on the assumption that the position of the vessel’s edges along two consecutively-examined columns separated by the distance $\Delta x$ from each
Chapter 3: The edge-tracking algorithm

Figure 3.6 This block diagram illustrates how the algorithm finds the real starting edges in all possible situations. The branch on the left applies to standard vessels with 4 clear edges. The branch on the right applies to narrow vessels. Note that the top loop starts from the first column in the image and only finished when two or four changes in the intensity are found.
Chapter 3: The edge-tracking algorithm

other, is approximately the same if $\Delta x$ is sufficiently small. This is the same as assuming that the vessel’s profile varies smoothly along the $x$ axis.

To illustrate how the edge-tracking algorithm works, it is enough to consider how the upper external edge is tracked along the vessel length. The algorithm works similarly for the other three edges. Let us imagine that for the column $x_i$, the $y$ co-ordinate of the real upper external vessel’s edge is $y_{ri}$. According to the above assumption, the approximate position for the upper external edge in the column $x_{i+1}=x_i+\Delta x$ is also $y_{ri}$ (see Figure 3.3). The real co-ordinate of the upper external edge is then found from the approximate value $y_{ri}$ in the same manner as the real starting upper external edge was found from the approximate upper external starting edge (see Section 2.2, Real starting edges).

Beginning at the starting column, the above process is repeated along the length of the vessel’s upper edge incrementing and decrementing the $x$ co-ordinate by the amount $\Delta x$ each time. The same method is employed to track the internal upper edge and the internal and external lower edges of the vessel.

To choose a value for $\Delta x$ is a difficult task. This increment must be sufficiently small so that the variations of the edges between two columns separated by the amount $\Delta x$ are smooth. Moreover, the increment $\Delta x$ must be sufficiently large so that the number of columns analysed is enough to study the diameter variation along the vessel, which is the aim of the algorithm. Furthermore, the selection of $\Delta x$ depends on the value of $\Delta y$ (interval around the approximate edges over which the derivative is calculated) and vice versa. For a selected value of $\Delta y$, $\Delta x$ must be small enough for the position of the real edges in the column $(x_i+\Delta x)$ to be contained within the intervals around their respective approximate edges in that column (see Section 2.2). We have selected $\Delta y=10$ and $\Delta x=20$. Therefore, only thirteen columns of the image are analysed. This number of columns has been found sufficient to study the
diameter variations along the vessel with the advantage of a considerable reduction of the computation time.

Figure 3.7 shows a block diagram which explains how the edges are tracked along the vessel from the starting edges.

2.4 Diameter calculation

Once the edges of the vessel are found the calculation of the internal and external diameter is an easy task.

For images, like ours, where the vessel is displayed horizontally a column is equivalent to a perpendicular profile of the vessel. Therefore, the edges found in a column of the image are equivalent to the edges found in a perpendicular profile of the vessel. The internal and external diameter can be found by a simple subtraction between the y co-ordinates of the position of the edges found. The external diameter is calculated by subtracting the y co-ordinate of the upper external edge from the y co-ordinate of the lower external edge:

\[ D = y_4 - y_1 \]  

where, \( D \) is the external diameter, \( y_4 \) is the y co-ordinate of the lower external edge, and \( y_1 \) is the y co-ordinate of the upper external edge.

The internal diameter is calculated by subtracting the y co-ordinate of the upper internal edges from the y co-ordinate of the lower internal edges. Mathematically:

\[ d = y_3 - y_2 \]
Beginning from the column $x_{init}$, maintain the $y$ co-ordinate of the edges and increment the $x$ co-ordinate in $\Delta x$ pixels, $x_{n+1} = x_n + \Delta x$

Apply the derivative to the column $(x_{n+1})$ around the approximate edges

Find the real edges in the column $(x_{n+1})$

$\begin{cases} x_{n+1} > x_{255} - 20 \text{?} \\ \text{NO} \\ \text{YES} \\ x_{n+1} > x_{255} - 20 \text{?} \\ \text{NO} \\ \text{YES} \end{cases}$

Beginning again from the column $x_{init}$, maintain the $y$ co-ordinate of the edges and decrement the $x$ co-ordinate in $\Delta x$ pixels, $x_{n+1} = x_n - \Delta x$

Apply the derivative to the column $(x_{n+1})$ around the approximate edges

Find the real edges in the column $(x_{n+1})$

$\begin{cases} x_{n+1} > x_{1} - 20 \text{?} \\ \text{NO} \\ \text{YES} \end{cases}$

End

**Figure 3.7** A block diagram which illustrates how the edges are tracked along the vessel from the starting edges (at the column $x_{init}$) in both the forward direction (upper loop) until the last column is reached ($x_{223}$); and the backward direction (lower loop) until the first column is reached ($x_{1}$).
where, \( d \) is the internal diameter, \( y_3 \) is the \( y \) co-ordinate of the lower internal edge, and \( y_2 \) is the \( y \) co-ordinate of the upper internal edge.

3 GENERALISATION OF THE ALGORITHM FOR A TIME-VARYING IMAGE

So far, the functioning of the edge-tracking algorithm has been explained in the context of a single blood vessel image. However, the same algorithm can be applied to a continuous sequence of images illustrating the temporal variations of the vessel's edges. To analyse an image series an iterative process is implemented. In the iterative process each image of the series is individually analysed using the edge-tracking algorithm. Each image of the series is read from a video tape while it is being played. The image is then analysed to find the vessel's edges and the lumen and external diameter calculated. Once the image has been analysed the results are stored. This process is repeated continuously for every image of the series. For a recording of a certain duration, the shorter it takes to analyse every single image the more images can be analysed. This is why the analysis process for a single image needs to be done in the shortest possible time.

4 IMPLEMENTATION OF THE ALGORITHM

The proposed algorithm has been implemented to be used as a utility in a commercially-available image processing package named Visilog. This package provides a suitable interface to build and execute the functions necessary for the implementation of the edge-tracking algorithm. Another advantage of using Visilog is that some of the image processing functions which are needed for the implementation are already available in the libraries provided with this application.
Chapter 3: The edge-tracking algorithm

To be consistent with Visilog, all the code has been written in C programming language. However, some of the type definitions and function declarations particular to Visilog have been adopted in order to be able to use Visilog's functions. Once a function has been written it has to be compiled and linked with Visilog for it to be executed within this application. For a function to be linked to Visilog, some extra lines of code must be added to the adequate Visilog files. For a complete revision on how to add a customed function to Visilog see [20], Chapter 3.

Seven major functions have been written for the algorithm to be executed within Visilog. With these functions a total of three main programs can be executed. Each one of these programs appears as a menu option (see Appendix A) in the user graphical interface of Visilog [21]. Figure 3.8 gives an overview of the menu options added to Visilog. The menu option 'Static Edge' analyses a single blood vessel image stored in disk (see Appendix A). The option 'Live Edge' analyses a single image read directly from the video tape while it is being played, (see Appendix A). Finally, the option 'Dynamic Edge' analyses a series of images directly read from a video tape while it is being played (see Appendix A).

![Extensions Diagram]

**Figure 3.8** An overview of the menu options added to those already existing in the user interface of Visilog.
In all of the three options described above the vessel edges are detected, whether it is for a single image or for a series of images describing the contraction and expansion processes suffered by a blood vessel. Once the image has been analysed, the results are presented simultaneously in two different ways. 1) The detected edges are drawn on the top of the original image for direct comparison with the actual vessel's edges (see Figure 3.9 (b)). 2) The results are also stored in a text file for further analysis (see figure 3.9 (c)).

The options 'Static Edge' and 'Live Edge' also allow the user to obtain in the screen the lumen and external diameter values for the vessel in question. Once the vessel image has been analysed, the internal and external diameters at a particular position in the vessel may be displayed by clicking the left button of the mouse on the desired position of the vessel image.

Figure 3.9 (displayed in the next page) (a) A standard blood vessel. (b) The edges detected by the edge-tracking algorithm are displayed on the top of the original image. (c) The detected edges as plotted from the data stored in a text file.
Chapter 3: The edge-tracking algorithm

Figure 3.9 (See legend in page 45)
Chapter 4

Results

In this chapter, the results obtained with the edge-tracking algorithm are presented. The algorithm is applied to blood vessel images with different characteristics in order to understand its functioning and limitations. All results presented here were obtained with $\Delta x=20$ pixels (see Chapter 3, Section 2.3 Edge-tracking method). In other words, the vessel’s edges were found only at 13 of the 255 columns of which an image is comprised. Despite the few number of columns analysed, the information obtained is enough to study the spatial and temporal variations of the vessel’s dimensions along the whole of the vessel’s length without requiring long computation times.

All figures in which the results obtained with the algorithm are presented consist of two panels. The top panel shows the blood vessel image as recorded and read in by the edge-tracking algorithm. The bottom panel shows the vessel’s edges as detected by the algorithm. Although the vessel’s edges are detected only in 13 columns of the image, for the sake of illustration intermediate values are approximated by interpolation with a straight line between two successive detected edges. The detected edges are stored in a tab-delimited file and plotted using any suitable plotting package.
Figure 4.1 (a) A standard blood vessel. (b) Edges detected by the edge-tracking algorithm. Notice how the algorithm overestimates the wall thickness at those locations where the impurities touch the vessel's walls (x=40 and x=80).
Figure 4.1(b) shows the edges detected by the algorithm for an image (Figure 4.1(a)) recorded from a non-contracting blood vessel. Since the lumen of the vessel is perfectly distinguishable from the vessel’s walls, this vessel is regarded as a standard one. Therefore, the starting edges are detected as explained in Chapter 3, Section 2.2: Starting edges.

There are some impurities (fat and/or air bubbles) in the lumen from approximately \( x=25 \) to the end of the vessel. Because of the way in which the edge-tracking method operates, the impurities are always avoided as long as they are not touching the vessel’s walls. Notice how the algorithm successfully avoids this type of impurity. On the contrary, impurities touching the lower external wall at approximately \( x=40 \) and \( x=80 \) can not be eluded. Therefore, at those locations the algorithm overestimates the wall thickness.

It is important to notice that the starting edges can be found only because the impurities inside the lumen do not extend along the whole of the vessel’s length. If this was not the case, there would not be any column in the image with only the four edges corresponding to the vessel’s walls (see Chapter 3, Section 2.2: Starting edges). Instead, all columns would produce at least six edges: four corresponding to the vessel’s walls (upper external, upper internal, lower internal and lower external) and two corresponding to the impurities.

Figure 4.2(b) shows the edges detected by the edge-tracking algorithm for an image recorded (Figure 4.2(a)) from a contracting blood vessel. It can be seen that the contraction is not uniform along the vessel’s length. Notice how the algorithm detects the vessel edges even when the vessel is very narrow (right side of the image).

