Computer Vision Center

Graduate School of Engineering
Universitat Autònoma de Barcelona

Final Year Project

Motion Analysis in Terms of Wireless Video Capsule Endoscopy

Author:
BSc. Markus Autengruber

Supervisor:
PhD. Fernando Vilariño

February 13, 2009

Abstract

El treball presentat suposa una visió general de la 'Endoscopia amb Càpsula de Vídeo Wireless' i la inspecció de sequències de contraccions intestinals amb les últimes tecnologies de visió per computador. Després de la observació preliminar dels fonaments mèdics requerits, la aplicació de visió per computador es presenta en aquestos termes. En essència, aquest treball proveïx una exhaustiva selecció, descripció i avaluació de cert conjunt de mètodes de processament d`imatges respecte a l`anàlisi de moviment, en el entorn de seqüències d`imatges preses amb una càpsula endoscópica. Finalment, es presenta una aplicació de software per configurar i emprar de forma ràpida i facil un entorn experimental.

El trabajo presentado supone una visión general de la 'Endoscopia con Cápsula de Video Wireless' y la inspección de secuencias de contracciones intestinales con las últimas tecnologías en visión por computador. Tras la observación preliminar de los fundamentos médicos requeridos, la aplicación de visión por computador se presenta en éstos términos. En sustancia, este trabajo provee una exhaustiva selección, descripción y evaluación de cierto conjunto de métodos de procesamiento de imágenes con respecto al análisis de movimiento, en el entorno de secuencias de imágenes tomadas con una cápsula endoscópica. Finalmente, se presenta una aplicación de software para configurar y usar de forma rápida y facil un entorno experimental.

The presented work yields a survey of Wireless Video Capsule Endoscopy and the examination of intestinal contraction sequences by the use of state-of-the-art computer vision technologies. After an introductory observation of required medical foundations, the application of computer vision within these terms is pointed out. In essence, this paper provides a comprehensive selection, description and evaluation of a certain set of image processing methods and algorithms with respect to the analysis of motion in an environment of capsule endoscopy image sequences. Finally, a software application for setting up a fast and easy to use experimental environment is presented.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wireless Video Capsule Endoscopy is an up-to-date medical field of research which focuses on the examination of the human gastrointestinal tract to provide physicians with a diagnostic tool for the detection and diagnosis of gastrointestinal diseases. To be brief, capsule endoscopy obtains a video sequence from the inside of the human gastrointestinal tract which can be later examined by a physician or trained medical expert. In essence, this is a very time-consuming and exhausting working task. For that reason, computer vision and artificial intelligence technologies are applied to improve and support this procedure. From the computer vision point of view, this implies the application of high-level image processing methods and algorithms for the establishment of relevant image features.

1.1 Providing a Set of Motion Analysis Tools

The presented work concentrates on the selection, description and evaluation of image processing tools for the estimation of motion in medical images obtained by Wireless Video Capsule Endoscopy. The main goal of this project has been to provide a theoretical description and evaluation of these tools as well as a software application for practical experimentation and visualisation. The main focus within the terms of capsule endoscopy has been the examination of intestinal contractions through the analysis of motion performed by the lumen and the intestinal wall. The provided set of image processing tools may be considered as a useful source of information when analysing this motion in a computer-supported way. Each tool description is completed by the proposal of optimal parameters which have

been obtained through extensive experimentation. For a clear understanding and a good manageability of this paper, the set of tools has been divided into three main categories: pre-processing, optical flow estimation and feature extraction.

1.2 Main Contributions

Basically, this project provides three main contributions to the appliance of computer vision within capsule endoscopy. First of all is the extensive testing of a phase-based optical flow algorithm including its Gabor filtering approach. During the project, the algorithm has been fully adapted for the application to intestinal contraction sequences.
A further contribution can be identified in the selection and evaluation of a certain set of image filtering methods for refining capsule endoscopy images before applying optical flow estimation algorithms. Especially the median filtering and the Laplacian of Gaussian approach for blob detection have been considered to be very useful.
The third main contribution of this work is represented by the development of a fast and easy to use visualisation application for motion analysis experimentation in terms of Wireless Video Capsule Endoscopy.

1.3 Material

All of the practical work for this project has been carried out on a portable Apple Macintosh computer, powered by a 1.42 GHz G4 PowerPC processor and 1 GB DDR SDRAM of main memory. The base operating system has been Mac OS X Tiger, software version 10.4.11. Programme code has been implemented with MATLAB 7.3.0 (R2006b) for Mac OS X. Due to the use of a Macintosh system and the lack of a corresponding version of the Intel Image Processing Library (IPL), previously implemented C++ code for optical flow estimation could not be used. For that reason, each considered method and algorithm had to be either already available for MATLAB or implemented newly. To provide a good co-operation of the source code with the graphical user interface, the software application has been implemented with the built-in MATLAB GUI editor. Furthermore, the source code of this project has been prepared to be ready for a later integration in the currently available software framework INTES SOFT 1.0.3 from the Medical Imaging Group

of the Computer Vision Center, Universitat Autònoma de Barcelona.
The considered medical image material has been provided by the hospital of Vall d'Hebron, Barcelona. Each of the images and sequences which are illustrated in this paper have been extracted from capsule endoscopy videos, captured by medical research personnel using the Wireless Video Capsule Endoscopy approach.
For testing the accuracy of the regarded motion estimation algorithms and for further comparison purposes, a ground truth video solution has been produced which has been captured using the built-in 2.0 mega-pixel camera device of a regular Sony Ericsson mobile phone.

1.4 How to Use This Paper

In essence, this paper is organised in three main parts. Starting with an introductory description of the medical framework, the first part of this paper is dedicated to a comprehensive explanation of the technological framework. In this spirit, chapter 2 illustrates the underlying medical area of application and describes the terms and definitions of Wireless Video Capsule Endoscopy. In addition, a brief illustration of the identification and classification of intestinal contractions is given as well as the mention of the intestinal contraction bouncing point to be of special interest. Chapter 3 explains and describes the set of image pre-processing methods and optical flow algorithms which is followed by an extensive evaluation and comparison on the basis of a synthetic ground truth solution and realistic intestinal contraction sequences. Chapter 4 is dedicated to the brief description of a simple feature extraction approach. Both chapters 3 and 4 are completed by explaining implementation information.
The second part of this paper is dedicated to the description of the graphical user interface application which has been implemented during this project. Chapter 5 explains the design and implementation process whereas a brief explanation of the developed source code is provided. Chapter 6 represents a detailed user manual and should be the main source of information belonging to graphical user interface issues.
The third and last part of this paper acts as a summary. Chapter 7 discusses the main contributions of the underlying work by pointing out the most important results of the theoretical and practical experimentation. This is summed up by final

conclusions. In the end, chapter 8 completes this work by presenting ideas for further experimentation with the presented set of tools and how it could be improved and extended.

Part I

Medical and Technological Framework

Chapter 2

Medical Foundations

When starting to work with medical images, it is of essential importance to understand the application area and basic terms and definitions of the underlying medical field. This chapter is dedicated to achieve the task of providing a basic introduction to the examination of the gastrointestinal tract, including the description of a modern capsule endoscopy approach. While medical image material is illustrated, the application of computer vision within this approach is pointed out. It has to be mentioned that the provided information is basic and that only essential terms and definitions are described.

2.1 Examination of the Gastrointestinal Tract

The examination of the human digestion is an important medical procedure for determining the cause of gastrointestinal symptoms such as abdominal pain, diarrhoea, bleeding or anaemia. Currently used methods for the exploration of the gastrointestinal tract are gastroscopy and colonoscopy. Those are generally performed by a physician who is manually moving a tube with a camera unit mounted to its top through the oesophagus or the anus respectively to reach the concerned area which is then examined at the time while the patient has to lay down calm for the duration of the routine. These procedures are complex and usually carried out in a clinical situation which means discomfort and stress for the patient. Wireless Video Capsule Endoscopy is committed to revise this procedure by proposing an modern endoscopy approach using a capsule-shaped camera device. [Ima08], [Vil06]

2.2 Wireless Video Capsule Endoscopy

Wireless Video Capsule Endoscopy (WVCE) is an up-to-date medical field of research which concentrates on the examination of the human gastrointestinal tract to provide physicians with a high standard diagnostic tool to detect and diagnose gastrointestinal diseases. As it has been mentioned before, the examination procedure is carried out by the use of an especially designed independently moving capsule (figure 2.1) rather than a tube which has to be moved manually by a physician. The capsule is swallowed by the patient and travels through the human body by following the digestion. Usually, this process takes from six to eight hours while the capsule is gathering information continuously. [Ima08]

Figure 2.1 – Picture of the PillCam SB (front) and the PillCam ESO (back) developed by Given Imaging.[1]

Figure 2.2 shows an illustrative visualisation of the PillCam SB which has been developed by Given Imaging, a medical imaging research facility. The PillCam SB measures around 11 mm by 26 mm and weighs not more than four grams. Regarding its inner life, it is equipped with state-of-the-art electronic devices. The PillCam SB is capable of capturing a video of around two frames per second over a period of time of up to eight hours. The video frames are limited to the size of 256-by-256 pixels. To get bright images, the area in front of the camera is illuminated by powerful light-emitting diodes (LEDs). Throughout the procedure, the captured video is constantly transmitted to a nearby computer terminal using a wireless infrared connection. [Ima08] In the following, the PillCam SB may be referred to as the capsule or the capture device.

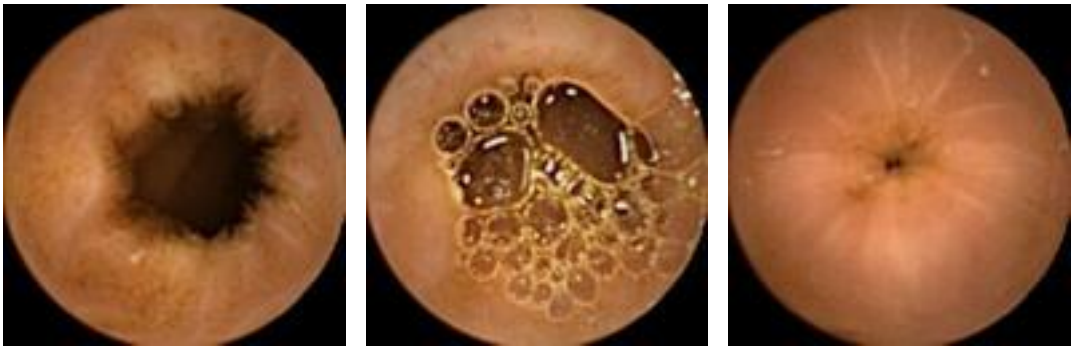
[1]Source: *http://www.vccafe.com/wp-content/uploads/2008/02/pillcam_large.jpg*, retrieved on January 27, 2009.

[2]Source: *http://www.fda.gov/cdrh/maturityhealthmatters/images/issue6-05.jpg*, retrieved on January 27, 2009.

Figure 2.2 – Illustrative visualisation of the PillCam SB.[2]

To provide an impression of the considered material, figure 2.3 shows example images which have been extracted from a random patient study. Looking at figure 2.3a, we can clearly identify the intestinal lumen as a dark area near the centre, surrounded by the intestinal wall. In this case, the capsule obtains a clear view on the scene. Figure 2.3b shows a similar situation in which the lumen is still visible but turbid liquid blurs the sight. Artefacts, such as turbid liquid and intestinal content, may narrow or completely blur the view on the lumen and influence the analysis of motion within a sequence. It has to be mentioned that during this project only images without the occurrence of such artefacts have been considered in order to focus completely on the lumen motion. Figure 2.3c illustrates how the lumen may disappear when the intestinal wall performs a contraction.

Once the whole video of the intestinal tract has been recorded, it needs to be examined in order to identify parts of special interest. For this reason, a physician or trained medical expert has to look through all of the video material to manually set flags wherever an interesting event occurs. Remembering the fact that these videos



(a) **(b)** **(c)**

Figure 2.3 – Capsule endoscopy images of the intestinal tract. (a) The lumen can be identified as a dark area near the centre, surrounded by the intestinal wall. (b) Turbid liquid is narrowing the view in the lumen. (c) The lumen may disappear when the intestinal wall performs a contraction

last from around six to eight hours, this is a very time-consuming and exhausting working task. To improve and support this procedure, computer vision and artificial intelligence technologies are applied to the video material in order to examine it automatically. Upcoming relevant events shall be identified by computer vision and classified by artificial intelligence technologies for a later inspection by a physician. [Vil06] For this project, the examination of an event identified as an intestinal contraction has been of major interest.

2.3 Identification, Examination and Classification of Intestinal Contractions

When digestive content moves along the intestinal tract it is forwarded by contractions of the intestinal wall. The way these contractions are performed describes the physical condition of the intestine. This is of important relevance for the detection and diagnosis of intestinal diseases. For this reason, the identification, examination and automatic classification of intestinal contractions is a crucial task in WVCE. [Vil06], [ISV+07], [SIV+08], [VSV+06]

The following figures provide an impression of the changeable appearance of intestinal contractions from the WVCE point of view. A series of representative examples has been picked out and illustrated. Each contraction consisting of nine subsequent

frames starts with the top left and ends with the bottom right image whilst considering one row after the other. Figure 2.4 shows a series of capsule endoscopy images which have been identified as an intestinal contraction by previous processing. Looking at the image sequence, it can be clearly identified that the lumen is positioned near the centre along almost all nine images and that it is totally closed at one time during the contraction is performed.