When finding the starting edges, this vessel can be treated as a standard or as a narrow one (see Chapter 3, Section 2.2: Starting edges). In general, if the blood
Figure 4.2 (a) A blood vessel partially contracted. (b) Result obtained with the edge-tracking algorithm. When finding the starting edges the vessel was regarded as a standard blood vessel.
vessel in question shows characteristics of both standard and narrow vessels, the starting edges can be detected by regarding the vessel as being of either type. In both cases the results will be the same. In order to find the starting edges, the edge-tracking algorithm begins to search from the column $x=0$ to the column $x=255$ (i.e., from the left to the right of the image). In this particular case (Figure 4.2), the lumen is perfectly distinguishable on the left hand side of the image and, therefore, the vessel is treated as a standard one. On the contrary, if the vessel was narrower on the left hand side, it would have been treated as a narrow vessel by the algorithm.

Notice that there is a fat cluster at approximately $x=25$ which can not be avoided because it is touching the vessel's walls. On the contrary, approximately between $x=130$ and $x=150$ there are some impurities that are eluded by the algorithm even though they are touching the vessel's boundaries. This is possible because the grey-level of the impurities is smaller than that of the walls and the algorithm has been designed to be sensitive to such differences.

Figure 4.3(b) shows the detected edges for the narrow blood vessel presented in Figure 4.3(a). In this image the vessel is highly contracted and the lumen is hardly distinguishable from the vessel's walls. Therefore, the vessel is regarded as a narrow one and the starting edges are detected as explained in Chapter 3, Section 2.2: Starting edges.

Notice that at approximately $x=25$ there are some impurities touching the upper external wall which cause an overestimation of the wall thickness at this co-ordinate. Despite this, the external edges are correctly detected. This is true for any type of vessel whatever type it is (standard or narrow). On the other hand, the detection of the internal edges is a difficult task. Very narrow vessels do not show the typical W-like cross-sectional profile characteristic of standard vessels. Instead, they show a U-like profile where the internal edges are not easily detected neither by the human eye.
Figure 4.3 (a) A narrow (fully contracted) blood vessel. (b) Edges detected by the edge-tracking algorithm. Because the vessel is very contracted it is very difficult for the algorithm (and even for the human eye) to detect the internal edges.
nor by the proposed algorithm. In any case, the edge-tracking method attempts to find some internal edges as shown in Figure 4.3(b). There is no guarantee that the detected internal boundaries correspond faithfully to the actual boundaries of the vessel but most times they are a good approximation.

Figures 4.4 and 4.5 show two examples where the edges detected by the edge tracking algorithm are incorrect. The vessel in the Figure 4.4(a) is the same as that in Figure 4.2(a). However, in Figure 4.4(a) the internal hair-like impurity is touching the vessel's lower wall. Because of the way in which the edge-tracking method operates, the hair-like impurity is erroneously regarded as the lower internal edge from the co-ordinate at which the impurity contacts the wall to the right end of the image. For this reason the estimated lumen diameter is smaller than the actual value.

On the other hand, as for the narrow vessel shown in Figure 4.3, the external diameter is correctly estimated. It should be noticed that because the hair-like impurity does not reach the left end of the vessel, it is possible for the algorithm to find the initial starting edges.

For the image presented in Figure 4.5(a) the algorithm detects the starting edges erroneously. In the image, there is a horizontal dark line at approximately \( y=75 \) that may have been caused by interference or damage of the video tape where the vessel was originally recorded. The algorithm arbitrarily detects the starting edges on this line. Consequently, the edge-tracking method considers this line as the upper wall of the vessel. This leads to incorrect estimation of the vessel's dimensions.

The reason the starting edges are found in the interference line rather than in the upper wall of the vessel is as follows. At approximately \( x=10 \) there is a local spot on the interference line darker than the vessel's upper wall. Indeed, because the starting edges are found on this spot, its grey-level is necessarily lower than the threshold applied (threshold=85), whereas the grey-level of the vessel's upper wall is higher.
Figure 4.4 (a) Image of a standard blood vessel with a hair-like impurity in the lumen. (b) Edges detected by the edge-tracking algorithm. Because the impurity touches the lower internal edge, the algorithm erroneously tracks the impurity instead of the lower internal edge.
Figure 4.5 (a) Image of a standard blood vessel. Notice the interference line at approximately $y=75$. (b) Results obtained with the edge-tracking algorithm. In this case the algorithm erroneously considers the interference line as the vessel’s upper wall (see text for details).
Chapter 4: Results

(see Figure 4.6). A possible solution to this particular problem is to use a higher threshold.

\[\text{Gray Level} \quad \begin{array}{c}
\text{threshold} \\
\hline
75 \quad 190
\end{array}\]

Figure 4.6 An approximate sketch of the cross-sectional profile at column \(x=10\) for the vessel presented in Figure 4.5(a). Notice how the grey level of the interference line \((y=75)\) is lower than the threshold, whereas that of the vessel's upper wall is higher. The starting edges are detected at around \(y=75\) and \(y=190\) (see Figure 4.5(b)).

The computation time to analyse each one of the vessels presented in Figures 4.1 to 4.5 is approximately 1 to 2 seconds in a 486DX2 (50MHz). The time of computation does not depend on the type of vessel, i.e. it is the same for standard and narrow blood vessels. However, the computation time does depend on the time that it takes for the algorithm to find the starting edges. This depends on the amount of impurities (e.g. fat clusters or bubbles in the lumen) in the image. The larger the number of impurities in and around the vessel, the more difficult it is to find the starting edges, and the longer the time of computation. The position of impurities in the image is also important. Since the starting edges are searched from the left to the
right side of the image, impurities on the left side of the vessel are more disturbing than those on the right. Therefore, the larger the number of impurities on the left side of the vessel the longer the computation time.

All results presented so far correspond to images taken at a single instant in time. This allows us to study the spatial variations of the vessels' boundaries. The edge-tracking algorithm has been designed and implemented so that these variations can be also studied in time. Figure 4.7 shows a series of images of a single blood vessel contracting and expanding in time. The series is comprised of thirty images sampled from a continuous video tape. The computation time to analyse the whole series is 48.23 seconds. Of those thirty images only ten are shown in Figure 4.7.

Notice how the blood vessel suffers two contractions in the period of time considered. The first contraction appears at approximately $x=75$ after 5 seconds (see Figure 4.7, second panel) and continues for 15 seconds. At time $t=20$ seconds the vessel begins to expand at $x=75$. Also, at this time, the vessel starts contracting on the right side of the image. This second contraction lasts until the end of the time interval studied.
Figure 4.7 **Left column:** A series of images sampled from a video recording of a single blood vessel contracting and expanding in time. There are two contractions: the first one at $t=5$ s. at $x=75$ and the second one from $t=20$ s. on the right hand of the image. **Right column:** Edges detected by the edge-tracking algorithm for the series shown on the left column. Notice that the images on the right column are smaller than those on the left one.
Chapter 4: Results

$t = 30s$

$t = 36s$

$t = 41s$

$t = 46s$
Chapter 5

An optical model of light transmission to correct the results obtained with the edge-tracking algorithm

In the previous chapters an edge-tracking algorithm to find the boundaries of a blood vessel in an image has been developed. It has been demonstrated (Chapter 4: Results) that, under appropriate conditions, this algorithm successfully detects the position of the vessel's edges according to what is seen in the image.

At this stage it is worth questioning whether the detected edges correspond to the actual edges of the vessel. Some doubt in this matter is provoked by the fact that, due to the optical diffusing properties of the vessel walls, scattering effects are likely to appear when recording the blood vessel images. These scattering effects would distort the formed image causing the observed edges to be a distorted version of the actual ones.

A plausible approach to answer the question formulated above is to compare the results obtained with the edge-tracking algorithm with some characteristic property of the vessels. In this direction, it is generally accepted that the vessels' walls are incompressible [22], i.e., the total volume of wall tissue remains constant.
independently of the estate of contraction or expansion of the vessel. If it is assumed that the incompressibility characteristic is a local property, the wall volume must remain constant locally. Therefore, the cross-sectional area of the vessels' walls must be constant in time, even when the vessels are contracting or expanding. On these grounds, it can be tested whether the cross-sectional area calculated from the results obtained with the edge-tracking algorithm remains constant at all times.

Figure 5.1 shows the time variation of the cross-sectional area of the walls of the vessel presented in Figure 4.7 (left column). The cross-sectional data presented in Figure 5.1 have been calculated from the vessel's inner and outer diameters (Figure 4.7, right column) assuming a circular cross-section. It is clear in the figure that the measured cross-sectional area is not constant in time.

**Figure 5.1.** The time-dependent variation of the cross-sectional area of the blood vessel series presented in Figure 4.7. The cross-sectional area, \( A \), is calculated as \( A = \pi b^2 - \pi a^2 \), where \( b \) is the external radius and \( a \) is the radius of the lumen.
Possible reasons for the inconsistency between the results obtained using the edge-tracking algorithm and the well established principle of incompressibility are:

1.- The vessel's walls are locally compressible.
2.- The vessel length varies in time.
3.- The edges detected by the edge-tracking algorithm do not correspond to the actual edges of the vessel.
4.- A combination of 1, 2, 3.

The first reason (namely, the vessel’s walls are locally compressible) is not plausible if it is accepted that the contractions suffered by the vessel occur locally and do not induce variations in the wall thickness at any other position along the vessel length. This is precisely the case in the example shown in Figure 4.7.

Regarding the second reason, it is possible that the cross-sectional area varies along the vessel’s length if the length of the vessel varies, even though the volume of tissue is constant. However, this is not considered here since the principles under which such changes might occur, if indeed they do so, are not established.

Finally, regarding the third reason, it is likely that the edges observed in the image (and, consequently, those detected by the edge-tracking algorithm) do not correspond faithfully to the actual edges of the vessel. Because of the illumination conditions under which the vessel images are produced, scattering effects caused by the diffuse illumination of the vessel lumen through the optically diffusing wall tissue will appear. The scattering effects distort the shape of the original vessel making its walls appear to be thicker in the recorded image than they actually are. This leads to an overestimation of the wall thickness detected by the algorithm and, therefore, to an underestimation of the lumen.
In the next sections a model of light transmission [23] is proposed to correct the results obtained with the edge-tracking algorithm assuming the latter explanation is the cause of the disagreement between the general assumption of the wall incompressibility and the measured cross-sectional areas.

1 THE MODEL

The structure of blood vessel images formed under diffused white light can be described using the Beer-Lambert's law [16]. This law describes the transmitted light through an absorbing medium. Beer-Lambert's law states that the light intensity transmitted through an absorbing medium is inversely proportional to the exponential of the optical density. In one dimension it is mathematically expressed as:

\[ T(x) = \exp(-OD(x)) \]  \hspace{1cm} (1)

where \( T(x) \) is the transmitted light through the medium along the \( x \) direction and \( OD \) is the optical density (sometimes called absorbance). The optical density is proportional to the path length, \( l(x) \), travelled by the light through the medium:

\[ OD(x) = \gamma l(x) \]  \hspace{1cm} (2)

where \( \gamma \) is the absorption coefficient which is characteristic of the medium.