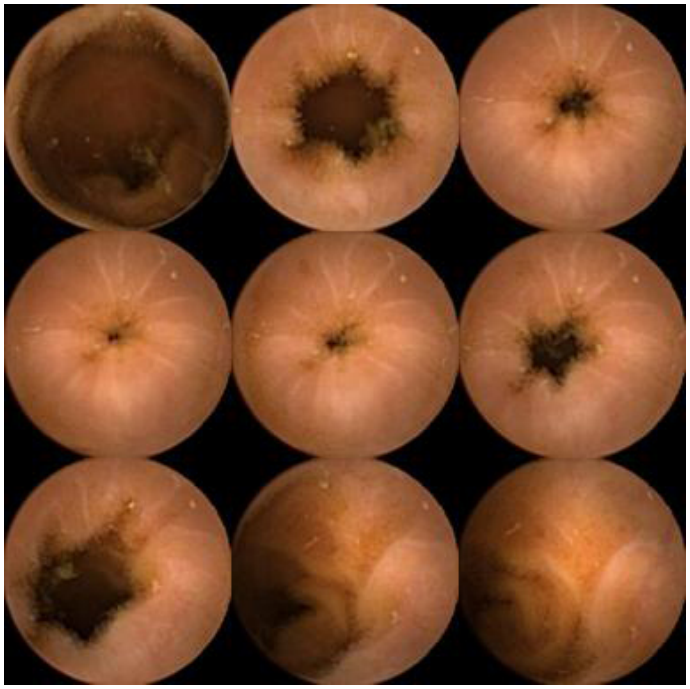


Figure 2.4 – Image sequence of an intestinal contraction. The lumen is completely closed at one time during the performance.

In comparison to that, figure 2.5 shows another image sequence of a correctly identified contraction whereas the lumen is never completely closed during the performance.

Furthermore, it can happen that the lumen is not visible as good as it has been in the previous examples. Figure 2.6 shows a contraction sequence where the lumen can not be clearly extracted from the intestinal wall along each of the nine frames. The previously illustrated circumstances describe only a few of several possibilities. As we are going to see later, the estimation of motion is of course seriously influenced by the varying appearance of intestinal contractions and the choice of optimal parameters for the considered methods and algorithms is a difficult working task.

For the objective of examining intestinal contractions, evaluation characteristics are

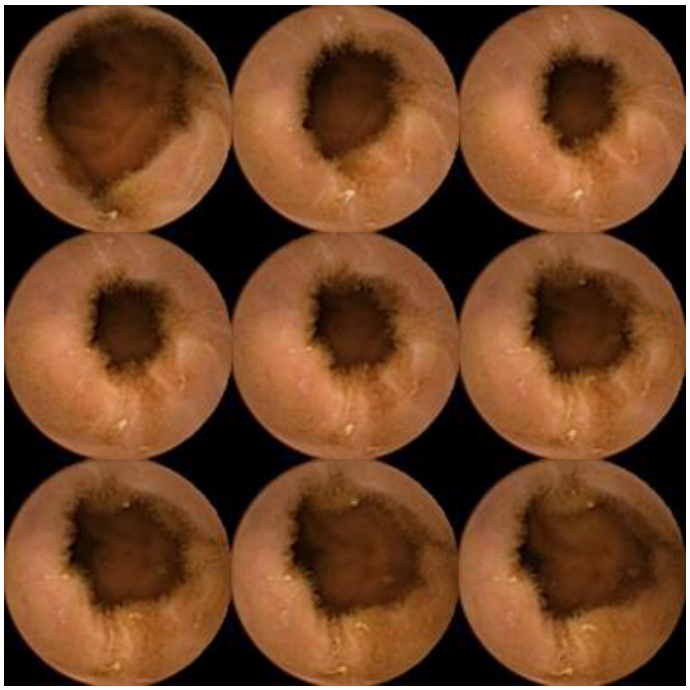


Figure 2.5 – Image sequence of an intestinal contraction. The lumen is never completely closed during the performance.

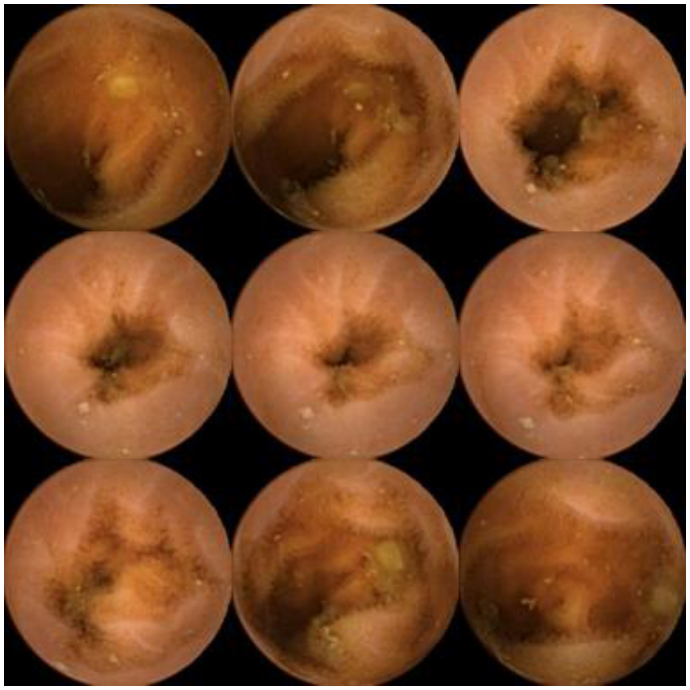


Figure 2.6 – Image sequence of a blurry intestinal contraction. The lumen can not be clearly extracted from the intestinal wall along each of the nine frames.

established in order to support the motility assessment of the intestinal tract [Vil06]. The following section mentions the location estimation of the intestinal contraction bouncing point which shall be considered as an example for such a characteristic.

2.4 Location Estimation of Intestinal Contraction Bouncing Points

Roughly, the bouncing point of a contraction is positioned in the centre of the intestinal lumen in the very exact moment when the contraction performance reaches its peak at this certain time. Its location may be used for examining the contractions performance. Figure 2.7 shows the image sequence from figure 2.4 with a manually marked bouncing point location (white plus). It has to be mentioned that the position of this point has not been estimated by a physician or trained medical expert. It has only been used as a ground truth solution for illustration and testing purposes during this project. The presented set of tools has been arranged to support a future approach for automatically estimating this location by considering motion information of the intestinal lumen.

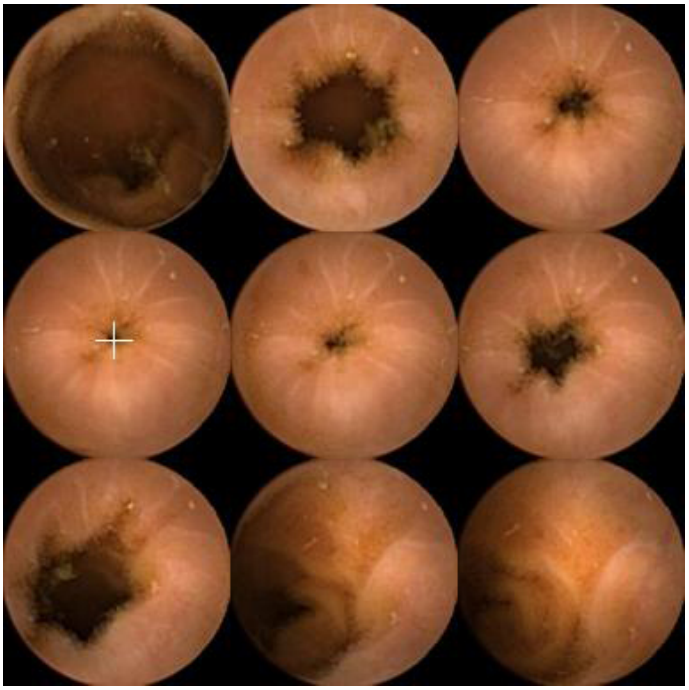


Figure 2.7 – Intestinal contraction sequence from figure 2.4 with manually marked bouncing point location (white plus). The position of this point has not been marked by a physician or trained medical expert. It has only been used as a ground truth solution for illustration and testing purposes during this project

Chapter 3

Motion Analysis

As it has been explained at the beginning of this paper, the main focus of this project has been set to the analysis of motion within intestinal contraction images obtained by WVCE. For this purpose, several computer vision strategies have been evaluated in order to provide a set of motion analysis tools which may be considered as a useful source of information when studying motion within capsule endoscopy image sequences. For a clear understanding and a good manageability of this chapter, the considered image processing procedures have been divided into three main categories: pre-processing, optical flow estimation and feature extraction. The selection and evaluation of appropriate pre-processing methods and optical flow estimation algorithms has been the most important part of this project and is described in this chapter. A simple feature extraction approach is presented in the following chapter. The following sections are dedicated to elaborated descriptions for each of the different tools, combined with illustrations of the corresponding results. The optical flow algorithms are applied to a self-produced ground truth solution and their differences are compared. In the end, representative experimental results on intestinal contraction sequences are discussed. This chapter is then completed by a brief presentation of the source code, which has been implemented for experimentation purposes.

3.1 Pre-Processing

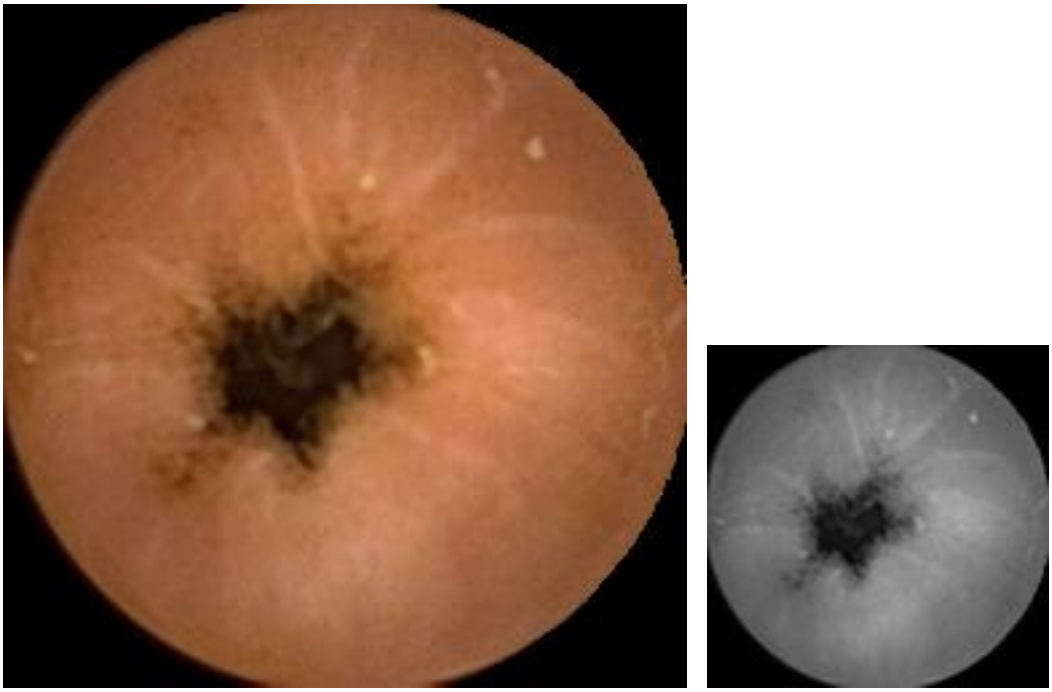
Pre-processing methods are applied to an image preceding to further image processing procedures. In general, this is done for the objective of reducing noise and to save expensive computation time. In this spirit, each image is initially downsized

from the original size of 256-by-256 to 128-by-128 pixels. Furthermore, each image is converted to greyscales. Both procedures are straightforward and do not need any further explanation. Instead, we want to focus now on more elaborated forms of pre-processing which shall prepare our images for the further application of high-level optical flow estimation algorithms.
In the following, several methods for pre-processing an image are described. Each of the methods has been evaluated during this project and considered to be relevant for the analysis of motion in capsule endoscopy images with respect to the further usage of optical flow algorithms. If a method is adjustable by the choice of one or more parameters, appropriate values are proposed which have been obtained by extensive empirical testing. Figure 3.1a shows an example image which has been extracted from an intestinal contraction. It may define the base image for the illustration of the results obtained by the proposed set of pre-processing methods. Figure 3.1b shows the same image after it has been initially re-sized to 128-by-128 pixels and converted to greyscales. Please take notice that both images are illustrated with their actual size. Looking at figure 3.1b, we can see that neither resizing nor conversion to greyscales did affect essential information of the original. Important artefacts, such as lumen and wrinkles of the intestinal wall, are still clearly visible whilst the amount of data has been successfully reduced and further high-level image processing should be less time-consuming and computational inexpensive.

3.1.1 Median Filtering

The median filtering method belongs to the category of rank filters.[3] It is performed by sorting the grey values of a certain M-by-N region around the regarded pixel and replacing the original grey value with the median which is considered to be representative for the whole region. This procedure is repeated recursively until every pixel of the original has been processed. The resulting image is a blurred representation of the original whereas noise has been reduced while edges have been preserved. As we are going to see later, this is very useful for the further application of an optical flow algorithm.
Median filtering can be adjusted by choosing the size of the considered region. The higher the values of M and N, the more blurred the result will become. Figure

[3]Source: *http://en.wikipedia.org/wiki/Median_filter*, retrieved on January 31, 2009.



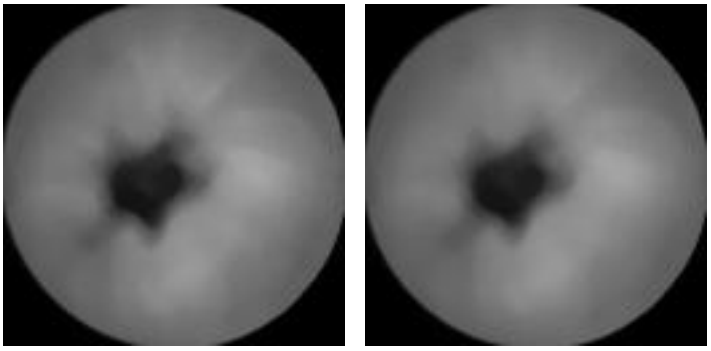
(a) **(b)**

Figure 3.1 – Base image for the evaluation of pre-processing methods and optical flow algorithms. (a) Original 256-by-256 capsule endoscopy image. (b) Re-sized 128-by-128 greyscale conversion. Please notice that both images are illustrated with their actual size.

3.2 illustrates the application of two different filter sizes to the image from figure 3.1b. Looking at figure 3.2a, the image is constantly blurred and noise, such as the small bright dots in the top right area of the image, has been removed while important edges, such as the border of the lumen and wrinkles of the intestinal wall, are still clearly visible. This result has been obtained by a 7-by-7 median filter. In comparison to that, figure 3.2b shows the result of a 9-by-9 filter region where noise has been reduced as well but wrinkles of the intestinal wall have been almost eliminated and the lumen border is starting to dissolve due to stronger blur effect. For that reason, the choice of a 7-by-7 median filter has been considered to be appropriate when working with 128-by-128 pixels capsule endoscopy images.

3.1.2 Logical Filtering

For further image processing, it has been useful to separate darker from brighter image regions. This may be performed by a simple logical filtering approach. Whilst we



(a) **(b)**

Figure 3.2 – Result of median filtering the image from figure 3.1b. (a) 7-by-7 median filter. (b) 9-by-9 median filter.

are already working with greyscales this procedure (equation 3.1) is straightforward.

$$value_{new} = \begin{cases} 1 & value_{old} \geq threshold \\ 0 & value_{old} < threshold \end{cases} \tag{3.1}$$

Every grey value is checked against a condition. If the value is higher than or equal to the threshold value, its new value is set to 1 (white pixel). If the value is lower than the threshold, the new value is set to 0 (black pixel). Figure 3.3 illustrates the application of the logical filtering method to the initial image from figure 3.1b. The resulting image is binary whereas the lumen has been extracted from the intestinal wall. The certain *threshold* of 60 has been obtained by empirical testing of several different values and intestinal contraction images. It has to be mentioned that the definition of a global threshold is not a convenient approach and may be revised by the following blob detection. Nevertheless, logical filtering is a fast and easy thresholding method and has been useful in different situations combined with other pre-processing methods.

3.1.3 Blob Detection

Whilst thresholding greyscale images is a rather simple approach for separating significantly different regions, blob detection is more sophisticated and appropriate for the separation of the lumen from the intestinal wall. Basically, blob detection examines an image for the existence of regions of either significantly dark or bright

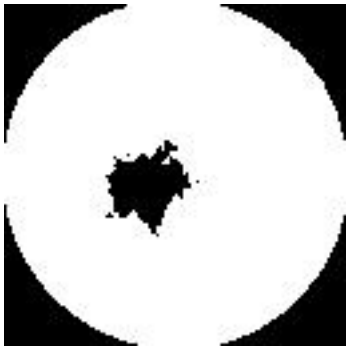


Figure 3.3 – Result of thresholding the image from figure 3.1b ($threshold = 60$).

pixels. This is usually performed by applying differential methods based on partial derivatives. For the objective of this project, we want to focus on a method which is based on the Laplacian of Gaussian (LoG). The LoG operator is widely used for blob detection nowadays.[4] In 2-dimensional problems, this is performed by applying the Laplace operator, or Laplacian, Δ to a 2-dimensional Gaussian kernel f. Equations 3.2 and 3.3 show the corresponding 2-dimensional Gaussian function $f(x, y)$ and the resulting function $g(x, y)$ after applying the Laplace operator.

$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.2}$$

$$g(x,y) = \Delta f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{3.3}$$

Looking at equation 3.3, the result is computed as the sum of the second partial derivatives of the 2-dimensional Gaussian function. This results in strong positive responses for dark and strong negative responses for bright blobs in the image which usually range around $\sqrt{2}$ for a 2-dimensional problem. The resulting LoG operator is then convoluted with the original and an edge detected representation of the image is the result.

This process can be adjusted by the choice of two parameters, the size s of the Gaussian filter kernel and its standard deviation σ. Figure 3.4 shows the application of the LoG operator to the image from figure 3.1b. Both parameters $s = 9$ and $\sigma = 0.35$ have been obtained through empirical testing. Looking at figure 3.4,

[4]Source: *http://en.wikipedia.org/wiki/Blob_detection*, retrieved on January 31, 2009.

strong edges, such as the border of the lumen and of course the border of the image, are pointed out clearly.

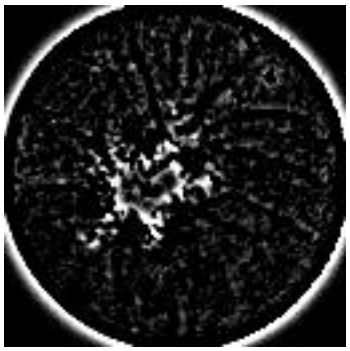


Figure 3.4 – Application of the LoG operator to the image from figure 3.1b (Gaussian filter kernel parameters: $s = 9, \sigma = 0.35$).

Furthermore, a combination of morphological procedures is carried out on the edge detected image provided by the LoG operator to extract the lumen. Figure 3.5 shows the image from figure 3.4 after morphological dilation with a flat 2-dimensional structure element.

Figure 3.5 – Morphological dilation on the image from figure 3.4 (parameters: disc element, radiusof 6 pixels).

The previously processed edges are pointed out by overlapping with disc elements of 6 pixels radius. The resulting image may then be improved by morphological closing in order to fill the areas between smaller blobs which are near to each other but not connected so far. The morphological closing procedure can be adjusted by the choice of the structure element and its size as well. With thresholding the image as an intermediate step, using the previously described logical filtering, for reducing noise surrounding the lumen, figure 3.6 shows the final result of this procedure. The

lumen has been successfully separated from the intestinal wall and has formed a blob whose movement may be estimated by an optical flow algorithm.

Figure 3.6 – Morphological closing on the image from figure 3.5 (parameters: disc element, radius of 6 pixels).

3.2 Optical Flow Estimation

After refining the images through the application of pre-processing methods, we want to concentrate now on the analysis of motion in intestinal contraction sequences by estimating the optical flow. Optical flow is a term of computer vision which has been formed in the last decades of image processing. In essence, it refers to the establishment of a motion vector field which describes the underlying motion of objects in a certain scene. Generally, there are several techniques for this objective including pixel comparison and signal processing algorithms.[5] [BA96], [ASM08], [BBPW04] In the following, two representative algorithms for both techniques are picked out, described and evaluated by the use of a ground truth solution and simple capsule endoscopy image sequences. After that, the application of the most promising algorithm to intestinal contraction sequences is discussed extensively.

3.2.1 Block-Matching Technique

Block-matching is a major pixel comparison technique which is widely used in video coding. The basic idea is to divide the concerned frame into macro-blocks and check these for similarities with corresponding macro-blocks of the subsequent frame of an image sequence. The supposition here is that macro-blocks belonging to a certain

[5]Source: *http://en.wikipedia.org/wiki/Optical_flow*, retrieved on January 31, 2009.

shape or object in the original frame will correspondingly move to form the same shape or object in the subsequent frame. It is straightforward to understand that only macro-blocks within a certain radius around the original block have to be observed in order to reduce computational effort. Figure 3.7 shows possible patterns for macro-block comparison.

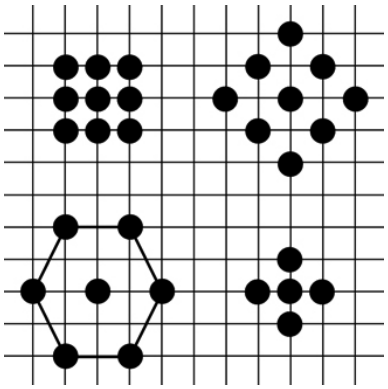


Figure 3.7 – Different block-matching search patterns including the exhaustive search (top left), the big diamond (top right), the hexagon (bottom left) and the small diamond (bottom right). [HWJJ06]

A certain cost function is used to compute the most suitable macro-block in order to finally obtain a motion vector from the original to the new position. Equation 3.4 illustrtes the Mean Absolute Difference (MAD) which is generally used as a macro-block matching criterion.

$$MAD = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |C_{ij} - R_{ij}| \tag{3.4}$$

Whereas C_{ij} refers to the pixels in the corresponding macro-block of the current frame and R_{ij} refers to the pixels from the corresponding macro-block of the reference frame. N declares the number of macro-blocks.

Ever since the computation of the cost function is a computational expensive task, its usage preferably needs to be avoided by the block-matching algorithm. In the last years, a number of different strategies has been developed. An explanation and evaluation of some of them can be found in [Bar04]. Most of the actual block-matching algorithms differ in the choice of the used search pattern. The choice of a good search pattern shall allow faster computations and a better motion vector field. The Variable Shape Search (VSS) algorithm [HWJJ06] has been considered

to form the base of further experimentation because of its use of variable search patterns and its availability for MATLAB. An open source version of the algorithm can be downloaded from the MATLAB File Exchange [Jal07].

Variable Shape Search (VSS) Algorithm

The Variable Shape Search (VSS) algorithm has been proposed by L. Hao et al. in 2006 at the University of Shanghai. In essence, it implements a block-matching strategy which uses combinations of both diamond and hexagonal search patterns for macro-block comparison. According to its authors, it can reduce computational complexity significantly and provides competitive computational speedups when compared to the Diamond Search (DS) algorithm which is generally used in block-matching and based on diamond search patterns only. Figure 3.8 illustrates the design of two basic search patterns.

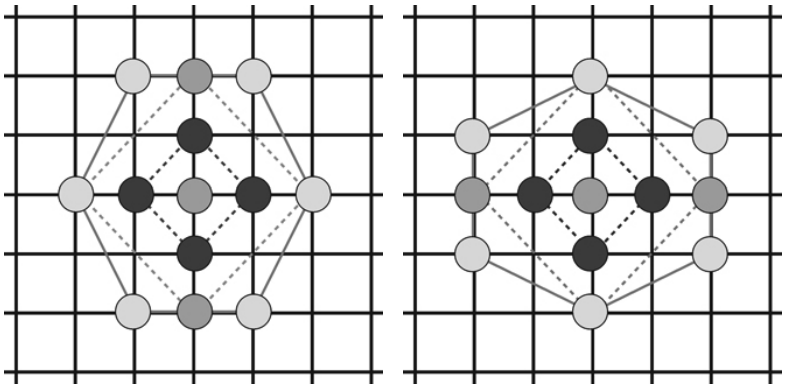


Figure 3.8 – Variable Shape Search (VSS) algorithm search patterns. (a) A combination of the big and the small diamond including a vertical hexagon. (b) A combination of the big and the small diamond including a horizontal hexagon. [HWJJ06]

For macro-block matching, the MAD (equation 3.4) is used. The following lines illustrate the function of the algorithm which basically includes four steps:

1. The big diamond pattern is checked. For each of its five points, the MAD of its macro-block is computed and compared to the according block from the subsequent image. The minimum MAD and its location are stored. If the minimum MAD is found at the centre of the big diamond, the algorithm proceeds directly to step 4, otherwise it continues by processing step 2.

2. At this stage, the algorithm needs to decide between the two basic search patterns, determining the direction of the following procedures. Therefore, it reconsiders the locations of the minimum MADs, found during step 1. If the minimum MAD has been found at a horizontal corner of the big diamond, search pattern 1 (figure 3.8a) is used and the points of the horizontal hexagon are checked. Otherwise, if the minimum MAD has been found at a vertical corner of the big diamond, search pattern number 2 (see figure 3.8b) is used.

3. The location of the minimum MAD, determined by the previous steps, is considered to be the centre point of a new search pattern. Whether the new pattern is using a vertical or horizontal shape is determined by step 2. Non-overlapping points of the new search patterns with the old ones are checked this time. If the minimum MAD is still found at the centre point, the algorithm proceeds to step 4. Otherwise, step 3 is repeated recursively whereas there is yet no restriction in the number of recursions.

4. Finally, the small diamond is checked. For each of its four points, the MAD is computed. If the minimum MAD found during this step is less than the one found by the previous steps, it is the new and final minimum MAD.

This procedure leads to a field of 2-dimensional motion vectors containing exactly one vector for each macro-block. It is straightforward to understand that the choice of the macro-block size determines the density of the final motion vector field and the accuracy of the algorithm. The computation of the motion vector field for a sequence of two images takes less than 1.5 seconds for images of 128-by-128 pixels. An evaluation of the obtained optical flow estimation is provided during the following comparison to the results of the signal processing approach.

3.2.2 Signal Processing

Instead of observing pixel data, such as colour and location information, signal processing methods establish the estimation of the optical flow by processing and analysing the image signal through various methods, such as differential techniques based on partial derivatives and methods based on the phase-response of the image signal. Generally, these methods provide smoother motion vector fields than block-matching techniques. However, the major drawbacks of signal processing methods

are a comparable great computational effort and their runtime.
Due to its usage of the promising Gabor filtering technique, the phase-based optical flow approach by T. Gautama and M. M. van Hulle [GvH02] has been considered to be the one to base further experiments on. Furthermore, its availability for MATLAB from the MATLAB File Exchange [Gau04] has been a major reason for this decision. In the following, a short introduction to Gabor filtering is provided, followed by the explanation of the phase-based optical flow approach.