For a blood vessel with circular cross-section (see Figure 5.2) the transmitted light intensity according to Beer-Lambert’s law in one-dimension is given by:

\[ T(x) = T_0 \exp\left[-\gamma \left( R \sqrt{b^2 - (x-x_0)^2} - R \sqrt{a^2 - (x-x_0)^2} \right) - \gamma \sqrt{a^2 - (x-x_0)^2}\right] \]  \hspace{1cm} (3)
where $T_0$ is the intensity of the incident light, $\gamma_B$ is the absorption coefficient of the wall tissue, $\gamma_A$ is the absorption coefficient of the vessel’s lumen, and $x_0$ is the coordinate of the centre of the vessel with respect to the defined co-ordinates system. \(\Re\) denotes the real part. The absorption coefficients $\gamma_A$ and $\gamma_B$ are assumed to be constant inside their respective media. In equation (3) the term \(\sqrt{b^2-(x-x_0)^2} - \sqrt{a^2-(x-x_0)^2}\) refers to the path length travelled by the light through the vessel’s wall, and $\sqrt{a^2-(x-x_0)^2}$ to the path length through the lumen [15].

After reorganising the terms in equation (3) the following expression for the transmitted light is obtained:

$$T(x) = T_0 \exp\left[-(\gamma_A - \gamma_B)\Re\sqrt{a^2-(x-x_0)^2} - \gamma_B\Re\sqrt{b^2-(x-x_0)^2}\right]$$ (4)

Figure 5.2 The transmitted light, $T(x)$, through a blood vessel with circular cross-section when the vessel is illuminated with white diffuse light of intensity $T_0$ from the left. $a$ represents the radius of the lumen and $b$ the external radius of the vessel.
Chapter 5: An optical model of light transmission

On the right hand side of Figure 5.2 the transmitted light, $T(x)$, through a circular blood vessel illuminated with white diffused light of intensity $T_0$ is shown. It is seen that the transmitted light through the lumen is larger than that through the walls. This allows to distinguish the vessel’s walls from the lumen in blood vessel images.

![Figure 5.3](image)

**Figure 5.3.** Curves showing normalised values of the transmitted intensity through the vessel for several values of the lumen radius, $a$, (between $a=17$ and $a=84$, five curves). These curves have been represented assuming the cross-sectional wall area is the same ($A=5000$) for all the considered values of $a$. The black dots indicate the position of the lumen edges.

Assuming the cross-section of the vessel is circular, there is a simple relationship between the external and the lumen radius:

$$b = \sqrt{\frac{A}{\pi} + a^2}$$ (5)
where $A$ is the cross-sectional area of the vessel's walls.

Figure 5.3 shows normalised values of $T(x)$ for different values of the lumen radius, $a$, assuming constant cross-section. The black dots indicate the actual position of the internal edges of the vessel. In other words, the distance between each dot and the y axis (ordinate axis) represents the value of the lumen radius for each value of $a$.

2 APPLICATION OF THE MODEL TO CORRECT THE RESULTS OBTAINED USING THE EDGE-TRACKING ALGORITHM

As it has been introduced before, because of the scattering effects that appear when the blood vessel is illuminated with white diffused light, the vessel edges observed in the formed image do not correspond to the actual edges of the vessel. Therefore, the edges detected by the edge-tracking algorithm must be a distorted version of the actual edges. A plausible approach to correct the results obtained with the algorithm is to establish a relationship between the experimental vessel profile and the profile obtained using the optical model of light transmission presented in the previous section. In other words, to establish a relationship between the experimental and the actual dimensions of the blood vessels.

As a first order approximation, it is assumed that the experimental intensity profile can be obtained from the model by performing a low-pass filter operation on the model profile given by (4). Mathematically this can be expressed as:

$$T_{exp}(x) = \int T(x') h(x - x') dx'$$  \hspace{1cm} (6)
where $T_{\text{exp}}(x)$ is the experimental intensity profile, $T(x)$ is the theoretical profile defined using the Beer-Lambert model described in equation (4), and $h(x-x')$ is a weighting factor that defines the type of filter applied. If, for instance, $h(x)=1 \forall(x)$, the filter operation is the common average.

Figure 5.4 shows how the filtered model profile (thick line) can be fitted to the experimental intensity profile (triangles) using the simplest possible filter, i.e., $h(x-x')=1$ in equation (6). The modelled results were obtained assuming $\gamma_A=0$, $\gamma_B=0.027$ and a cross-sectional area of the wall, $A=5000$. It was also assumed that the vessel’s walls are incompressible, i.e. $A$ is constant for all the values of the lumen radius, $a$, considered. The external radius, $b$, was calculated using equation (5).

![Figure 5.4](image)

**Figure 5.4.** The Beer-Lambert model (thin line) and filtered Beer-Lambert model (thick line) for a range of vessel profiles with constant cross-section ($A=5000$), $\gamma_A=0$, $\gamma_B=0.027$ and five different lumen radii between $a=17$ and $a=84$. A real profile (triangles) taken from the image in Figure 4.2 is also shown for comparison.
In Beer-Lambert's model of light transmission for a blood vessel with circular cross-section the internal edge is located at the minimum of the transmission function, i.e. at the minimum of the model profile (see Figure 5.3). On the other hand, in the filtered model the internal edge is placed at the minimum of the gradient of the filtered transmission function, i.e. at the minimum of the gradient of the filtered profile. Figure 5.5 shows that there is a discrepancy between the internal edges predicted by these two methods. In other words, it shows that the minimum of the model profile does not coincide with the minimum of the gradient of the filtered profile. Moreover, the internal edges predicted by the model are further from the y axis than those calculated as the minimum of the gradient of the filtered profile. Therefore, the lumen radius predicted by the model is larger than that predicted by the filtered model.

Figure 5.5. The Beer-Lambert model (thin line), filtered Beer-Lambert model (thick line) and the gradient of the filtered Beer-Lambert model (dotted line) are shown for a range of vessel profiles with constant cross-section (A=5000) and different lumen radius between \(a=17\) and \(a=84\).
It has been demonstrated that a correspondence between the experimental profile and the filtered theoretical profile can be established (see Figure 5.4). In addition, in both cases, the filtered model and the experimental profiles, the vessel's internal edges are detected at the minimum of the gradient of the transmission profile. Therefore, all that has been said above for the filtered profile is also applicable for the experimental transmission profile. On the basis of the 'equivalency' between the filtered model and the experimental profiles, it is clear that the edge-tracking algorithm would produce a underestimation of the lumen diameter. The results in Figure 5.5 show that the underestimation in this case is of the order of five pixels, which approximately corresponds to a correction of 10% of the detected lumen radius.

3 DISCUSSION

The actual lumen radius is obtained from the experimental transmission profile as follows. From the diameter values detected by the edge-tracking algorithm when the vessel is at maximum expansion, the cross-sectional area, $A$, can be calculated. This is possible if we are ready to accept that at maximum expansion the scattering effects (see above) are minimal for both the internal and external radii and, therefore, the detected edges correspond to the actual edges of the vessel. With the measured value of $A$ and the experimental values of $\gamma_A$ and $\gamma_B$, the actual lumen radius, $a$, at any state of compression or expansion will be given by that value which produces the best fit of the filtered model profile to the experimental transmission profile. Obviously, the incompressibility of the vessel's wall is also assumed, so that $A$ is the same for all values of $a$ and, hence, for any state of compression. This is the approach that has been followed above to illustrate the proposed model.

We have devised a possible approach to determine the actual vessel's diameters from the experimental transmission profiles. Ideally, however, the corrected values of the vessel's diameters should be obtained directly from the edges detected by the edge-
tracking algorithm without having to analyse the experimental profiles of the vessel. Statistical studies must be carried out in order to determine the most appropriate filter so that the filtered model profiles adjust systematically to the experimental ones. In other words, to determine an analytical expression of $h(x-x')$ in equation (6) which relates the actual vessel radii with the measured values obtained with the edge-tracking algorithm. Alternatively, a different theory of light transmission through absorbing media which intrinsically considers scattering effects could be developed.

Once the analytical relationship between the theoretical and the experimental radii is established, the correct lumen radius can be obtained from such a relationship simply by assuming that the cross-sectional area of the vessel's wall can be calculated from the measured diameters at maximum expansion of the vessel. This is exactly the same assumption as the one explained above. Once the value of $A$ is known, equation (5) relates the two unknown theoretical radii, $a$ and $b$. The analytical relationship (equation (6)) between the experimental and actual radii reduces, therefore, to an equation with a single unknown variable, namely the actual lumen radius, $a$.

It should be noticed that the values of $A$, $\gamma_A$ and $\gamma_B$ used above are arbitrary. To date we have not been able to find any published data regarding the absorbing coefficients $\gamma_A$ and $\gamma_B$. Regarding the cross-sectional area of the wall, $A$, the value used above has the only purpose of illustrating the functioning of the proposed optical model. Even though that value was not calculated at maximum expansion, but is rather arbitrary, it is obvious that similar behaviour would been obtained had we used an actual calculated value. In the latter case, however, the best fit to the experimental transmission profile would have been surely obtained for a different value of $a$. 
Chapter 6

General discussion and Conclusions

In this work an image processing system to perform a dimensional analysis of blood vessel has been developed. This system combines image processing techniques with an optical model of light transmission through absorbing media.

Using image processing techniques an algorithm to analyse blood vessel images has been developed. The algorithm finds the edges of a blood vessel combining an edge-tracking method with common techniques of image processing, including gradient operators and threshold techniques. Once the edges of the vessel have been found, the external and the internal vessel diameters are calculated.

It is important to notice that the proposed algorithm has been designed to analyse the morphological changes of blood vessels on a perfusion myogram. The images are recorded by video-microscopy techniques in a video tape. By connecting a video player to the input port of an image card connected to a computer, the recorded images are read into the computer where the image analysis is carried out. Although the basic principles of the algorithm are general, the implementation is specific for the image processing package Visilog. This allows us to use some of the image analysis functions provided in the package and the graphical user front-end of Visilog.
Chapter 6: General discussion and Conclusions

The main target to be accomplished with this project was to develop a tool to study the spatial and temporal variations of blood vessel's dimensions. As explained in Chapter 2 (Background), this requires a fully automatic analysis method, both in time and space. The proposed edge-tracking algorithm successfully keeps this condition. No user interaction is necessary to either detect the vessel diameters along the vessel's length in the image, or their variation in time. The implemented algorithm automatically reads an image from the video tape, analyses it and displays the detected vessel's edges over the original sampled image. Additionally, the position of the detected edges and the internal and external diameters of the vessel are stored in a text file for further reference. For the temporal analysis this process is automatically repeated for every read image.

The analysis of each blood vessel is done in quasi-real time. This is possible because the algorithm only detects the vessel's edges in 13 of the 256 columns of which the image is comprised. Although the analysis could be done for each one of the 256 columns, no further information about the spatial variations of the blood vessel's dimensions would be obtained. However, the computational time would be much longer making the real-time temporal analysis impossible.

One of the main problems encountered when designing a method for dimensional analysis of blood vessels is the presence of impurities and air bubbles in the image (inside and around the vessel). The results presented in Chapter 4 show that the proposed edge-tracking algorithm successfully overcomes these problems as long as the impurities or bubbles do not touch the vessel walls. Because of the way in which the edge-tracking algorithm operates, small impurities in contact with the vessel only lead to a local overestimation of the wall thickness. However, larger impurities placed parallel to the vessel can produce a serious disagreement between the detected edges and the actual edges of the vessel. A clear solution to this problem is to develop an improved method of dissecting and cleaning the blood vessel.
For the temporal analysis of the images, it is essential to have a continuous sequence of events. If in the middle of a series of blood vessel images contracting and expanding in time there is a blank interval (i.e., a 'cut' in the series), it will be interpreted as another blood vessel image to be analysed. If the image is blank, the algorithm still attempts to detect the vessel's edges, but fails and stops the analysis.