Gabor Filtering

The Gabor filtering method belongs to the category of linear filters and is widely used in computer vision. [BNB04], [NMR92], [Mov08] Basically, it is represented by a harmonic base function which is multiplied by a Gaussian. Equation 3.5 illustrates a 2-dimensional Gabor function [PW08].

$$g_{\lambda,\theta,\psi,\sigma,\gamma}(x,y) = exp(-\frac{x^{\prime 2} + \gamma^2 y^{\prime 2}}{2\sigma^2})cos(2\pi\frac{x^{\prime}}{\lambda} + \psi) \tag{3.5}$$

Whereas $x^{\prime}$ and $y^{\prime}$ are defined as follows, $x^{\prime} = xcos\theta + ysin\theta$ and $y^{\prime} = -xsin\theta + ycos\theta$. The variables λ, θ, γ and ψ represent prime parameters of the Gabor function while σ belongs to the Gaussian. In fact, λ describes the wavelength of the cosine factor of the Gabor filter kernel and is specified in pixels. The variable θ refers to the orientation angle of the Gabor filter kernel and is specified in degrees. The ellipticity of the Gabor function may be adjusted by choosing γ and the value of σ describes the standard deviation of the Gaussian envelope. Finally, ψ describes the offset of the Gabor filter kernel. For the objective of this project, the offset has not been taken into account and is set to 0.
The Gabor filter kernel is convolved with the image in order to compute an edge detected representation. For the objective of reducing computational effort, usually a bank of pre-computed filter pairs with different scales (referring to λ and γ) and rotations (θ) is provided while each filter pair is applied to the image subsequently. This procedure results in a so-called Gabor-space, containing an edge detected representation for each of the filter pairs.[6] Figure 3.9 shows the application of a single Gabor filter to the base image from figure 3.1b. Detected edges have been pointed

[6]Source: *http://en.wikipedia.org/wiki/Gabor_filter*, retrieved on February 2, 2009.

out clearly (bright spots) on which basis the following phase-based algorithm is executed.

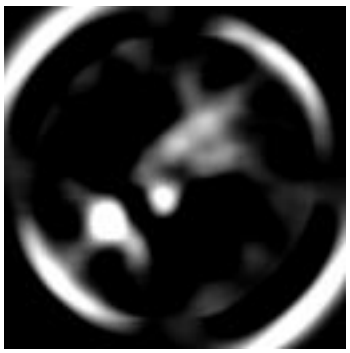


Figure 3.9 – Gabor filtered representation of the image from figure 3.1b. Edges have been pointed out clearly and can be identified as bright spots.

Phase-Based Approach

The phase-based optical flow approach based on spatial filtering [GvH02] has been introduced in 2002. According to its authors, it provides a reasonable fast and accurate technique for estimating the optical flow, based on the phase-information of the image signal. The considered images are initially processed by the previously described Gabor filtering method and the resulting contours are tracked by considering only the ones of which provide a constant phase-response over a certain period of time.
Basically, the algorithm runs through three main steps for a sequence of input images which are explained in the following:

1. Gabor filtering. Initially each frame of the sequence is spatially filtered using a bank of Gabor filter pairs. The algorithm uses pre-computed pairs of different scales and rotations in order to save computation time, as it has been mentioned before. However, the wavelength λ and the bandwidth b, which is a relation of λ and σ (equation 3.6 [PW08]), of the filter pairs can still be adjusted and represent essential parameters for the accuracy of the optical flow estimation. For good results, both parameters need to be adjusted to the current image size.

$$b = log_2 \frac{\frac{\sigma}{\lambda}\pi + \sqrt{\frac{ln2}{2}}}{\frac{\sigma}{\lambda}\pi - \sqrt{\frac{ln2}{2}}} \tag{3.6}$$

Furthermore, the filtering procedure is carried out by performing cascaded 1-dimensional convolutions rather than 2-dimensional which would be computational more expensive and time-consuming. The length l of the used 1-dimensional filter template is of essential importance as well. The correct choice of wavelength, bandwidth and template length is explained later during the discussion of the experimental results. After the application of the Gabor filter pairs at each spatial location, the component velocity of each phase gradient is derived and all of them are combined to form a set of velocity components which represent a first estimation of the full velocity at this location. Each velocity component direction is orthogonal to the orientation of the corresponding filter pair.

2. Linearity criterion. The linearity criterion is defined by a certain treshold. If a component velocity is not linear over a given time span (i.e. not reliable) it is rejected and not taken into account when computing the full velocity during the next step.

3. Recurrent neural network. At this point, each spatial location yields a set of reliable velocity components. The algorithm proceeds by computing the full velocity out of the set of components. According to the authors of the algorithm, every component velocity constrains the full velocity to lie on a so-called constraint line in a hypothetical velocity space. As there are several component velocities at a single spatial location, the intersection of the corresponding constraint lines is supposed to yield the full velocity of this location. Ever since the component velocities are effected by noise, they do not intersect in a single point. For achieving the most suitable intersection location, referring to the optimal full velocity constraint line, a goal programming strategy is performed which strives towards all goals of the problem simultaneously by minimising the sum of all orthogonal distances between each constraint line and the full velocity. This is achieved by the application of a recurrent neural network.

This approach leads to a smooth motion vector field whereas its density can be customised by choosing gx, the number of evaluation points along the x-axis of the considered frames. Furthermore, the linearity threshold $thres\_lin$ and the minimum number of valid component velocities nc_min can be adjusted in order to adapt the algorithm to the current problem. Optical flow estimation results on the basis of both synthetic ground truth and realistic capsule endoscopy image sequences are illustrated in the following. Moreover, it has to be mentioned that the computation of a sequence of two frames (i.e. one optical flow estimation) takes from 3 to 5 seconds on the basis of 128-by-128 pixels images which is around three times the runtime of the block-maching technique.

3.2.3 Evaluation and Comparison

For initial testing and evaluation purposes, a ground truth video solution has been established, focusing a black spot on a whiteboard while the camera is moving along two axes. The video frames have been limited to the size of 128-by-128 pixels and provide good contrasts combined with a simple object motion which has been suitable to initially evaluate the accuracy of the optical flow algorithms. Furthermore, the ground truth solution provides a good environment to adapt the parameters of both algorithms to the current problem. The ground truth video has been initially median filtered in order to minimise compression noise resulting from the capture device.

Figure 3.10 illustrates subsequent frames of the ground truth video which have been overlaid with the optical flow estimation of the VSS block-matching algorithm. A macro-block size of 8 pixels has been considered to be appropriate for 128-by-128 pixels images while trying to find a compromise between motion vector field density, accuracy and computational effort. In comparison, figure 3.11 shows the same sequence after it has been overlaid with the optical flow estimation obtained by the phase-based algorithm. It has to be mentioned that in this phase of the project the adaptation of the included Gabor filtering has not been taken into account and the algorithm has only been configured by setting $gx = 16$, $thres\_lin = 0.025$ and $nc\_min = 9$ which are proposed to be reasonable parameters. Please take notice that the black spot is moving up and around the corner to the right during the sequence.

Looking at figure 3.10, the optical flow estimation is not smooth and contains several

arbitrarily pointing motion vectors. These mostly result from the straight movement of the black spot where the block-matching criterion fails to determine the optimal macro-block. In comparison to that, figure 3.11 shows the much better result obtained by the phase-based approach. The motion vector field is smooth and describes the movement of the black spot correctly. However, the motion vector field does not exactly fit the area of the motion. It appears enlarged around the main area of the motion. Furthermore, noise data can be clearly identified in regions where there is no motion at all. Due to the fact that this could not result from compression noise, this has been the first evidence of a necessarily need of adaptation of the included Gabor filtering approach.

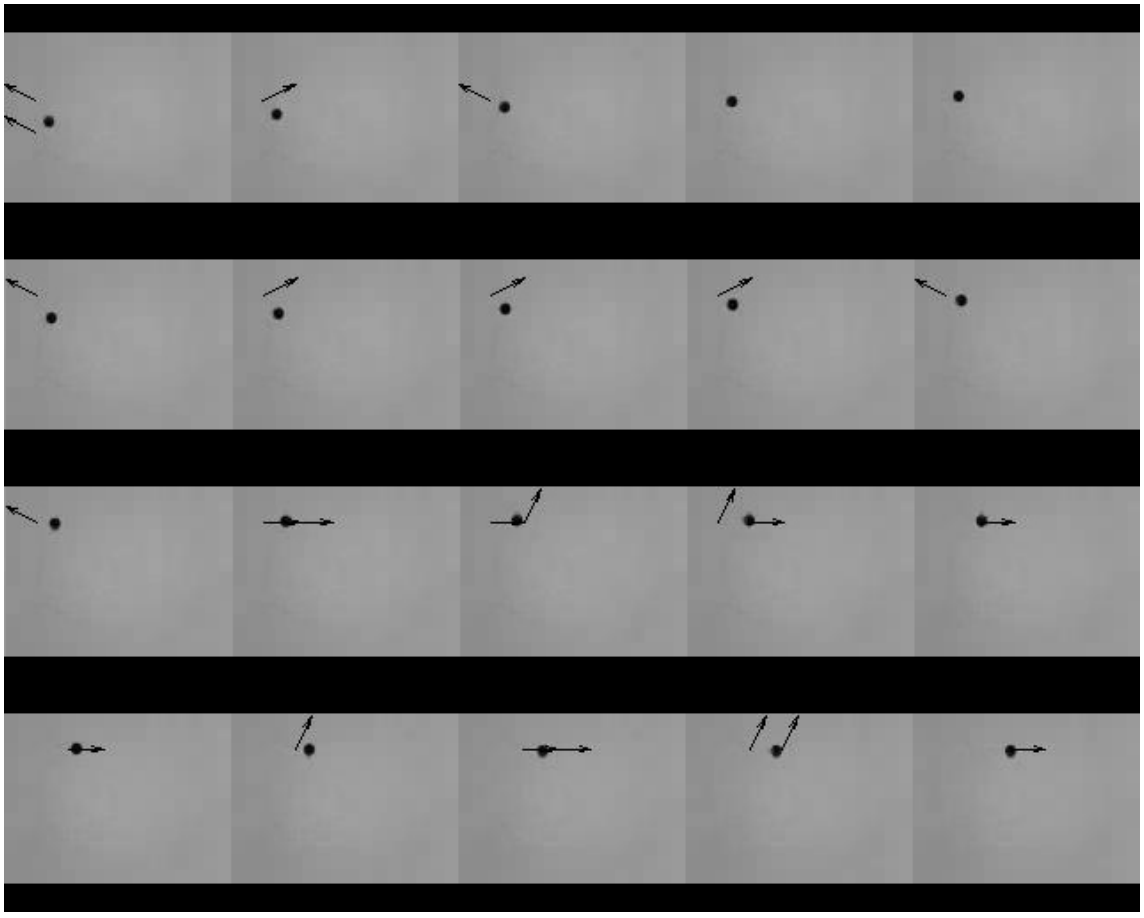


Figure 3.10 – Ground truth optical flow estimation obtained by the VSS algorithm. Sequence starts with top left and ends with bottom right image. The black spot is moving up and around the corner to the right.

After that, both algorithms have been applied to simple capsule endoscopy sequences

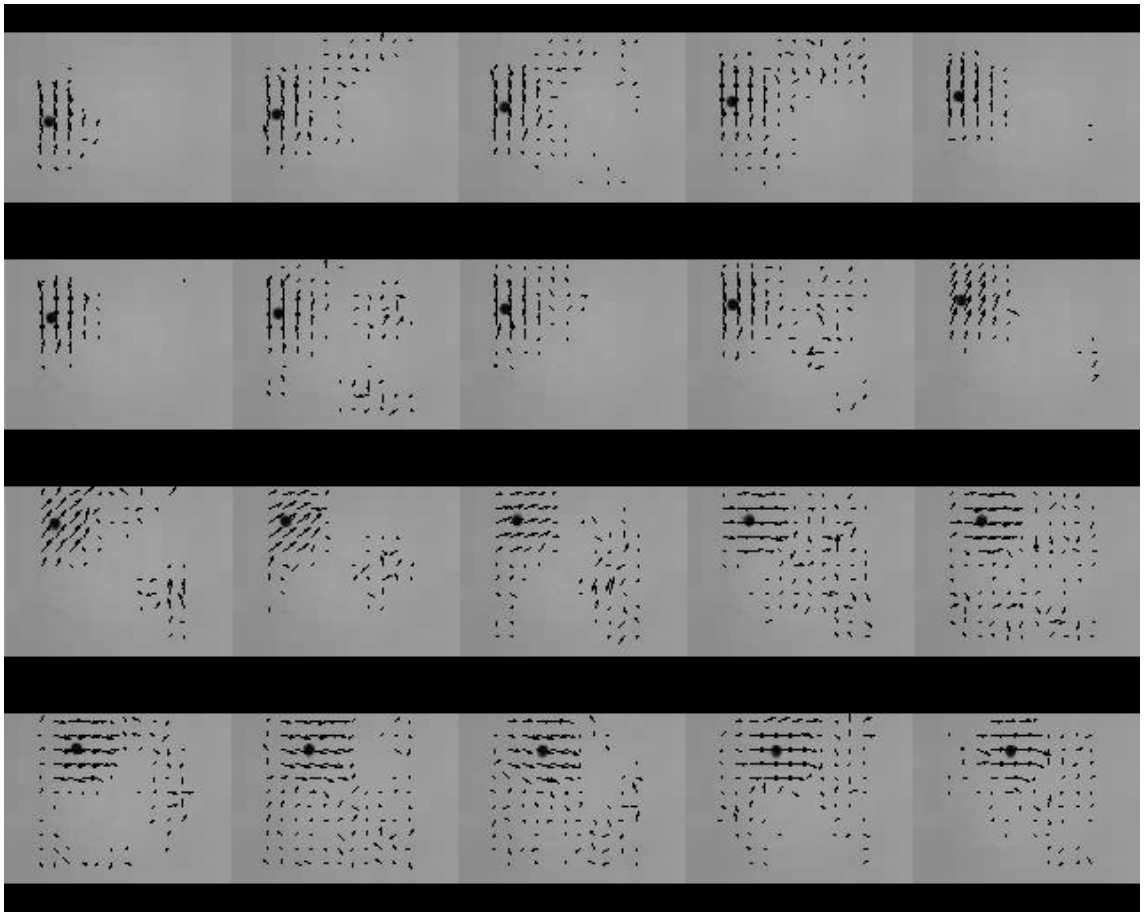


Figure 3.11 – Ground truth optical flow estimation obtained by the phase-based algorithm. Sequence starts with top left and ends with bottom right image. The black spot is moving up and around the corner to the right.

where the movement of lumen and intestinal wall has been clearly identifiable. Figures 3.12 and 3.13 show the corresponding results of both algorithms with respect to the previously proposed parameters. As expected, the VSS block-matching algorithm shows an unsatisfactory result consisting of arbitrarily seeming motion vectors whereas the phase-based approach shows a combination of accuracy and outliers. At this stage, further work on the VSS algorithm has been considered to be unnecessary for the objective of this project because the block-matching technique could not be evaluated as appropriate for the approach of studying motion in realistic capsule endoscopy images. The focus has been set to the further improvement of the phase-based approach.