At the moment, the algorithm automatically distinguishes two types of vessels, namely, standard and narrow vessels (see Chapter 3, Section 2.2). The tracking method has been designed to track always the four edges of the vessel. For very narrow vessels the lumen is not always clear and, therefore, further analysis (contrast enhancement) needs to be carried out in order to detect the internal edges of the vessel. The distinction between narrow and standard vessels would not be necessary if the contrast between the lumen and the wall is the same for both vessel types. This can be achieved by introducing a contrast agent inside the vessel at the time of recording.

From the above discussion, an optimum functioning of the edge-tracking algorithm would be obtained provided the following:

1.- The number and the size of the impurities which touch the vessel's walls in the image are reduced to a minimum. Ideally they would be absent.

2.- The images are recorded using a contrast agent in the lumen.

3.- For temporal analysis, there are no cuts in the sequence of blood vessel images.

Despite all efforts made to obtain optimum edge detection by the edge-tracking algorithm, there will always be a disagreement between the detected and the actual edges of the vessel. This, as explained in Chapter 5, is caused by the scattering effects which appear when the images are formed under white-light diffused illumination. In our case, the scattering effects are actually caused by the diffuse
illumination of the lumen through the optically diffusing wall tissue. The scattering effects are most acute when the vessel is maximally contracted. Because of the incompressibility of the wall tissue, the more contracted the vessel, the thicker the walls and, therefore, the more diffusing.

A possible approach to correct the dimensional data obtained with the edge-tracking algorithm has been proposed and tested in Chapter 5. The correction is established by comparing experimental light transmission profiles of the vessel with modelled profiles. A model based on Beer-Lambert’s law of light transmission through absorbing media has been developed and used to generate the modelled profiles.

It should be understood that the proposed model constitutes only a first order approach to solve our problem. Further work should be carried out to develop, if at all possible, an analytical solution to the relationship between the actual and the detected edges of the vessel. Nevertheless, an illustration and discussion to the way in which our optical model can be employed to correct the detected edges has been given. In the example given the actual lumen radius has been found to be five pixels smaller than the actual radius. Future statistical analysis in required to confirm the validity of these preliminary results across a range of vessel undergoing complete cycles of contraction and expansion.

FURTHER RESEARCH

Following the above discussion, some future work can be proposed in order to obtain an improved method for the dimensional analysis of blood vessels. The method can be improved at both levels, the edge-tracking algorithm and the optical model to correct for the scattering effects.

Regarding the edge-tracking algorithm, the next topics should be addressed:
1.- For narrow vessels, a contrast-enhancement technique could be applied to the whole image to increase the contrast between the lumen and the vessels walls. This contrast operation has not been performed in the proposed algorithm because we have been unable to find a suitable function which requires a reasonable time for computation (notice that in the proposed algorithm, the contrast operation is applied only to the column where the external starting edges are found and not to the whole image).

2.- For a series of images, a function which checks that none of the read images is blank should be written. This would make the temporal analysis possible even when there are some cuts in the video sequence.

Regarding the optical model for correcting the experimental results, further research should be carried out to cover the following issues:

1.- To perform a statistical analysis to obtain a general type for the filter with which the experimental transmission profiles are modelled from the Beer-Lambert's law of light transmission.

2.- To generalise the proposed optical model from one to two dimensions.

3.- To develop a physical model of light transmission which intrinsically considers scattering effects. Notice that the scattering effects in the proposed model of light transmission are simulated by a low-pass filter applied to Beer-Lambert's law.

Once a suitable method for correction has been developed, it should be implemented together with the edge-tracking algorithm so that the edges detected by the whole system correspond directly to the actual edges of the vessel.
References


Appendix A

Code listings

This appendix contains the C code written to implement the edge-tracking algorithm. It contains the necessary code to analyse the blood vessel images in the three following cases (see Chapter 3, Section 4):

1.- Analysis of a blood vessel image stored in disk:
   Associated menu option: ‘Static Edge’.

2.- Analysis of a blood vessel grabbed from a video tape:
   Associated menu option: ‘Live Edge’.
   Related main functions: IpGrabEdge.

3.- Analysis of a series of blood vessel images varying in time:
   Associated menu option: ‘Dynamic Edge’.
   Related main functions: IpTimeEdge, IpTimeMemr, IpTimeFree.

In addition to these main programs, a header file named ‘ipe.h’ is listed. This file includes the declaration of those functions which are accessed by the three main programs. The necessary code to display and execute the three menu options within the graphical user interface of Visilog is also listed. This code is presented in the sections ‘display menu options’ and ‘menu functions’ of this appendix.
Appendix A: Code listings

#include "v4incl.h"
#include "ipe.h"
#define THS 85
#define INFINITY 3e38f

#ifndef global variables
long *diameter[3];

FRONT END CODE

V4Ecode IpMyEdge(char*fnamein, char*fnameout, char *iname)
{
    return RunProc ("IpMyEdge", fnamein, fnameout, iname);
}
V4Ecode IpMyDiamet (long column, long extint[2])
{
    return RunProc ("IpMyDiamet", column, extint);
}
V4Ecode IpMyFree ()
{
    return RunProc ("IpMyFree");
}

*******************************************************************************
/**************************************************** MAIN FUNCTION (IpMyEdge) ****************************************************

* This function reads an image (.tif) from a file (fnamein) stored in disk.
* finds the first column with 2 or 4 edges using the Lut_function
* finds the edges at first point using: Contrast (if 2 edges found)
* Threshold=80 (if 4 edges found)
* finds the final edges using: the Derivative function.
* draws the result on the top of the image and displays this image inside Visilog
* finds the external and the internal diameters (and stores them in file 'diameter')
* stores the data (edges) in a file (fnameout).

*******************************************************************************/

V4Ecode amyEdge (char*fnamein, char*fnameout, char *iname) 
{
    IMAGE img;
    int j,n,init;
    long xinit, x, xmin, xmax, ymin, ymax, initedge[4];
    long *edge[4];
    FILE *errprog;
    ProcName ("amyEdge");

    img=OpenImage (fnamein);
    if (img==NULL) 
    
    for (j=0; j<4; j++) 
        free (edge[j]);
    return ABORT;
    }
    for (j=0; j<4; j++)
    edge[j]=(long*)malloc((img->gx/20+1)*sizeof(long));
    xmin=1;
    ymin=10;
    xmax=img->gx;
    ymax=img->gy-10;
    xinit=InitPoint2_4 (img, xmin, xmax, ymin, ymax, &init);
    if (xinit==0) 
        for (j=0; j<4; j++)
            free (edge[j]);
        return ABORT;
    if (init==4) 
        InitialEdges4 (img, initedge, xinit, ymin, ymax);
    } else if (init==2) 
        InitialEdges2 (img, initedge, xinit, ymin, ymax);
    } else if (init! =2 && init!=4) 
        return ABORT;
    }
    for (j=0; j<4; j++) 
    
    if (j==1)
        edge[j][n=(int)((xinit-1)/20)]=NewEdge (img, xinit, initedge[j], initedge[j+1], j, j%2);
    if (j>2)
        edge[j][n=(int)((xinit-1)/20)]=NewEdge (img, xinit, initedge[j], initedge[j-1], j, j%2);
    }
for (x=xinit+20; x<xmax; x+=20) {
    for (j=0; j<4; j++) {
        if (j<1)
            edge[j][n+1]=NewEdge (img, x, edge[j][n], edge[j+1][n], j, j%2);
        if (j>=2)
            edge[j][n+1]=NewEdge (img, x, edge[j][n], edge[j-1][n+1], j, j%2);
    }
    n++;
}

n=(int)((xinit-1)/20);
for (x=xinit-20; x>xmin; x-=20) {
    for (j=0; j<4; j++) {
        if (j<1)
            edge[j][n-l]=NewEdge (img, x, edge[j][n], edge[j+l][n], j, j%2);
        if (j>=2)
            edge[j][n-l]=NewEdge (img, x, edge[j][n], edge[j-1][n+1], j, j%2);
    }
    n--;
}

DrawResult (img, edge, xinit-20*(xinit/20), xmax);
IpRead ("lmEdge", iname, V_FTYPE_AUTO);
for (j=0; j<3; j++)
    diameter[j]=(long*)malloc((img->gx/20+1)*sizeof(long));
if (!Diameter (edge, xinit-20*(xinit/20), xmax, diameter)) {
    for (j=0; j<4; j++)
        free (edge[j]);
    return ABORT;
}
if (!StoreData (fnameout, edge, xinit-20*(xinit/20), xmax)) {
    for (j=0; j<4; j++)
        free (edge[j]);
    return ABORT;
}
for (j=0; j<4; j++)
    free (edge[j]);
return NOERROR;

/***********************************************************************************
** MAIN FUNCTION (IpMyDiamet) *****************************/
* *
* This function receives a column of the image.
* finds the external and internal diameter at that column
***********************************************************************************/ 
V4Ecode amyDiamet (long column, long extint[2])
{
    long    x;
    FILE    *errdiamet;
    ProcName("amyDiamet");
x=(column-diameter[0][0])/20;
extint[0]=diameter[1][x]; //External diameter
extint[1]=diameter[2][x]; //Internal diameter
return NOERROR;

/*******************************
MAIN FUNCTION (IpMyFree)
*******************************

V4Ecode amyFree()
{
    int i;
    ProcName("amyEdge");
    for (i=0; i<3; i++)
        free (diameter[i]);
    return NOERROR;
}

FUNCTION DEFINITIONS

IMAGE OpenImage (char*imageFileName)
{
    IMAGE f;
    IFORMAT format;
    FILE *errfile;

    format = IpAllocFormat();
f=IpOpenImage("/visilog4/image/tubetest", V_READ, format); //if the format is .tif
f=IpOpenImage("ImEdge", V_WRITE, format);
if (f==NULL) {
    errfile=fopen("errfile", "a");
    fprintf (errfile, "JpOpenImage() error=%ld\n", V4LError());
    fclose (errfile);
    return NULL;
}
IpRead (imageFileName, f, V_FTYPE_AUTO);
IpCloseImage("/visilog4/image/tubetest");
Appendix A: Code listings

/* Appendix A: Code listings */

IMAGE GrabImage()
{
  FORMAT    format;
  IMAGE     f;
  FILE      *errfile;

  format=IpAllocFormat();
  IpGetCurLogCamFormat (format);
  f=IpOpenImage ("ImEdge", V_WRITE, format);
  if (f==NULL) {
    errfile=fopen ("errfile", "a");
    fprintf (errfile, "IpOpenlmage() error=%ldx\n", V4LLError());
    fclose (errfile);
    return NULL;
  }
  IpGrabImage (f, IL);
  return f;
}

/* Appendix A: Code listings */

long InitPoint2_4 (IMAGE img, long xmin, long xmax, long ymin, long ymax, int *init)
{
  int    i=0, two, four;
  long   x, y;
  float  *lut;
  FILE   *errfile;

  four=0;
  two=0;
  *init=0;
  lut=(float*)malloc(img->gy*sizeof(float));
for (x=xmin+1; x<Xmax; x++) {
    Lut (img, x, lut);
    i=0;
    for ( y=4; y<ymax-4; y++) { 
        if (((lut[y]<=THS) && (lut[y-l]>THS)) || ((lut[y]>THS) && (lut[y-1]<=THS)))
            i++;
        if (i==4)
            four++;
        if (i==2)
            two++;
        if (two==1) {
            free(lut);
            *init=2;
            return x;
        }
        if (four==1) {
            free(lut);
            *init=4;
            return x;
        }
    }
    errfile=fopen ("errfile", "a");
    fprintf (errfile, "failed
to find point with four edges 4\n");
    fclose (errfile);
    free (lut);
    return 0;
}

/*******************************
* InitialEdges2
* This function receives an image and the value xinit
* and attempts to find the edges at xinit using contrast and threshold.
* The function returns initedges which are the approximate edges at xinit.
*******************************/

long*InitialEdges2 (IMAGE img, long* edge, long x, long ymin, long ymax) {
    int j, ytwo[2];
    long y;
    float contmax, contmin, cont;
    float *contrast, *lut;

    lut=(float*)malloc(img->g*y* sizeof(float));
    j=0;
    Lut (img, x, lut);
    for ( y=4; y<ymax-4; y++) { 
        if (((lut[y]<=THS) && (lut[y-1]>THS)) || ((lut[y]>THS) && (lut[y-1]<=THS)))
            if (j%2==0)
                ytwo[j]=y;
            else
                ytwo[j]=y-1;
j++;
}
}
contmax=0;
contmin=255;
contrast=(float*)malloc(img->gy*sizeof(float));
Contrast (img, x, contrast);
for (y=ytwo[0]; y<=ytwo[1]; y++) {
    contmax=max(contmax, contrast[y]);
    contmin=min(contmin, contrast[y]);
}
cont=(contmax+contmin)/2;
y=ytwo[0];
while (contrast[y]<cont & & y<=ytwo[1])
    y++;
edge[1]=y;
y=ytwo[1];
while (contrast[y]<cont & & y>=ytwo[0])
    y--;
edge[2]=y;
edge[0]=ytwo[0];
edge[3]=ytwo[1];
free(lut);
free(contrast);
return edge;

*************************************************************************** InitialEdges4 ***************************************************************************
* *
* This function receives an image and the value
* xinit and attempts to find the edges at the column specified by xinit using THS.
* The function returns initedges which are the approximate edges at xinit.
***************************************************************************
long*InitialEdges4 (IMAGE img, long*edge, long x, long ymin, long ymax)
{
    int j;
    long y;
    float *lut;
    lut=(float*)malloc(img->gy*sizeof(float));
    j=0;
    Lut (img,x,lut);
    for (y=ymin; y<ymax-4; y++) {
        if (((lut[y]<=THS) & & (lut[y-1]>THS)) || ((lut[y]>THS) & & (lut[y-1]<=THS)))
            if (j%2==0)
                edge[j]=y;
            else
                edge[j]=y-1;
        j++;
    }
```c
long NewEdge (IMAGE img, long x, long lasty, long lastyud, int e, int topEdge)
{
    int y, edge;
    long xmed, upy, downy;
    float der, min;

    der=0;
    if (topEdge)
        min=-INFINITY;
    else
        min=INFINITY;
    if (e<=1) {
        upy=lasty-10;
        downy=min (lasty+10,lastyud);
    }
    if (e>=2) {
        upy=max (lasty-10, lastyud);
        downy=lasty+10;
    }
    for (y=upy; y<=downy; y++) {
        der=Derivative (img, x, y);
        if (topEdge) {
            if (der>min) {
                min=der;
                edge=y;
            }
        }
        else {
            if (der<min) {
                min=der;
                edge=y;
            }
        }
    }
    y=edge;
    return y;
}
```
Appendix A: Code listings

/**************************** StoreData ****************************/

This function receives a file name and attempts to store data in that file.
* The function returns the value 1 unless the file is not valid, in which case
* 0 is returned.

int StoreData (char*fileName, long**edge, long xinit,long xmax)
{
    int x;
    FILE *file, *errfile;
    file=fopen (fileName, "a");
    if (file==NULL) {
        errfile=fopen ("errfile","a");
        fprintf(errfile, "Can't open fileout");
        fclose(errfile);
        return 0;
    }
    fprintf(file, "x	y I 
y2	y3
");
    for (x=0; xinit+x*20<xmax; x++) {
        fprintf(file, "%ld	%ld	%ld	%1d	%1d
", xinit+x*20, edge[0][x], edge[1][x], edge[2][x],
            edge[3][x]);
    }
    fclose (file);
    return 1;
}

/**************************** Diameter ****************************/

This function receives a matrix (edge) with the positions of the vessel’s edges along the length
of the vessel and attempts to find the lumen and external diameters.
* The function returns the value 1

int Diameter (long**edge, long xinit, long xmax, long**diameter)
{
    int x;
    for (x=0; xinit+x*20<xmax; x++) {
        diameter[0][x]=xinit+x*20;
        diameter[1][x]=edge[3][x]-edge[0][x];
        diameter[2][x]=edge[2][x]-edge[1][x];
    }
    return 1;
}
Appendix A: Code listings

/*******************************
DrawResult *************************************/

int DrawResult (IMAGE img, long **edge, long xinit, long xmax)
{
    int x, i;
    long num;
    float color[256];

    for (i = 0; i < 256; i++)
        color[i] = 256;

    for (x = 0; xinit + x * 20 < xmax - 20; x++)
    for (y = 0; xinit + x * 20 < xmax - 20; y++)
    {
        IpWriteLine (img, xinit + x * 20, edge[0][x], xinit + (x + 1) * 20, edge[0][x + 1], &num, color);
        IpWriteLine (img, xinit + x * 20, edge[1][x], xinit + (x + 1) * 20, edge[1][x + 1], &num, color);
        IpWriteLine (img, xinit + x * 20, edge[2][x], xinit + (x + 1) * 20, edge[2][x + 1], &num, color);
        IpWriteLine (img, xinit + x * 20, edge[3][x], xinit + (x + 1) * 20, edge[3][x + 1], &num, color);
    }
    return 1;
}

/*******************************
Contrast *************************************/

float *Contrast (IMAGE img, long x, float *contrast)
{
    int i;
    long y, ymax;
    float *dil, *ero, *acuml, *acum2;

    ymax = img->gy;
    dil = (float*)malloc(img->gy * sizeof(float));
    ero = (float*)malloc(img->gy * sizeof(float));
    acum1 = (float*)malloc(img->gy * sizeof(float));
    acum2 = (float*)malloc(img->gy * sizeof(float));

    for (y = 1; y < ymax; y++)
    {
        IpReadPix (img, x, y, &contrast[y]);
        dil[y] = contrast[y];
        ero[y] = contrast[y];
    }

    for (i = 1; i <= 10; i++)
    for (y = 3; y < ymax - 1; y++)
    {
        acum1[y] = max(dil[y], dil[y - 1]);
    }
}

90
acuml[y+1]=max(acuml[y], dil[y+1]);
acum2[y]=min(ero[y], ero[y-1]);
acum2[y]=min(acum2[y], ero[y+1]);
}
for (y=4; y<ymax-2; y++)
{  
dil[y]=acuml[y];
ero[y]=acum2[y];
}
for (y=4; y<ymax-2; y++)
{  
  if ((dil[y]-contrast[y])<contrast[y]-ero[y])
    contrast[y]=dil[y];
  else
    contrast[y]=ero[y];
}
free(ero);
free(dil);
free(acuml);
free(acum2);
return contrast;

/**************************** Lut ****************************

This function receives an image and a column and applies the
lut transformation to that column.
The function returns a pointer to a long (lut). The pointer is a matrix with the new values
of the pixels of that column.

****************************

float *Lut (IMAGE img, long x, float *lut)
{
  long     y, ymax;
  float    val, maxval, minval;

  IpReadPix (img, x, 3, &val);
  maxval=val;
  minval=val;
  ymax=img->gy;
  for (y=3; y<ymax-3; y++)
  {  
    IpReadPix (img, x, y, &val);
    maxval=max(maxval, val);
    minval=min(minval, val);
  }
  for (y=3; y<ymax-3; y++)
  {  
    IpReadPix (img, x, y, &val);
    lut[y]=(255*(val-minval))/(maxval-minval);
  }
  return lut;
}
/******************** Derivative ********************
* This function receives an image, a column (x) and a row (y).
* It attempts to find the derivative in the y-direction of the average in the x-direction.
* The function returns the value der that is the derivative
********************/

float Derivative (IMAGE img, long x, long y)
{
    float less, more, mid, der;
    less=AverageX (img, x, y-1);
    mid=AverageX (img, x, y);
    more=AverageX (img, x, y+1);
    der=(more-less)/2/mid;
    return der;
}

/******************** AverageX ********************
* This function receives an image, a column (x) and a row (y).
* It attempts to find the average of (x,y), (x-1,y) and (x+1,y)
* The function returns a value acum which is the calculated average.
********************/

float AverageX (IMAGE img, long x, long y)
{
    int i,j;
    float val, acum;
    acum=0;
    for(i=x-1; i<=x+1; i++)
    {
        IpReadPix (img, i, y, &val);
        acum+=val;
    }
    acum=acum/9;
    return acum;
}
Appendix A: Code listings

#include "v4incl.h"
#include "ipe.h"

#define THS 85
#define INFINITY 3e38f

/* ANALYSIS OF A BLOOD VESSEL IMAGE GRABBED FROM A VIDEO TAPE *

* Unless otherwise stated this code has been written by:
* 
* Almudena Eustaquio-Martin.
* Optical Engineering Group
* Loughborough University of Technology
* Loughborough (LE11 3TU)
* U.K.
*
* 1995.
*
* This section contains the main function necessary to build the menu option Live Edge:
* IpGrabEdge
*
* Although every effort has been made to ensure the integrity of the code here presented,
* the author, the Optical Engineering Group and Loughborough University of Technology
* are not responsible for any damages that its use may cause.
*
*******************************************************************************/

long *diameter[3];

V4Ecode IpGrabEdge (char*fnameout, char*iname)
{
    return RunProc ("lpGrabEdge",fnameout, iname);
}

*******************************************************************************/

This function grabs an image (.tiff) from the video,
finds the first column with 2 or 4 edges using the Lut_function
finds the edges at first point using: Contrast (if 2 edges found)
finds the final edges using: the Derivative function.
draws the result on the top of the image and displays this image inside Visilog
finds the external and the internal diameters (and stores them in file 'diameter')
stores the data (edges) in a file (fnameout).
*******************************************************************************/
V4Ecode aGrabEdge (char*fnameout, char*iname)
{
  IMAGE img;
  int j, n, init;
  long xinit, x, xmin, xmax, ymin, ymax, initedge[4];
  long *edge[4];