Through the optimisation of the phase-based algorithm for capsule endoscopy images and extensively testing and adapting the Gabor filter parameters, the optical flow

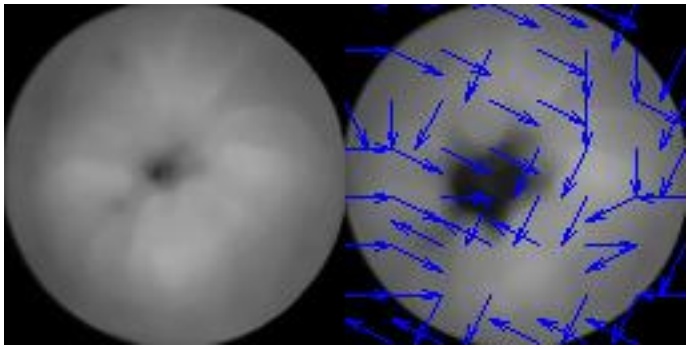


Figure 3.12 – WVCE optical flow estimation obtained by the VSS algorithm.

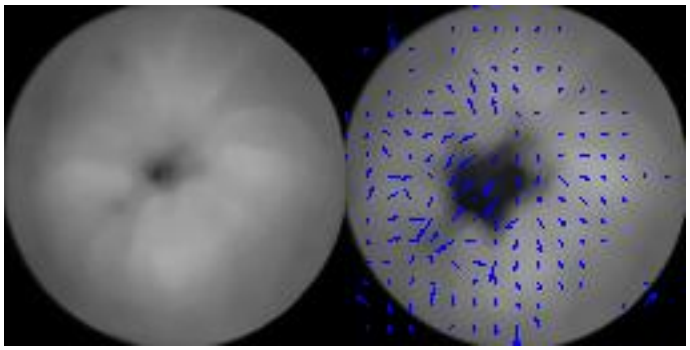


Figure 3.13 – WVCE optical flow estimation obtained by the phase-based algorithm.

estimation could be improved to produce expected results. Figure 3.14 illustrates the final result with regard to appropriate parameters: wavelength $\lambda = 4$, bandwidth $b = 0.5$, template length $l = 1$.

3.3 Experimental Results

In the following, the application of the phase-based optical flow algorithm to several intestinal contraction sequences is illustrated and the effect of different pre-processing methods is discussed. Each optical flow estimation has been carried out on initially re-sized 128-by-128 pixels greyscale converted frames.
Figure 3.15 shows the contraction sequence from figure 2.7 after it has been overlaid with the corresponding optical flow estimation. For this experimentation, no further improvement of the images by pre-processing methods has been performed. Only the Gabor filtering method, which is of course included in the phase-based algorithm, is used. The resulting sequence shows a good motion estimation for both

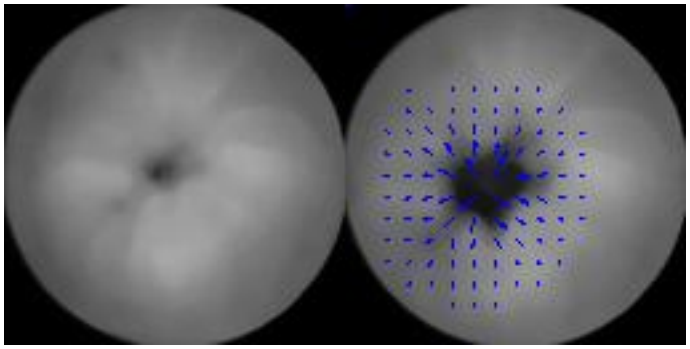


Figure 3.14 – WVCE optical flow estimation obtained by the optimised phase-based algorithm ($\lambda = 4$, $b = 0.5$, $l = 1$).

the lumen and the intestinal wall. Even in single sequences where the lumen performs a greater change of its size (e.g. between the first and the second frame) the algorithm delivers a convenient motion vector field. As it has been explained before, this follows from the fine-tuning of the Gabor filtering wavelength and bandwidth parameters. In comparison to that, figure 3.16 shows the optical flow estimation whereas the frames have been initially processed by a 7-by-7 median filter. This results in almost the same motion vector field but on closer inspection we can see that the length of the arrows has changed. This implies that the magnitudes of the motion vectors have been influenced by the median filter resulting in diminishing the total motion vector field and reducing noise. Although, prime directions seem to be left almost unchanged.

Figures 3.17 and 3.18 show the same contraction sequence from figure 2.7 whereas the LoG operator blob detection approach has been initially applied to the frames. Looking at figure 3.17, the motion vector field has been reduced to the area of the lumen. Apparently, this follows from the application of the blob detection approach which converts the frames to the binary representation, shown in figure 3.18. The resulting motion vector field illustrates that the movement of the lumen is lost when there is a greater change of its size between single frames. This is a matter of parameter choice (i.e. the size and the shape of the structure element) and may be improved by dynamically estimating these parameters for each single sequence. Moreover, the area of motion vectors around the lumen has been enlarged which results from the morphological dilation procedure as well. Nevertheless, the blob detection approach may be used to extract the single movement of the lumen from the movement of the intestinal wall in order to observe both motions separately.

Figures 3.19 and 3.20 illustrate again the same contraction sequence whereas a combination of median filtering and blob detection has been applied initially. Compared to figure 3.17, figure 3.19 shows again the reduction of noise resulting from the median filtering. The final motion vector field has been reduced even more and limited to the movement of the lumen, which can be clearly identified from the middle frames of the sequence.
In comparison, figure 3.21 shows the application of the algorithm to the contraction sequence from figure 2.5 where the lumen is not closed completely during the contraction is performed. The frames have not been improved by further pre-processing to illustrate the general optical flow estimation. The resulting sequence shows furthermore a good motion estimation whereas noise occurs inside the intestinal lumen. The same sequence has been processed by initially applying median filtering and blob detection with the same parameters as before (figure 3.22). The results shows that the previously estimated blob detection parameters could not extract the lumen from the intestinal wall correctly in this case.
Finally, figure 3.23 shows an optical flow estimation for the contraction sequence from figure 2.6 where the lumen cannot be clearly extracted from the intestinal wall. The resulting motion vector field is clearly distracted. Median filtering and blob detection could not significantly improve the result (figure 3.24) whereas the same parameters for blob detection have been used as before. This shows once again that the choice of global parameters may not be a convenient approach.

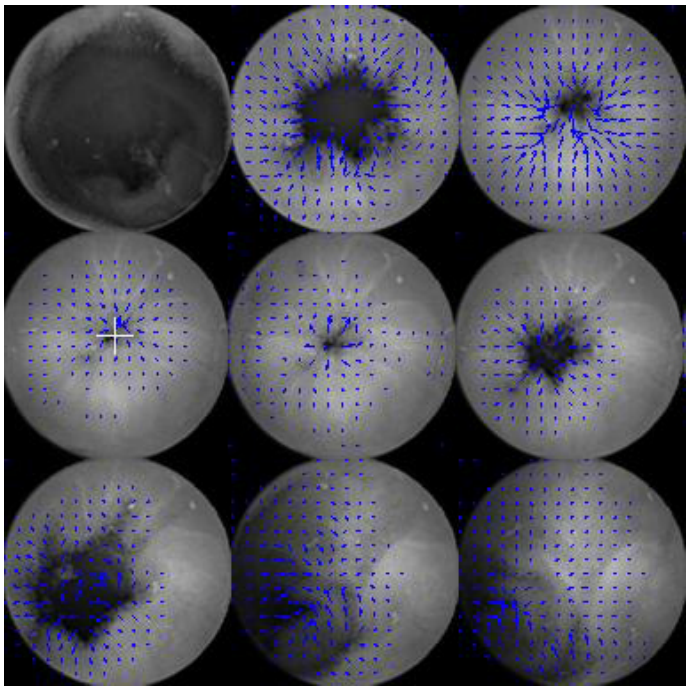


Figure 3.15 – Intestinal contraction sequence from figure 2.7 after estimating the optical flow. No further pre-processing has been performed.

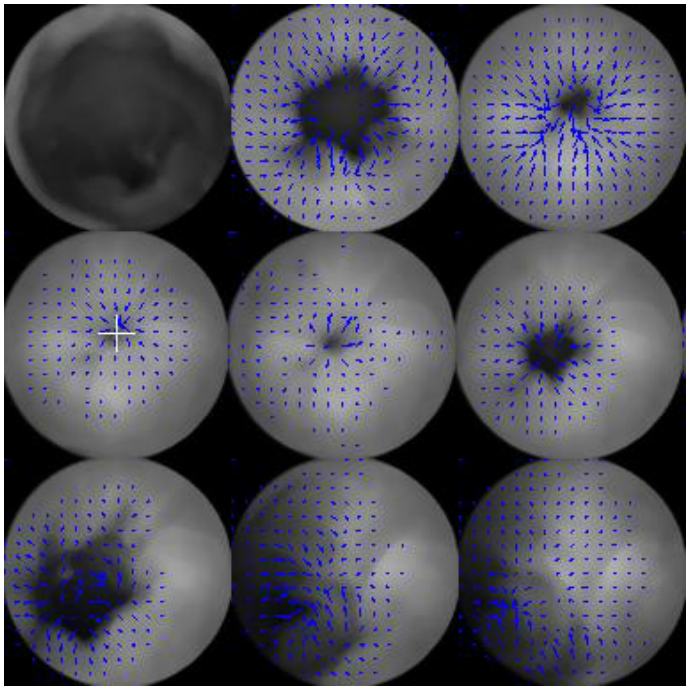


Figure 3.16 – Optical flow estimation from figure 3.15 after initially applying a median filter.

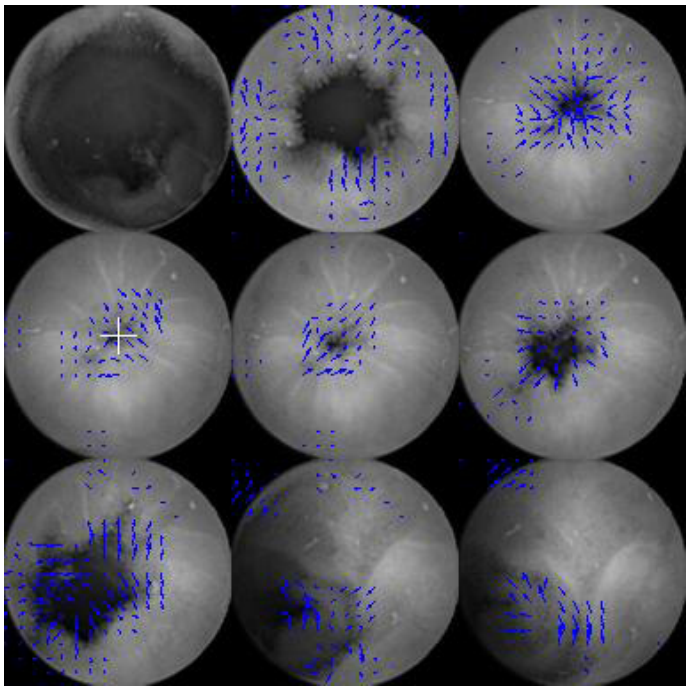


Figure 3.17 – Optical flow estimation from figure 3.15 after initially applying the blob detection approach. Note: original frames are illustrated.

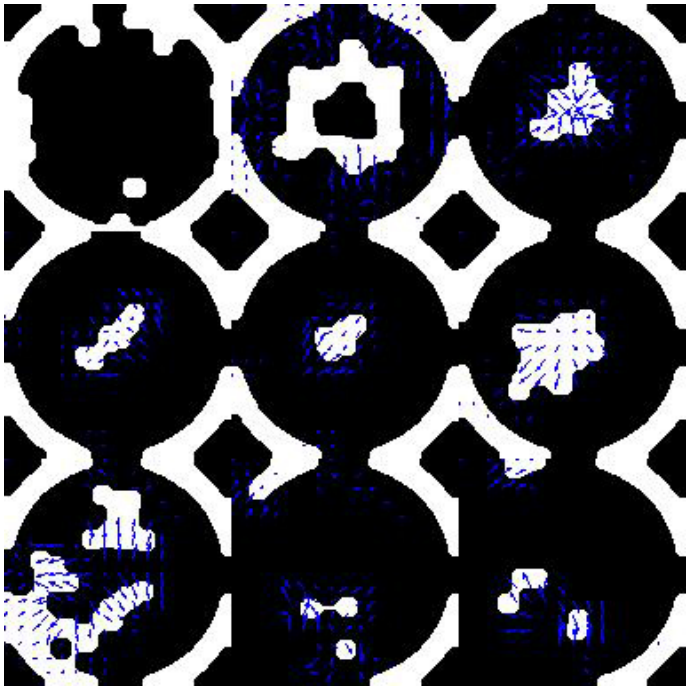


Figure 3.18 – Optical flow estimation from figure 3.15 after initially applying the blob detection approach. Note: blob detected frames are illustrated.

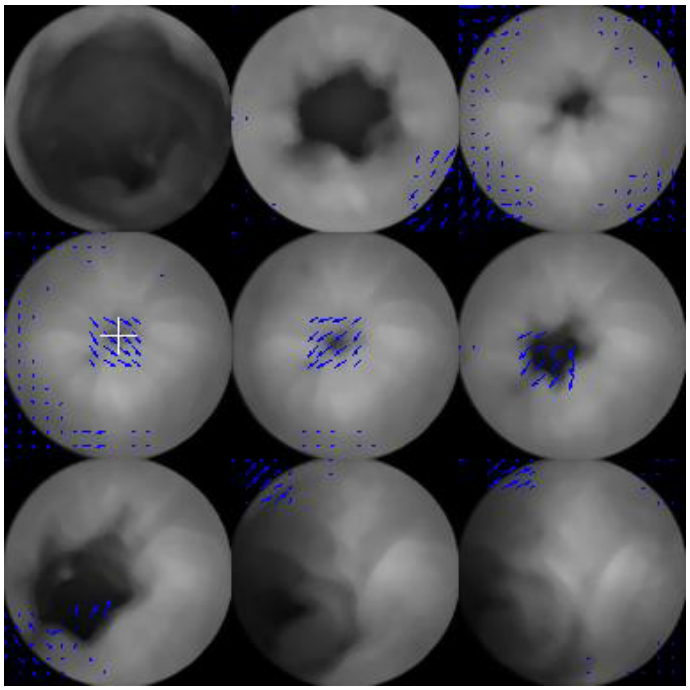


Figure 3.19 – Optical flow estimation from figure 3.15 after applying a median filter and the blob detection approach. Note: median filtered frames are illustrated.

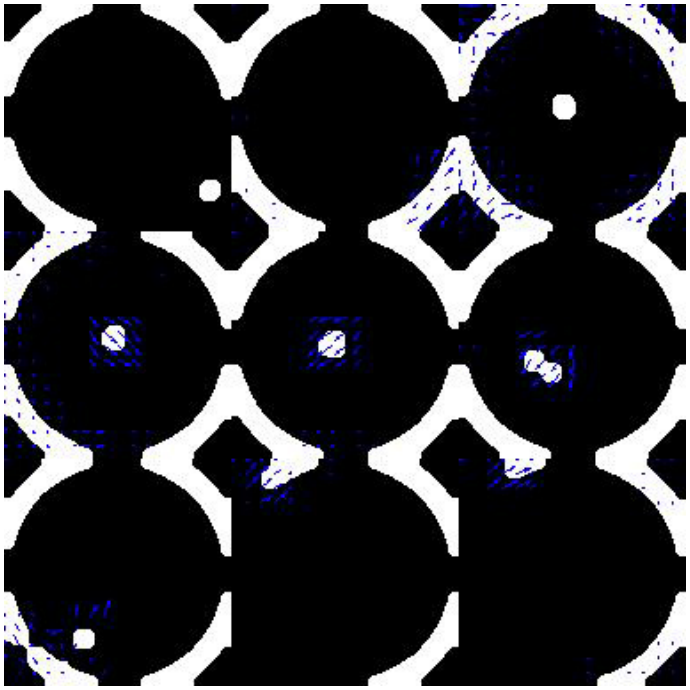


Figure 3.20 – Optical flow estimation from figure 3.15 after applying a median filter and the blob detection approach. Note: blob detected frames are illustrated.

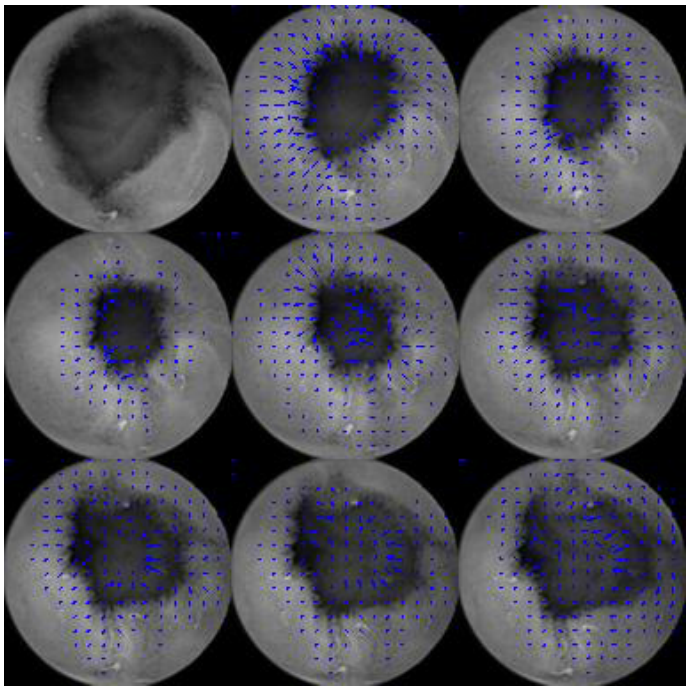


Figure 3.21 – Intestinal contraction sequence from figure 2.5 after estimating the optical flow.

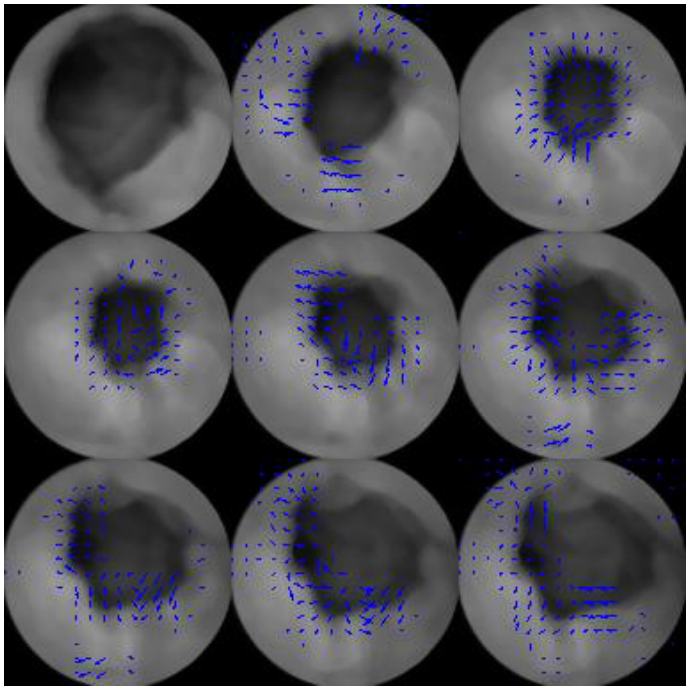


Figure 3.22 – Optical flow estimation from figure 3.21 after applying a median filter and the blob detection approach. Note: median filtered frames are illustrated.

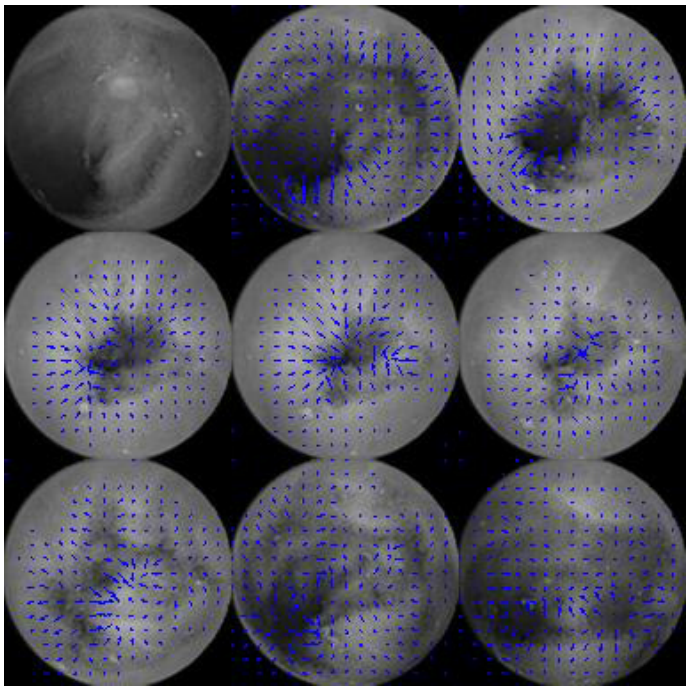


Figure 3.23 – Intestinal contraction sequence from figure 2.6 after estimating the optical flow.

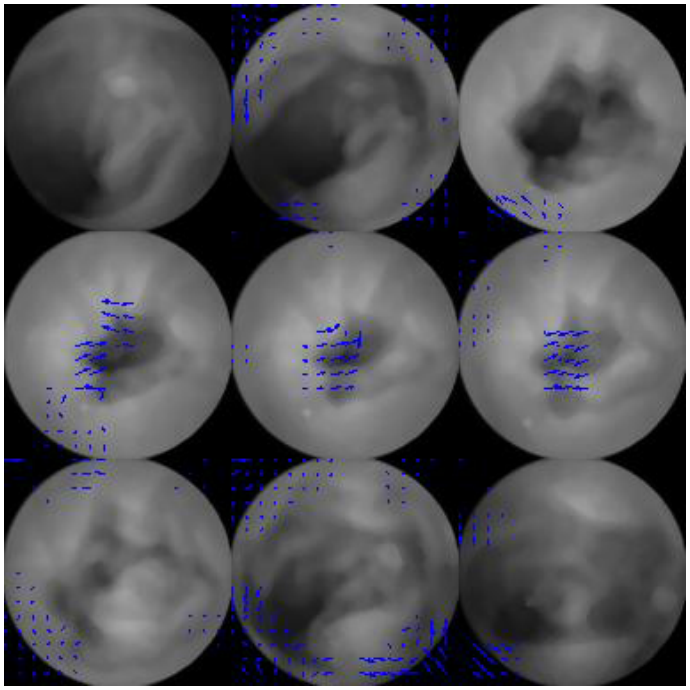


Figure 3.24 – Optical flow estimation from figure 3.23 after applying a median filter and the blob detection approach. Note: median filtered frames are illustrated.

3.4 Implementation

This section provides a brief presentation of the source code which has been implemented during the motion analysis phase of the project. For a smooth integration of the code in the current software framework, its structure has been followed crucially. In this spirit, all of the MATLAB source code files have been embedded in a separate module using the prefix `ma_`. In the following, this module may be referred to as the Motion Analysis Module. For the sake of this papers manageability, the explanation of the code has been kept very compact and only the most important facts are explained. For a more detailed description of the source code, the interested reader may be forwarded to the MATLAB files. Each file has been commented extensively to improve its understanding and to give the ability to be modified and extended easily.
Table 3.1 contains a list of all MATLAB files which have been created for this module, containing a brief description of their purpose. Moreover, the following subsections describe a selection of important files whereas useful information is provided.

`ma_compute_bma_vss.m`

The file `ma_compute_bma_vss` contains source code for the previously described VSS block-matching algorithm for estimating the optical flow of two input images. The implementation is based on the MATLAB version by Arash Jalalian [Jal07] which is available from the MATLAB File Exchange. Unfortunately, the original source code contains several bugs which had to be erased during its adaptation to the current software framework. Furthermore, detailed testing of the algorithm with both synthetic and realistic medical images showed irregularities which have been attributed to the implementation. Further information can be found in the source code file. As a matter of fact, the VSS algorithm could not produce satisfying results with realistic medical images and further work on it has been considered to be too time-consuming and dispensable for the objective of this project.

`ma_compute_pb_gautama.m`

The file `ma_compute_pb_gautama.m` contains source code the previously described PB algorithm for estimating the optical flow of two input images. The implemen-

tation is based on the MATLAB version by Temujin Gautama [Gau04] which is available from the MATLAB File Exchange as well. The original source code has been adapted to be used with the current software framework and in order to make more parameters of its Gabor filtering approach adjustable. Detailed information on the changes can be found in the source code file.

`ma_helper_filter.m`

The MATLAB function `ma_helper_filter` is used to execute pre-processing methods by demanding the original image, filter name and its parameters and delivering the filtered result. When extending the source code by adding new pre-processing methods, their implementation or execution, respectively, should be done with this function.

Table 3.1 – List of implemented MATLAB source code files for the Motion Analysis Module.

`ma_init.m`	Script for the initialisation of the Motion Analysis Module;
`ma_module_path.m`	Function for returning the path of the Motion Analysis Module;
`ma_start.m`	Script for testing purposes of the Motion Analysis Module;
`ma_compute_bma_vss.m`	Block-matching algorithm (BMA) through Various Shape Search (VSS);
`ma_compute_pb_gautama.m`	Phase-based approach by T. Gautama et al.;
`ma_helper_compare.m`	Function for comparing different optical flow algorithms;
`ma_helper_filter.m`	Function for filtering an input image with a given filter and its parameters;
`ma_helper_greyscale.m`	Function for checking an image and converting it to greyscales;
`ma_helper_LoG.m`	Function for applying the Laplacian of Gaussian (LoG) filter to an image;
`ma_helper_logical.m`	Function for creating a logical image by using a desired threshold;
`ma_helper_normalise.m`	Function for vector normalisation;
`ma_helper_resize.m`	Function for checking and resizing an image;
`ma_helper_template.m`	Motion Analysis Module template;

Chapter 4

Feature Extraction

To use the obtained optical flow data for further processing by artificial intelligence technologies it is necessary to establish appropriate features which represent the data set in a convenient way. This chapter is dedicated to the design and extraction process of a simple feature extraction approach. At the end, a short presentation of the implemented source code for the purpose of feature extraction is provided.

4.1 Feature Extraction

The final result of the optical flow estimation is represented by a certain field of motion vectors. Within the terms of this project, these motion vectors correspond to conventional 2-dimensional vectors which may be described by their direction and their magnitude. A common approach for designing simple features is to establish representative histograms. In this spirit, both direction and magnitude of the motion vectors should be the basis of a corresponding histogram feature. Furthermore, the mode of each histogram is stored as well to create another feature. Finally, all of those are combined to form up the dimensions of the final feature vector which may then be used for further classification methods.
Figure 4.1 shows a sequence of two capsule endoscopy images which have been extracted from an intestinal contraction. The second image has been overlaid with the corresponding motion vector field obtained by the phase-based optical flow algorithm. In the following, this motion vector field is processed by the simple histogram feature extraction approach and corresponding results are illustrated.