  ProcName("aGrabEdge");

  img=GrabImage();
  if(img==NULL) {
    for (j=0; j<4; j++)
      free(edge[j]);
    return ABORT;
  }
  for (j=0; j<4; j++)
    edge[j]=(long*)malloc((img->gx/20+1)*sizeof(long));
  xmin=1;
  ymin=10;
  xmax=img->gx;
  ymax=img->gy-10;
  xinit=InitPoint2_4 (img, xmin, xmax, ymin, ymax, &init);
  if (xinit==0) {
    for (j=0; j<4; j++)
      free(edge[j]);
    return ABORT;
  }
  if (init==4)
    InitialEdges4 (img, initedge, xinit, ymin, ymax);
  else if (init==2)
    InitialEdges2 (img, initedge, xinit, ymin, ymax);
  else if (init!=2 && init!=4)
    return ABORT;
  for (j=0; j<4; j++) {
    if (j<1)
      edge[j][n=(int)((xinit-l)/20)]=NewEdge (img, xinit, initedge[j], initedge[j+1], j, j%2);
    if (j>=2)
      edge[j][n=(int)((xinit-l)/20)]=NewEdge (img, xinit, initedge[j], initedge[j-1], j, j%2);
  }
  for (x=xinit+20; x<xmax; x+=20) {
    if (j>=1)
      edge[j][n+1]=NewEdge (img, x, edge[j][n], edge[j+1][n], j, j%2);
    if (j>=2)
      edge[j][n+1]=NewEdge (img, x, edge[j][n], edge[j-1][n+1], j, j%2);
    n++;
  }
  n=(int)((xinit-1)/20);
  for (x=xinit-20; x>xmin; x-=20) {
    if (j>=1)
      edge[j][n-1]=NewEdge (img, x, edge[j][n], edge[j+1][n], j, j%2);
    if (j>=2)
      edge[j][n-1]=NewEdge (img, x, edge[j][n], edge[j-1][n-1], j, j%2);
  }
}
edge[j][n-1]=NewEdge (img, x, edge[j][n], edge[j-1][n+1], j, j%2);

} n--;

} 
)
DrawResult (img, edge, xinit-20*(xinit/20), xmax);
IpRead ("ImEdge", iname, V_FTYPE_AUTO);
for (j=0; j<3; j++)
diameter[j]=(long*)malloc((img->gx/20+1)*sizeof(long));
if (!Diameter (edge, xinit-20*(xinit/20), xmax, diameter)) {
    for (j=0; j<4; j++)
        free (edge[j]);
    return ABORT;
}
if (!StoreData (fnameout, edge, xinit-20*(xinit/20), xmax)) {
    for (j=0; j<4; j++)
        free(edge[j]);
    return ABORT;
}
for (j=0; j<4; j++)
    free(edge[j]);
return NOERROR;
Appendix A: Code listings

/**********************************************************************
* ANALYSIS OF A SERIES OF BLOOD VESSELS IMAGES VARYING IN TIME
* Unless otherwise stated this code has been written by:
* Almudena Eustaquio-Martin.
* Optical Engineering Group
* Loughborough University of Technology
* Loughborough (LE11 3TU)
* U.K.
* 1995.
* This section contains the three main functions necessary to build the menu option Dynamic Edge:
* IpTimeEdge, IpTimeMemr, IpTimeFree. Additionally, it contains some analysis functions.
* Although every effort has been made to ensure the integrity of the code here presented,
* the author, the Optical Engineering Group and Loughborough University of Technology
* are not responsible for any damages that its use may cause.
**********************************************************************/

#include "v4incl.h"
#include "ipe.h"
#include "time.h"

#define THS 85
#define INFINITY 3e38f

/**********************************************************************
* global variables *******************************************************
long **timediarnet[3];
time_t start;
**********************************************************************/

/**********************************************************************
* FRONT END CODE *********************************************************
V4Ecode IpTimeEdge(char*fnameout, char*iname, long img_num)
{
    return RunProc ("IpTimeEdge", fnameout, iname, img_num);
}
V4Ecode IpTimeMemr (long num_img)
{
    return RunProc ("IpTimeMemr", num_img);
}
V4Ecode IpTimeFree (long num_img)
{
    return RunProc ("IpTimeFree", num_img);
}
**********************************************************************/
Appendix A: Code listings

V4Ecode aTimeEdge (char*fnameout, char*iname, long img_num)
{

IMAGE img;
char *name;
int j, n, init;
long xinit, x, xmin, xmax, y, ymin, ymax, initedge[4];
long *edge[4];
FILE *file, *errfile;
time_t now;

ProcName("aTimeEdge");

if (img_num==1)
  time(&start);
time(&now);
img=GrabImage();
if (img==NULL) {
  for (j=0; j<4; j++)
    free(edge[j]);
  return ABORT;
}
for (j=0; j<4; j++)
  edge[j]=(long*)malloc((img->gx/20+1)*sizeof(long));
xmin=1;
ymin=10;
xmax=img->gx;
ymax=img->gy-10;
xinit=InitPoint2_4 (img, xmin, xmax, ymin, ymax, &init);
if (init==0) {
  for (j=0; j<4; j++)
    free(edge[j]);
  return ABORT;
}
if (init==4)
  InitialEdges4 (img, initedge, xinit, y, ymax);
else if (init==2)
  InitialEdges2 (img, initedge, xinit, y, ymax);
else if (init==2 & & init!=4)
  return ABORT;
for (j=0; j<4; j++)

97
if (j<=l) 
edge[j][n=(int)((xinit-1)/20)]=NewEdge (img, xinit, initedge[j], initedge[j+1], j, j%2);
if (j>=2) 
edge[j][n=(int)((xinit-1)/20)]=NewEdge (img, xinit, initedge[j], initedge[j-1], j, j%2);

for (x=xinit+20; x<xmax; x+=20) {
    for (j=0; j<4; j++) {
        if (j<=l) 
edge[j][n+1]=NewEdge (img, x, edge[j][n], edge[j+1][n], j, j%2);
        if (j>=2) 
edge[j][n+1]=NewEdge (img, x, edge[j][n], edge[j-1][n+1], j, j%2);
    }
    n++;
}

n=(int)((xinit-1)/20);
for (x=xinit-20; x>xmin; x-=20) {
    for (j=0; j<4; j++) {
        if (j<=l) 
edge[j][n-1]=NewEdge (img, x, edge[j][n], edge[j+1][n], j, j%2);
        if (j>=2) 
edge[j][n-1]=NewEdge (img, x, edge[j][n], edge[j-1][n-1], j, j%2);
    }
    n--;
}

DrawResult (img, edge, xinit-20*(xinit/20), xmax);
IpRead ("ImEdge", iname, _FTYPE_AUTO);
if (!TimeDiamet (fnameout, edge, xinit-20*(xinit/20), xmax, timediamet, img_num-1, now- start)) {
    for(j=0; j<4; j++) 
        free(edge[j]);
    return ABORT;
}
for (j=0; j<4; j++) 
    free(edge[j]);
return NOERROR;

/**************************** MAIN FUNCTION (IpTimeMemr) ****************************
*  *
* This function allocates the necessary memory to analyse the images with the function *
* IpTimeEdge
* ****************************

V4Ecode aTimeMemr(long num_img) {
    int i,j;
    FILE *file;
    ProcName("aTimeMemr");
    for (i=0; i<3; i++) {
        timediamet[i]=(long**)malloc((num_img)*sizeof(long*));
        }
for (j=0; j<num_img; j++)
    timediamet[i][j]=(long*)malloc((256/20+1)*sizeof(long));
}
return NOERROR;

/****************************** MAIN FUNCTION (lpTimeFree) **********************/

int aTimeFree(num_img)
{
    int i, j;
    FILE *file;
    ProcName("aTimeFree");
    for (i=0; i<3; i++) {
        for (j=0; j<num_img; j++)
            free(timediamet[i][j]);

    } return NOERROR;
}

/******************************* FUNCTION DEFINITIONS ******************************/

/**************** TimeDiamet *****************************/

int TimeDiamet (char*fnameout, long**edge, long xinit,long xmax, long***timediamet, long n, time_t real time)
{
    int x;
    long wall;
    FILE *file, *errfile;

    file=fopen (fnameout, "a");
    if (file==NULL) {
        errfile=fopen("errfile", "a");
        fprintf(errfile, "Can't open fileout");
        fclose(errfile);
        return 0;
    }

    ProcName("TimeDiamet");
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)

            free(timediamet[i][j]);

    return NOERROR;
}
if (n==0) {
    fprintf (file, "INTERNAL DIAMETER\n\nWALL THICKNESS\n\n");
    fprintf(file, "time(s.)x1x2x3x4x5x6x7x8x9x10x11x12x13\n");
    fprintf(file, "time(s.)x1x2x3x4x5x6x7x8x9x10x11x12x13\n");
}
    fprintf(file, "%ld\n", realtime);
    for (x=0; xinit+x*20<xmax; x++) {
        timediamet[0][n][x]=xinit+x*20;
        timediamet[1][n][x]=edge[3][x]-edge[0][x];
        timediamet[2][n][x]=edge[2][x]-edge[1][x];
        fprintf(file, "%ld\n", timediamet[2][n][x]);
    }
    fprintf(file, "\n");
    fprintf(file, "%ld\n", realtime);
    for (x=0; xinit+x*20<xmax; x++) {
        wall=timediamet[1][n][x]-timediamet[2][n][x];
        fprintf (file, "%ld\n", wall);
    }
    fprintf (file, "\n");
    fclose (file);
    return 1;
}
Appendix A: Code listings

/*******************************************************************************
* FUNCTIONS LIBRARY DECLARATION (ipe.h)
* Unless otherwise stated this code has been written by:
* Almudena Eustaquio-Martin.
* Optical Engineering Group
* Loughborough University of Technology
* Loughborough (LE11 3TU)
* U.K.
* 1995.
* This is a header file in which all the shared functions are declared.
* Although every effort has been made to ensure the integrity of the code here presented,
* the author, the Optical Engineering Group and Loughborough University of Technology
* are not responsible for any damages that its use may cause.
*******************************************************************************/

#define my_vessel

IMAGE Openlmage(char*imageFileName);
IMAGE Grablmage();
long InitPoint2_4(IMAGE img, long xmin, long xmax, long ymin, long ymax, int *init);
long *InitialEdges2(IMAGE img, long*edge, long x, long ymin, long ymax);
long *InitialEdges4(IMAGE img, long*edge, long x, long ymin, long ymax);
long NewEdge (IMAGE img,long x,long lasty,long lastyud, int e, int topEdge);
int StoreData(char*fileName, long**edge, long xinit, long xmax);
int Diameter (long**edge, long xinit, long xmax, long**diameter);
int TimeDiamet (char*fnarneotu, long**edge, long xinit, long xmax, long***timediamet, long n,
  time_t realtime);
int DrawResult(IMAGE img, long**edge, long xinit, long xmax);
float AverageX (IMAGE img,long x,long y);
float *Contrast (IMAGE img, long x, float *contrast);
float *Lut (IMAGE img, long x, float *Jut);
float *ContrastLut (IMAGE img, long x, float *contrastlut);
float Derivative (IMAGE img, long x, long y);
/* MENU FUNCTIONS 
* 
* Unless otherwise stated this code has been written by: 
* 
* Almudena Eustaquio-Martin. 
* Optical Engineering Group 
* Loughborough University of Technology 
* Loughborough (LE11 3TU) 
* U.K. 
* 
* 1995, 
* 
* These are the programs to build the menus in Visilog. 
* static_edge: is the menu for the program which analyses a single image stored in disk. 
* live_edge: is the menu for the program which analyses a single image from the video. 
* dynamic_edge: is the menu for the program which analyses a series of images form the video. 
* 
* Although every effort has been made to ensure the integrity of the code here presented, 
* the author, the Optical Engineering Group and Loughborough University of Technology 
* are not responsible for any damages that its use may cause. 
*******************************************************************************/