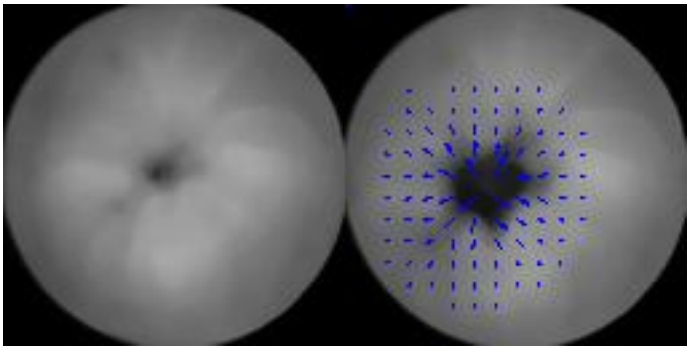


Figure 4.1 – Capsule endoscopy image sequence extracted from an intestinal contraction. The second image has been overlaid with the corresponding optical flow estimation.

4.1.1 Direction Histogram

The direction of a motion vector can be seen as the angle between itself and a certain pre-defined vector. The corresponding angle may then be computed by reforming the scalar product (i.e. dot product) of the two vectors. To make this clear, we want to take a look at the polar co-ordinate system. Figure 4.2 shows an illustrative visualisation with associated angle values. For this simple feature, the scalar product between the actual motion vector and the vector $\binom{1}{0}$ is reformed and expressed in degrees of the polar co-ordinate system. Finally, all of those values are sorted in a certain number of histogram classes (i.e. bins). Figure 4.3 shows a bar chart for the motion vector field from figure 4.1.

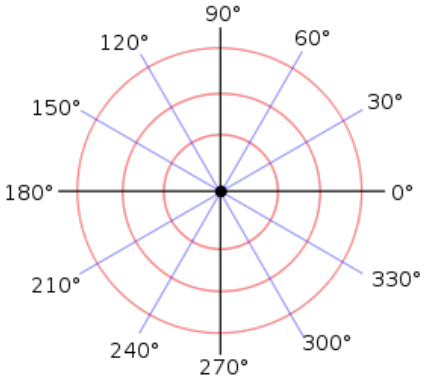


Figure 4.2 – Illustrative visualisation of the polar co-ordinate system.[7]

4.1.2 Magnitude Histogram

The magnitude (i.e. length) of each motion vector is computed using the Euclidean norm which is straightforward and does not need further explanation. The resulting values are sorted in a certain number of histogram classes (i.e. bins) as well. Figure 4.4 shows the corresponding bar chart for the motion vector field from figure 4.1.

4.1.3 Histogram Mode

The computation of the histogram mode is straightforward. It represents the histogram class with the greatest number of elements. In our terms, this refers to the class with the greatest number of motion vectors of a certain direction or magnitude. The mode should provide us with a simple and fast estimation of prime directions and magnitudes of a certain motion vector field. In fact, the histogram mode would be fairly useless when used as a stand-alone feature. But in combination with other features it may be useful in order to extend the number of characteristics for the final feature vector.

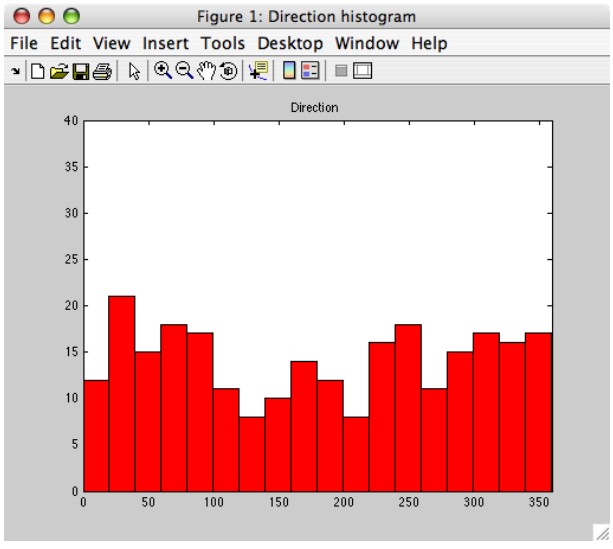


Figure 4.3 – Direction histogram plot for the motion vector field from figure 4.1.

[7]Source: *http://en.wikipedia.org/wiki/File:Polar_graph_paper.svg*, retrieved on February 6, 2009

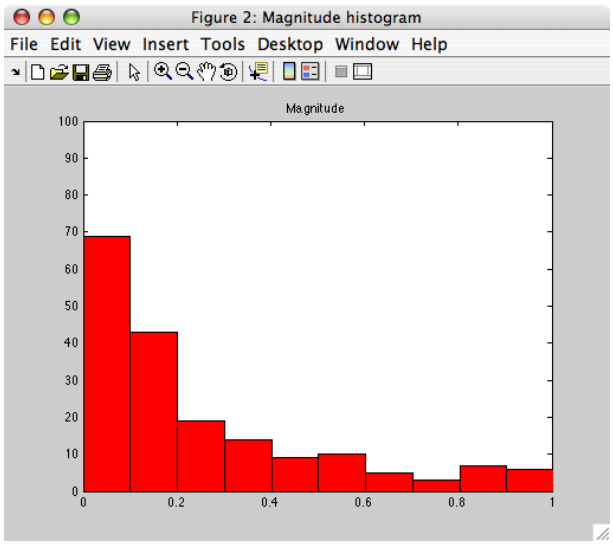


Figure 4.4 – Magnitude histogram plot for the motion vector field from figure 4.1.

4.2 Implementation

This section provides a brief presentation of the MATLAB source code which has been developed during the feature extraction phase of the project. All of the source code files have been embedded in a separate module using the prefix `hist_`. In the following, this module is referred to as the Histogram Module. Table 4.1 contains a list of the MATLAB files for this module, containing a brief description of their purpose. To complete the description of the implementation process of the Histogram Module, the two most important source files are picked out and additional information is explained.

`hist_compute_direction_single.m`

The MATLAB function `hist_compute_direction_single` computes a direction histogram for the motion vector field through reforming the scalar product. As it has been described before, the reference vector for the angle computation is pointing to 0 of the polar co-ordinate system (figure 4.2). The resulting direction values are represented by degrees. The certain number of histogram bins is set to 18 which delivers a range of 20 degrees for each histogram bin. The mode is stored for further computations.

hist_compute_magnitude_single.m

The MATLAB function `hist_compute_magnitude_single` computes the magnitude component for each vector of a motion vector field, normalises the values with respect to the greatest magnitude of the field and hands them over to the MATLAB `hist` function. The number of histogram bins is set to 100 to get a histogram bin for each hundredth of the normalised motion vector magnitudes ranging from 0 to 1. The corresponding mode is computed as well.

Table 4.1 – List of implemented MATLAB source code files for the Histogram Module.

`hist_init.m`	Script for the initialisation of the Histogram Module;
`hist_module_path.m`	Function for returning the path of the Histogram Module;
`hist_start.m`	Script for testing purposes of the Histogram Module;
`hist_compute_direction_single.m`	Function for computing the direction histogram of a motion vector field;
`hist_compute_magnitude_single.m`	Function for computing the magnitude histogram of a motion vector field;
`hist_helper_template.m`	Histogram Module template;

Part II

Graphical User Interface

Chapter 5

The Visualisation Application

One of the main objectives of this project has been the establishment of a software application for experimentation with the previously described set of motion analysis tools. The application should provide a fast and easy to use visualisation environment for the study of motion within the domain of WVCE images.
This chapter is dedicated to the design and implementation process of the visualisation application whereas the development of the graphical user interface (GUI) is explained and useful details on the source code files are provided. In the following, the visualisation application may be referred to as the Visualiser.

5.1 Design

For the sake of a fast implementation process and good co-operation with the previously developed source code, the built-in MATLAB GUI editor (figure 5.1) has been used for the development of the Visualiser GUI. The editor can be run by executing the command `guide` from the MATLAB console. Due to the plainness of the editor and its limited abilities, the final GUI is indeed quite simple. In essence, it follows a strategy of modular boxes (figure 5.2):

- In the top area of the application window we can see the `Information` box. Once a patient study has been loaded, it contains useful information about the study and its image files.

- The `Player` box in the middle implements the functionality of a regular image or video player. It is possible to play the images of a study as a video stream,

with respect to the currently set rate of frames per second, and pause or stop this video at any time to perform further processing on the current image. This is carried out by switching the image over to the `Detail` box.

- The `Detail` box can be identified in the middle area on the right hand side of the `Player`. Once an image has been selected, further image processing steps can be performed, with respect to the selected methods and algorithms in the bottom area of the application window.

- The `Configuration` box, which can be identified in the middle area on the left hand side of the `Player`, is used to manually navigate to either a desired frame or a contraction sequence and to configure further parameters of the application. There is space left for future configuration parameters.

The bottom area of the application window contains selection and configuration options for pre-processing steps and optical flow estimation procedures. All of these steps are carried out on the single image which has been selected to be viewed in the `Detail` box. Each result will be shown in the `Detail` box as well, except for the simple histogram feature approach which will be displayed in a separate figure. Furthermore, each of the procedures can be performed on a sequence of images by choosing the corresponding option from the menu-bar on top of the Visualiser window.
Mainly, this design strategy should make the software highly user-friendly and easy to understand. For a good usability, the Visualiser has been fully equipped with tool-tip texts which briefly explain the functionality of the corresponding user interface component or configuration parameters. Furthermore, each parameter is initially set to proposed optimal values. Altogether, its design should make the Visualiser easily modifiable and extendable which can be performed by adjusting old and appending new boxes respectively.

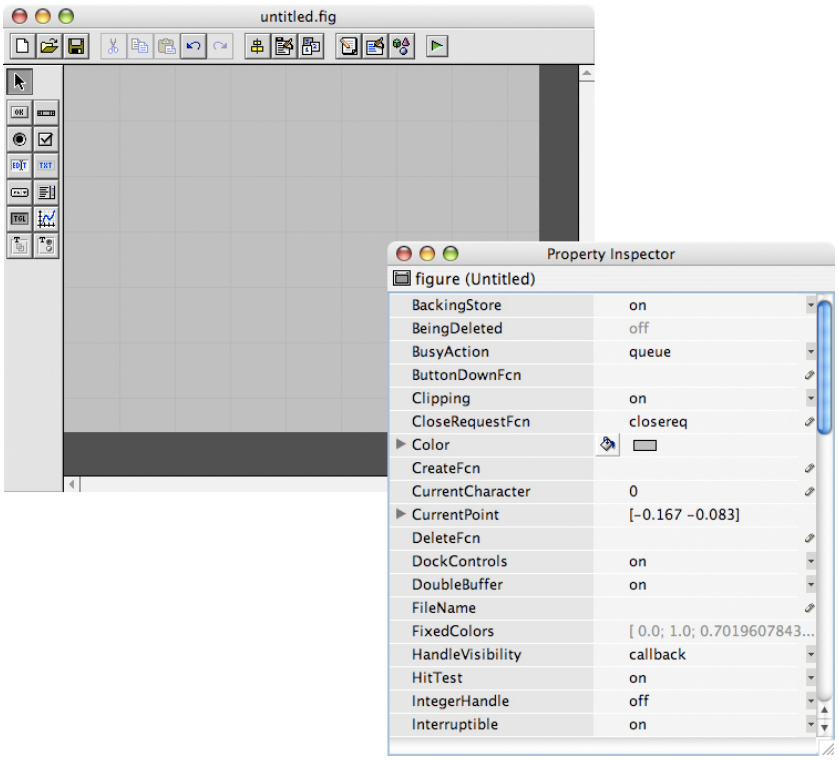


Figure 5.1 – Screenshot of the MATLAB GUI editor.

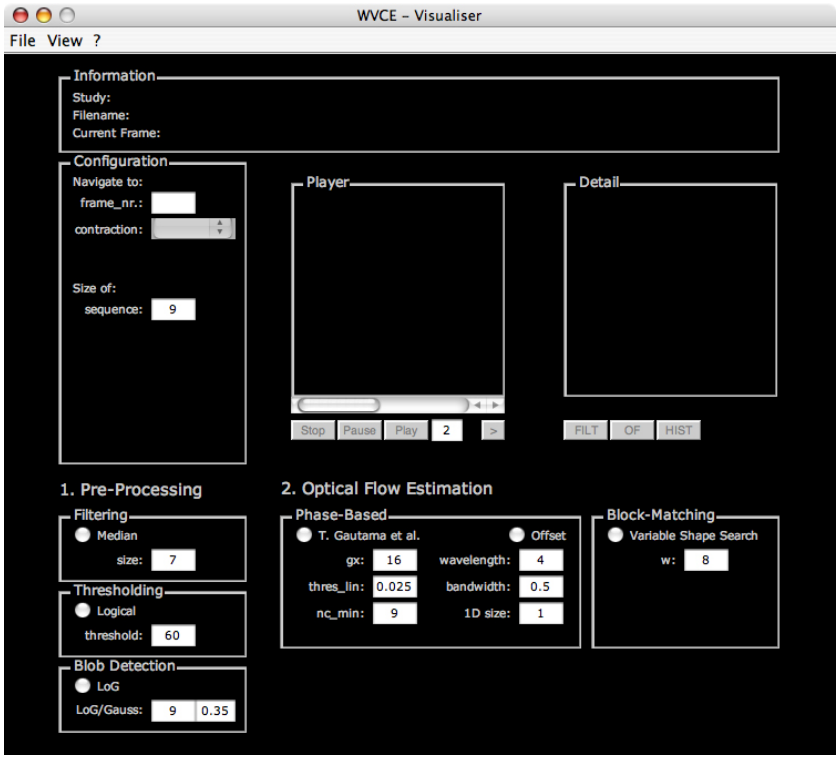


Figure 5.2 – Screenshot of the Visualiser main window.

5.2 Implementation

All of the source code files for the Visualiser have been embedded in a separate module using the prefix `vis_`. In the following, this module will be referred to as the Visualisation Module. Table 5.1 contains a list of MATLAB files which have been created for the Visualisation Module, containing a short description of their purpose. To complete the description of the implementation process, a selection of important source code files and additional information is presented for the objective of reusability.