CMD static_edge 
{
    PAD=PA_card2;
    SIZE_ADJUST=[{PA_card2, 22, 0, 61, 60},{PA_cardGraphPad, 22, 60, 61, 38}];
    CHILD=[S_input, S_output, IM_output];
    CALL() {
        int xr, xw, yr, yw;
        int button;
        long diamet[2];

        WRmPopDown(W_image[0]);
        IpMyEdge(S_input, S_output, IM_output); 
        WRmPopUp (W_image[0]);
        Display("ImEdge");
        sprintf(message, "BEGIN\n");
        WinPrint(message);
        RmSetMouse(1, 20, 20);
        button=0;
        while (button!=3) {
            RmGetMouse(&xr, &yr, &xw, &yw, &button);
            if (button==1) {
                IpMyDiamet(xw, diamet);
                sprintf(message, "%d,%d\nD=%ld, d=%ld\n", xw, yw, diamet[0], diamet[1]);
                WinPrint(message);
                button=0;
            }
        }
        IpMyFree();
        sprintf(message, "%d,%d", xw, yw);
    }
}
CMD live_edge
{
  PAD=PA_card2;
  SIZE_ADJUST={PA_card2, 22, 0, 61, 60}, {PA_cardPad, 22, 60, 61, 38};
  CHILD={S_output, IM_output};
  CALL()
  {
    int xr, xw, yr, yw;
    int button;
    long diamet[2];

    WRmPopDown(W_image[0]);
    IpGrabEdge(S_output, IM_output);
    WRmPopUp (W_image[0]);
    Display("ImEdge");
    sprintf(message, "BEGIN\n");
    WinPrint(message);
    RmSetMouse(1, 20, 20);
    button=0;
    while (button!=3) {
      RmGetMouse(&xr, &yr, &xw, &yw, &button);
      if (button==1) {
        IpMyDiamet(xw, diamet);
        sprintf(message, \"%d, \%d\nD=%ld, d=%ld\n\", xw, yw, diamet[0], diamet[1]);
        WinPrint(message);
        button=0;
      }
    }
    IpMyFree();
    sprintf(message, \"%d, \%d\n\", xw, yw);
    sprintf(message, \"END\n\);
    WinPrint(message);
    return 1;
  };
};

CMD Static
{
  PAD=PA_card2;
  CHILD={S_output, IM_output, I_size};
  CALL()
  {
    IpTimeEdge(S_output, IM_output, I_size);
    return 1;
  };
};

CMD dinamic_edge
{
PAD=PA_card2;
CHILD={S_output, L_size {MSG = "Number of images"};};
CALL()
{
   auto I;
   
   EmptyImageList();
   IpTimeMemr(L_size);
   for (i = I; i <= L_size; i++)
   
      Static (S_output, i, i);
   
   IpTimeFree(L_size);
   return I;
};
MENU M_vessel
{
    LOC=[0,0];
    DIM=[100,100];
    PARENT=PA_card1;
    TYPE=TERMINAL_MENU;
    MSG="Vessel Edge";
    ENT={
        "Static Edge", static_edge,
        "Live Edge", live_edge,
        "Dinamic Edge", dinamic_edge,
    };
};
Appendix B

Dimensional analysis of blood vessel images in real time

In this appendix a paper reporting some of the work developed in this thesis is included. The paper was presented at the European Biomedical Optics conference celebrated in Barcelona (Spain) in September, 1995.
Dimensional Analysis of Blood Vessel Images in Real Time

Peter R. Smith, Almudena Eustaquio-Martin, Harry Thomason*, M. Bennett* and H. Thurston*

Department of Electronic and Electrical Engineering,
Department of Physical Education and Sports Science,
Loughborough University of Technology, Leicestershire LE11 3TU, UK.
*Department of Medicine & Therapeutics, Clinical Sciences Building,
Leicester Royal Infirmary, PO Box 65, Leicester, LE2 7LX, UK.

ABSTRACT

The physiology and pathology of dissected blood vessels are studied by perfusion myography combined with video microscopy. Images of the vessels are formed under diffuse white light illumination and contrast is achieved by differential absorption with respect to the vessel wall. To obtain the vessel dimensional information in quasi real time an edge-tracking algorithm is used, allowing the edges to be found by applying common image processing tools to a very small number of pixels rather than the whole image. Employing a low order optical model of the light transmission properties of vessels with circular cross section, a relationship between the positions of edges found by a typical image processing algorithm and actual dimensions is derived. The dimensional analysis is demonstrated on rat mesenteric resistance arteries (internal diameter <300μm) mounted in a perfusion arteriograph. Segments of vessels are secured on two glass cannulae using single strands of a nylon braided suture. The artery is perfused with physiological salt solution and the perfusion pressure maintained at 60mmHg before starting the experiment. Changes in vascular diameter to the vasoconstrictor noradrenaline and the endothelium-dependent vasodilator acetylcholine were then observed.

Keywords: Blood vessels, myography, video microscopy, edge detection, image processing, vessel diameter.

1. INTRODUCTION

Evidence is now accumulating which suggests that abnormalities of resistance arteries may contribute towards the elevated peripheral resistance seen in hypertension [1]. Until recently, information about the properties of resistance arteries had to be inferred from haemodynamic studies and from visual examination of vessels in vascular beds that were readily accessible. However, the development of a small artery myograph permitted studies of the structure and function of isolated sections of resistance arteries to be investigated for the first time [2]. In this technique arteries are stretched between two mounting wires in a horizontal plane, producing a non-circular conformation. Information concerning the structure of the mounted arteries can be obtained by direct measurement of wall thickness and lumen diameter by light microscopy. More recently a technique has been developed which allows pressurisation of cannulated resistance arteries, which allows the vessels to assume a conformation that resembles the in vivo condition [3]. Pressure myographs are clearly a more physiological arrangement than the wire myograph and are also less traumatic to the vessel segment.

In the pressure myograph changes in lumen diameter can be monitored directly on a video screen or measured electronically or optically. Images of the vessels are usually formed under diffuse white light illumination and contrast is achieved by differential absorption with respect to the vessel wall. The external and internal surfaces of the vessel would normally be detected by an edge detection algorithm, involving the computation of gradient and Laplacians combined with filtering operators depending on the texture of the
image features. Such a procedure would certainly find edges within the image and these edges will be functionally related to the vessel dimensions but the relationship may not always amount to a linear scaling factor. This communication explores the relationship between edges found by a typical image processing algorithm and actual dimensions, by using a low order optical model of the light transmission properties of vessels with circular cross section. A correction procedure is derived and tested on vessel images derived from perfused resistance arteries (internal diameter < 300 μm).

2. MODEL

The structure of blood vessel images formed by light microscopy of a perfusion myograph depend on the propagation of light in the particular tissue volume. A simple representation of the geometry of the vessel and imaging system is shown in Figure 1. Assuming the effective absorption in the vessel wall and lumen are governed by a one-dimensional Beer Lambert law, the transmission coefficient across the vessel can be approximated by

\[
\tau(x) = \tau_0 \exp\left[-\left(\gamma_A - \gamma_B\right) R \sqrt{a^2 - \left(x - x_0\right)^2} - \gamma_B R \sqrt{b^2 - \left(x - x_0\right)^2}\right]
\]

where absorbance in the lumen (\(\gamma_A\)) and wall (\(\gamma_B\)) are assumed constant, \(x\) labels the pixel value, \(x_0\) is the centre of the vessel image and \(R\) indicates the real part. For the sake of simplicity the consequences of optical scattering are incorporated into the effective absorbance.

Figure 1: Schematic of blood vessel and image formed by diffuse illumination.

Conventional edge detection in image analysis uses gradient and Laplacian operators but because of the functionality of equation (1) it will be convenient to examine extrema of the function

\[
f(x) = \frac{\partial \ln \tau(x)}{\partial x}
\]

which is simply

\[
f(x) = \frac{(\gamma_A - \gamma_B)(x - x_0)}{\sqrt{a^2 - \left(x - x_0\right)^2}} + \frac{\gamma_B(x - x_0)}{\sqrt{b^2 - \left(x - x_0\right)^2}}
\]
where the $\mathcal{R}$ have been dropped for convenience. Extrema of $f(x)$ denoted by $x_{\text{ext}}$ are given by

$$
(x_{\text{ext}} - x_0)^2 = \left\{ \frac{a^2 - b^2 F}{1 - F} \right\}, \quad F = \left[ \frac{a^2}{b^2} \left( \frac{\gamma_B - \gamma_A}{\gamma_B} \right) \right]^{1/3}.
$$

[4]

At first sight these extrema might be interpreted as the detected internal edge of the lumen, however although they strictly correspond to a maximum in the gradient this is only because the gradient changes sign and takes a small positive value inside the lumen. The maximum absolute gradient is always infinitesimally close to the actual lumen edge. Nevertheless given the mathematical significance it is worthwhile, in passing, to explore the consequences of equation (4) if interpreted literally. Estimating typical values of the absorbance for white light illumination of arteries in a saline bath is not straightforward as there exists little spectroscopic information and there is likely to be some variation for tissue type and physiological loading as well as for source wavelength. Some work has been performed on cadaveric aorta samples suggesting an absorption coefficient of $0.52 \text{cm}^{-1}$ at 633nm [4], compared with approximately $0.05 \text{cm}^{-1}$ for water, although the temperature would be a significant factor. These figures suggest a crude estimate of $-0.9$ for the differential absorption ratio in equation (4).

It is appropriate to investigate the relation (4) when the vessel wall is assumed to be incompressible; which is a good approximation for many physiological loadings [5]. Since the imaging system samples the vessel cross-section, in principle it is possible for the cross-sectional area to vary along the vessel length even though the tissue volume is a constant. However this is not considered here since the principles on which such changes might occur, if indeed they do so, are not established. Thus for vessels held in a perfusion myograph the lowest order approximation assumes the cross-sectional area $A$ of the wall is constant, leading to a simple relationship between external radius and lumen radius.

$$
b = \sqrt{\frac{A}{\pi} + a^2}.
$$

[5]

Using (5) in (4) yields a relation between measured and actual lumen diameter, as plotted in Figure 2.

![Figure 2: Relationship between measured and actual lumen diameter for incompressible vessel with cross sectional wall area 5000. Arbitrary units can be applied although for the images considered later, microns are appropriate.](image-url)
Clearly application of equation (4) will lead to an underestimate of the lumen diameter and the discrepancy is most acute as the contrast ratio increases. A complementary relationship exists for the wall thickness. The large differences between “measured” and actual lumen dimensions would only be found if an unwary programmer applied equation (4) or its inverse literally and we stress that this interpretation is only included for mathematical completeness. Later we describe the physical usefulness of this model when interpreting real data.

It may be noted that as an alternative to a parameter fit, recovery of the optical property of the vessel and lumen can be performed by inverting equation (1) at two suitable locations, together with an estimate of $\tau_0$ from the region of the image outside the vessel. The most obvious points to use are $\tau(x_0 \pm a)$ and $\tau(x_0)$ whence

$$\gamma_B = -\frac{1}{\sqrt{b^2 - a^2}} \ln\left(\frac{\tau(x_0 \pm a)}{\tau_0}\right)$$

$$\gamma_A = \gamma_B \left(1 - \frac{b}{a}\right) - \frac{1}{a} \ln\left(\frac{\tau(x_0)}{\tau_0}\right)$$

\[6\]

3. EXPERIMENTAL

The experiments were performed using 12-16 week old female Wistar-Kyoto rats bred in the Biomedical Services Unit at the University of Leicester. The rats were culled by stunning followed by cervical dislocation and the mesentery and feeding vasculature removed. Arterial resistance vessels were taken from the superior mesenteric bed which supplies the jejunum at a point 8-10cm from the pylorus. A 10-15mm long segment of the third branch of the mesenteric resistance artery was carefully dissected and cleaned of adhering adipose tissue under a dissection microscope and transferred to an arteriograph chamber filled with cold physiological salt solution (PSS). The arteriograph comprised a 30ml vessel chamber in which are housed two glass microcannulas used to perfuse the artery. In this arrangement the distal cannula is fixed whilst the proximal cannula is attached to an adjustable micrometer allowing it to be moved to the appropriate position. The proximal end of the artery segment was carefully slipped onto the proximal cannula and secured by means of a single strand (20µm) of a surgical braided nylon suture. Any residual blood in the artery was then gently flushed out. The distal end of the artery was then threaded onto the distal cannula and secured by the same means. The chamber was then washed several times with PSS and the axial length of the artery set by adjusting the micrometer. A polyethylene catheter was placed near to the afferent cannula and connected to a pressure transducer for measurement of perfusion pressure. The artery was then perfused with PSS and equilibrated under an optimal perfusion pressure of 60mmHg for 30 minutes before the experiment was started. The perfusate was supplied from a 500 ml reservoir bottle through a drip infusion tube connected to a afferent cannula and the perfusion pressure kept constant by leaving the reservoir at a defined height. The arteriograph was transferred to the stage of a light microscope with a TV camera attached to the viewing tube. The signal derived from the video image of the vessel was then analysed as described below. The arteriograph chamber was heated to 37°C and gassed with 95% O₂ - 5% CO₂ to achieve a pH of 7.4. Changes in lumen diameter to noradrenaline (10⁻⁶ mol/l) and acetylcholine (10⁻⁵ mol/l) were then recorded.

4. IMAGE ANALYSIS

Based on image processing techniques, an algorithm to find the external and internal diameters of a series of blood vessels in quasi-real time has been developed. The algorithm tracks the inner and outer edges of a blood vessel image from some initial starting edges.

The tracking method is based on the assumption that the position of the vessel's edges along two consecutively-examined columns, separated by a distance $\Delta x$ from each other, is approximately the same if $\Delta x$ is sufficiently small. That is the same as assuming that the vessel's profile varies smoothly along the length.
of the vessel. In order to explain the tracking algorithm it is enough to describe how it operates on the upper external edge of the vessel (the algorithm works similarly for the other three edges). If \( y_i \) is the y co-ordinate of the real upper external edge in the column \( x_i \), according to the above assumption, the co-ordinate \( y_i \) is the y co-ordinate of an approximate edge in the column \( x_{i+1} = x_i + \Delta x \). The y co-ordinate of the real upper external edge in the column \( x_{i+1} \) is defined at the point at which the gradient \([6]\) of the grey-level function its maximal within an interval \((y_i - \Delta y, y_i + \Delta y)\) around the approximate co-ordinate, \( y_i \). The same process is repeated along the whole length of the vessel incrementing and decrementing the x co-ordinate by the amount \( \Delta x \) each time.

Because of the way in which the tracking method operates, it is obvious that some real starting edges are required in order to begin the tracking operation. These starting edges are found using a threshold technique \([6,7]\). As a result of applying a threshold, an ordinary image becomes a binary one whose pixels can only have two possible values, namely "zero" and "one", making possible the separation between the background and the objects in the image. In a thresholded image, the changes in the intensity from "zero" to "one" and vice versa correspond to the edges of the objects in the image. In our case we are only dealing with images where the main objects are the walls of the blood vessels, although sometimes impurities such as air bubbles may also be present. To find the starting edges the algorithm distinguishes two kind of images: those in which the vessel's lumen is perfectly distinguishable from the vessel edges and those with a small inner diameter, in which the inner lumen is not distinguishable or hardly distinguishable from the vessel's edges. They will be referred to as standard vessels and narrow vessels respectively.

To find the starting edges in a blood vessel a threshold is applied column by column counting the number of intensity changes in each column until either two or four changes are found. The column keeping this condition is referred to as the starting column. If four changes are found, the vessel is regarded as a standard vessel. In this case, every one of the four changes corresponds necessarily to each one of the four vessel's edges, and its co-ordinates are taken as the approximated co-ordinates of the edges in the starting column (i.e. approximate starting edges). On the other hand, if only two changes in intensity are found, the vessel is regarded as a narrow one and every one of the two intensity changes corresponds necessarily to each one of the two vessel's external edges. The internal edges in a narrow vessel are found after applying a contrast enhancement \([8]\) to the pixels between the external edges in the starting column. A suitable threshold is then applied to the contrasted pixels in order to find the approximate internal edges. In either case, whether the vessel is a standard or a narrow one, the starting edges found in the way explained above are only approximated because they depend on the chosen threshold. The real starting edges are placed where the gradient of the grey-level function is maximal in an interval around the approximate starting edges.

Once the vessel's edges have been tracked, the internal and external diameters are calculated from the co-ordinates of the real edges of the vessel not only in a single column but along the whole of its length. One of the advantages of this tracking method is that it is computationally cheap, because for each examined column, only a small number of pixels around the edges are analysed. This allows the algorithm to be applied in quasi-real time to every image of a series in order to study the variations in time of the internal diameter of the vessel and its wall thickness.

Figure (3) shows the result obtained with the edge-tracking algorithm for a single blood vessel image. Notice how the algorithm successfully avoids the air bubbles present in the lumen visible between \( x = 200 \) and \( x = 250 \). Because of the way in which the edge-tracking method works, the edges of the impurities are always avoided as long as they do not touch the vessel walls. This is why, in this case, the bubbles' edges are not confused with those of the vessel when calculating the vessel diameters. On the contrary, the fat cluster which is touching the wall at approximately \( x = 200 \) can not be avoided and therefore the algorithm predicts that the vessel's wall at that particular location is thicker than it actually is.
Figure (4) shows the result obtained when the algorithm is applied to an image of a blood vessel which is in the process of contracting. Note that the contraction is non-uniform along the vessel length.

A qualitative examination of the image in Figure (4) would appear to indicate that the vessel wall expands with the contraction as would be necessary if the tissue is incompressible. We can test that proposition by assuming that the edge detection algorithm is accurate when the vessel is uncontracted, that is towards the left edge of the image. We also assume that the outer edge of the wall is accurately obtained. Figure (5) shows the values of the lumen diameter and the wall thickness obtained with the edge tracking method together with the corresponding values obtained by assuming the wall is incompressible. The latter values are obtained by taking the measured value of $b$, an estimate of the uncontracted vessel cross sectional area and applying equation (5) to deduce the value of $a$. 
Figure 4 (a): Image of a partially contracted blood vessel. (b): Numerical result of edge tracking algorithm.

For the majority of the vessel length, the measured and model estimates are in good agreement, given the errors attributable to incomplete dissection of the vessel itself. Perfect dissection of the vessel is not realistic given their size. There is a significant divergence between the measured and modelled values near pixel number 150, where the algorithm underestimates the lumen diameter and overestimates the wall thickness compared with the calculated value. This effect is not predicted by the model equation (1) and therefore some development is necessary to provide consistency with a locally incompressible tissue.
Figure 5: Comparing the measured and modelled dimensions of a partially contracted vessel.

A generalisation of the above model would require a two-dimensional diffusion model of the light propagation through the vessel and this will be addressed in future. A crude estimate of the consequence of such a generalisation can be seen by performing a low pass filter operation on the profile given by (1). Figure (6) shows the results of such a process along with an example of a real data profile.

Figure 6: Comparing the model, filtered model and real transmission profiles. The Beer Lambert model (thin line), filtered Beer Lambert model (thick line) and gradient, filtered Beer Lambert model (dotted line) are shown for a range of vessel profiles for $A=5000$, $\gamma_a=0$, $\gamma_b=0.027$ between $\alpha=17$, and $\alpha=84$ (five curves). A real profile (triangles) taken from the image shown in Figure 4 is also shown for comparison.
On the basis of the filtered profiles it is clear that an edge detection algorithm would determine a lumen edge somewhat inside the real edge. For the example shown here an error of ~5 pixels is observed. Though consistent with samples of data profiles, this value can only be justified by its plausibility and the full diffusion model is expected to reveal further interesting features. It is plausible because we expect light scattering to degrade the image quality but this effect will be dependent on the effective optical density and tissue geometry. Therefore it is likely that the diffusion process will affect the image integrity non-linearly across the image.

5. CONCLUDING REMARKS

The dimensional analysis of blood vessels using video microscopy has necessarily made use of edge detection algorithms to obtain estimates of the lumen diameter and vessel wall thickness. We have demonstrated how this procedure may lead to an inconsistency with respect to the well established principle of tissue incompressibility. The physical origin of the inconsistency is likely to be attributable to a scattering effect caused by diffuse illumination of the vessel lumen through the optically diffusing tissue wall. This paper has described a fast algorithm for the vessel dimensional analysis suitable for quasi real time implementation; an important consideration for morphological studies of blood vessels in vitro and in vivo. We have also demonstrated on a real vessel image how the edge detection algorithm is consistent with an overestimate of the wall thickness and corresponding underestimate of the lumen diameter, given the assumption of incompressibility. We have derived a correction factor to transform the measured edge into the position predicted by a filtered Beer Lambert model. Future statistical analysis is required to confirm the validity of these preliminary results across a range of vessels undergoing complete cycles of contraction and expansion. Such studies could be conclusive if performed on vessels with well defined outer edges free from attached tissue. Quantitative tests of the assumptions underlying the model can only be performed via a separate and definitive measurement of the vessel dimensions, which could be achieved by enhancing the image contrast between wall and lumen. A two-dimensional, numerical diffusion model of the light propagation through the model vessel should also provide significant progress in understanding and interpreting video microscopy of perfused blood vessels in vitro.

6. REFERENCES