`vis_start.m`

The MATLAB script `vis_start` represents the prime starting point of the Visualiser. The main purpose of this script is to initialise required global variables and execute the GUI by calling `vis_main` and receiving its handle. The handle is currently unused but may be relevant when the application is extended or included in another MATLAB application framework.

`vis_main.fig`

The file `vis_main.fig` is automatically created by the MATLAB GUI editor and contains the layout of the Visualiser interface in form of a MATLAB figure. It should only be modified by using the editor. Modifying the file manually can make it totally unusable.

`vis_main.m`

The file `vis_main.m` is automatically created by the MATLAB GUI editor as well and contains the main functionality of the Visualiser interface in form of many separate MATLAB functions (i.e. callbacks). If there are modifications to the interface layout via the GUI editor, the corresponding callbacks are automatically created within this file. The functionality has to be implemented manually within the corresponding function inside this file. Each of the GUI elements yields own callbacks which are executed on different behaviour (e.g. key press, mouse click, etc.). Furthermore, each GUI element has a `create` function which is initially executed when the application is launched. This is very useful for setting initial appearance and behaviour of GUI elements.

`vis_map_sequence.m`

The MATLAB function `vis_map_sequence` contains functionality for mapping a sequence of images, with respect to the currently selected pre-processing step and optical flow algorithm. After plotting the whole image sequence, the user can manually estimate a certain position to plot a marker. The result is stored as a high resolution JPEG image for illustration and further image processing.

5.3 Known Bugs and Incompleteness'

This section contains a listing of known bugs and incompleteness' which have been discovered while working with the Visualiser:

- The Visualiser must be closed using the `Close` option from the `File` menu. Otherwise, it may come to irregularities.
- Only one pre-processing step can be enabled at a time.
- Mapping all contractions of a study folder is not yet implemented correctly because the first approach did crash the Visualiser due to the great amount of data.
- Viewing all contraction histograms of a study folder is not yet implemented correctly because the first approach did crash the Visualiser due to the great amount of data.
- Only one study folder can be loaded at a time.
- Mapping a contraction can crash the Visualiser. In this case it may be restarted and shut down correctly by using the `Close` option from the `File` menu. After that, it should work fine again.

This listing is related to user interface issues and should complete the explanation of the GUI design and implementation process. Furthermore, it should be assistant to further programmers who are planning to reuse, modify and extended the Visualiser.

Table 5.1 – List of implemented MATLAB source code files for the Visualisation Module.

`vis_init.m`	Script for the initialisation of the Visualisation Module;
`vis_module_path.m`	Function for returning the path of the Visualisation Module;
`vis_start.m`	Script for running and maintaining the GUI;
`vis_main.fig`	Visualiser GUI layout;
`vis_main.m`	Visualiser GUI functionality;
`vis_map_sequence.m`	Function for mapping a sequence of images;
`vis_helper_template.m`	Visualisation Module template;

Chapter 6

User Manual

This chapter is dedicated to the interaction process between the presented software and its user. On the one hand, it provides a comprehensive user manual with detailed application descriptions and actual screenshots of each user interface component. On the other hand, it illustrates the usage of the Visualiser on the basis of an example procedure which is carried out in an iterative manner and may be reproduced step by step. Altogether, it should be considered as the prime source of information belonging to the correct usage of the Visualiser.

6.1 The Visualisation Application

First of all, the Visualisation Module needs to be initialised by executing `vis_init` from the MATLAB console which will set-up all necessary MATLAB paths. After that, the Visualiser is started by executing `vis_start` from the MATLAB console. The application will then be loaded and the main window of the GUI (figure 5.2) will be displayed.
The first step of each session should be to load a patient study by choosing the option `Open Study ...` from the `File` menu of the menu-bar on top of the main window (figure 6.1). This will display the `Browse For Folder` user interface dialogue which allows you to browse your hard drive and select a study folder (figure 6.2). Once a correct folder is selected, the MATLAB workspace is set-up by loading initially required data and the GUI is updated by showing the first image of the study in the `Player` box and listing current information in the `Information` box on top of the window (figure 6.3). The Visualiser is then ready to perform further experimentation

with the study. In essence, there are two main functions:

1. The player functionality. The `Player` box right in the middle of the application window implements the functionality of a regular video player. It allows you to browse through the images in several different ways in order to select a certain image for further image processing.

2. The image processing functionality. Image processing functions include pre-processing methods and optical flow estimation algorithms which can be selected by using the corresponding box from the two sections, visible in the bottom area of the application window.

An important fact is that every image processing function is carried out on a single image and the previous one, in case of an optical flow estimation respectively. If an image sequence is regarded, image processing functions may be carried out on the whole sequence by selecting the corresponding option from the `View` menu of the menu-bar on top of the Visualiser window. Those options will view additional MATLAB figures with corresponding results. Starting number and length of a sequence can be adjusted by selecting the start image in the `Detail` box and setting the sequence length parameter in the `Configuration` box of the main window.

6.2 The Player

The player functionality allows you to browse through the images of a patient study using four different methods:

1. Press the `Play` button. This will automatically play the images as a video stream consisting of single frames and with respect to the currently set rate of frames per second which can be changed by entering a valid number in the corresponding field below the `Player` box and pressing the enter-key of the keyboard. The automatic playback can be paused and stopped at any time by pressing the corresponding buttons `Pause` and `Stop`.

2. Drag the slider below the `Player` box to jump to any desired frame.

3. Enter the desired frame number in the corresponding field of the `Configuration` box and hit the enter-key. If the entered frame number is valid, the `Player` box will display the desired frame.

4. Select the desired contraction number from the pop-up menu in the `Configuration` box. If a valid contraction number is selected, the `Player` box will display the first frame of the desired contraction sequence.

If an image is displayed in the `Player` box, the `>` button will be enabled. It allows you to select the currently displayed image and flip it over to the `Detail` box for further image processing (figure 6.4). By pressing the `FILT` button, the selected function from the pre-processing section will be executed on the currently selected image and the results will be shown in the `Detail` box. Further explanation on how to use and configure pre-processing methods can be found in the next section of this chapter. By pressing the `OF` button, the selected function from the optical flow estimation section will be applied to the currently selected image. The preceding image will be the reference for the estimation of the optical flow. Once again, the result will be shown in the `Detail` box. Further explanation on usage and configuration of the optical flow estimation algorithms can be found in the corresponding section of this chapter. If the estimation of the optical flow has been executed successfully and the corresponding motion vector field is shown in the `Detail` box, the `HIST` button will be enabled. By pressing it, a new MATLAB figure is displayed which contains a direction and a magnitude histogram for the currently available optical flow motion vector field.

6.3 Pre-Processing Section

The lower left area of the Visualiser window contains boxes for adjusting pre-processing methods including Median and logical filtering as well as a blob detection approach through the application of the Laplacian of Gaussian filtering method. Those will be applied previous to further computations and should therefore be the first thing to think about when starting a new experimentation session. By clicking the corresponding radio button, a method will become enabled, or disabled respectively. By leaving out the selection, none of the just mentioned methods will be applied. Furthermore, it should be mentioned that currently only one of these pre-processing steps can be enabled via the graphical user interface at the same time. If a method can be configured by choosing one or more parameters, this is performed by typing the desired value into the corresponding field and pressing the enter-key of the keyboard to set-up its value. All parameters are initially set to optimal de-

fault values which are also explained in the corresponding tool-tip text. To find out more about the parameters, please study the corresponding section of this paper. Figure 6.5 shows the result of a 7-by-7 median filter which has been applied to the previously selected image from figure 6.4.

6.4 Optical Flow Estimation Section

The lower right area of the Visualiser window contains boxes for adjusting optical flow estimation algorithms including the phase-based approach and the block-matching technique. The estimation of the optical flow is the main purpose of the Visualiser and should be considered after selecting a pre-processing step. Only one algorithm or none can be selected at the same time by clicking the corresponding radio button. Once more, parameters can be changed by typing the desired value into the corresponding field and pressing the enter-key to set-up the new value. When starting the Visualiser, all parameters are set to optimal default values which are explained in the corresponding tool-tip text too. Figure 6.6 shows the result of applying the phase-based optical flow estimation approach to the previously selected image from figure 6.4. The `Player` box displays the preceding reference image.

6.5 The Mapping Tool

The term mapping tool describes all functions and user interface elements which have been implemented to map a sequence of images. Mainly, separate MATLAB figures are created which can be saved to high resolution JPEG files and printed out using the MATLAB figure print function. The mapping tool can be executed by selecting the option `Map sequence ...` from the `View` menu of the menu-bar on top of the Visualiser window.

Choosing the option `Map sequence ...` displays a new MATLAB figure and plots a sequence of images with respect to the currently selected pre-processing method, optical flow estimation algorithm and sequence size (figure 6.7). A sequence of images can be plotted with or without both pre-processing and optical flow estimation steps. The plotting procedure takes some time within no other user input should be performed. Otherwise the image sequence may not be plotted correctly and the Visualiser could crash. After plotting, the whole image sequence is displayed and

you can manually select a point by moving the mouse to the desired position and clicking the left mouse-button. This function is optional and can be omitted by pressing the enter-key on the keyboard to proceed without marking.

6.6 The Histogram Tool

The term histogram tool describes all functions and user interface elements which have been implemented to create motion vector field histograms. The main histogram tool (figure 6.8) can be executed by choosing the option `View histogram ...` from the `View` menu of the menu-bar on top of the Visualiser window.
Choosing the option `View histogram ...` creates a new MATLAB figure with both direction and magnitude histograms for each motion vector field of the the currently computed sequence. The histograms may be stored as JPEG or MATLAB files by using the corresponding MATLAB save function.

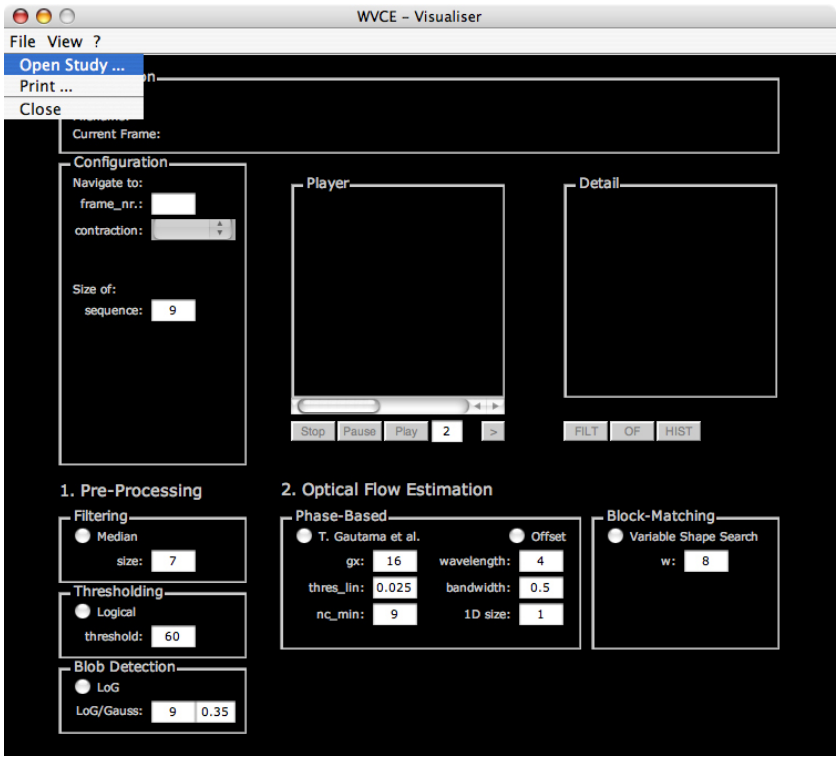


Figure 6.1 – Screenshot of the `File` menu.

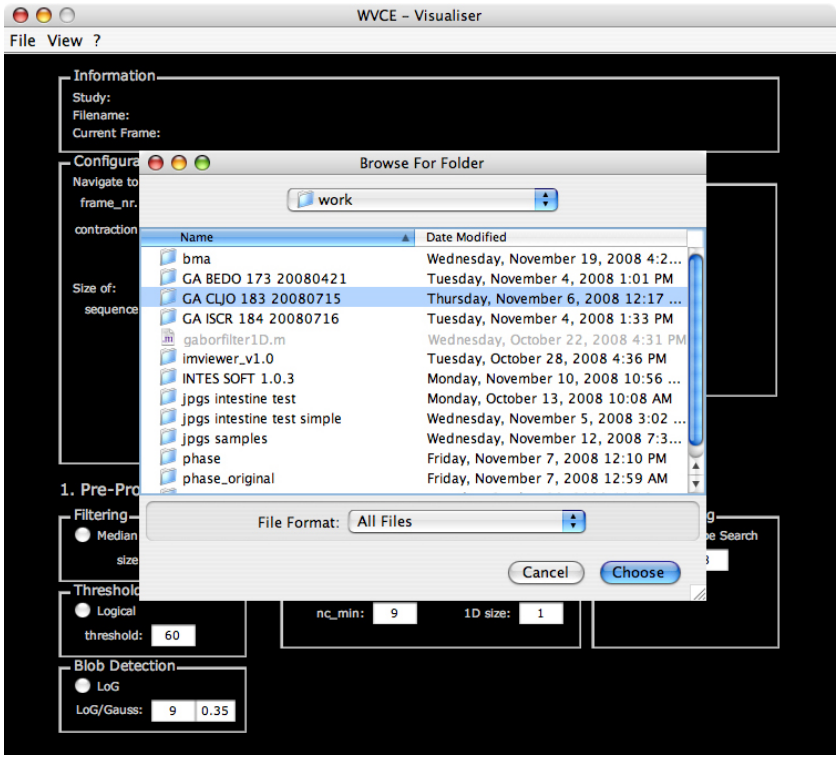


Figure 6.2 – Screenshot of the `Browse For Folder` user interface dialogue.

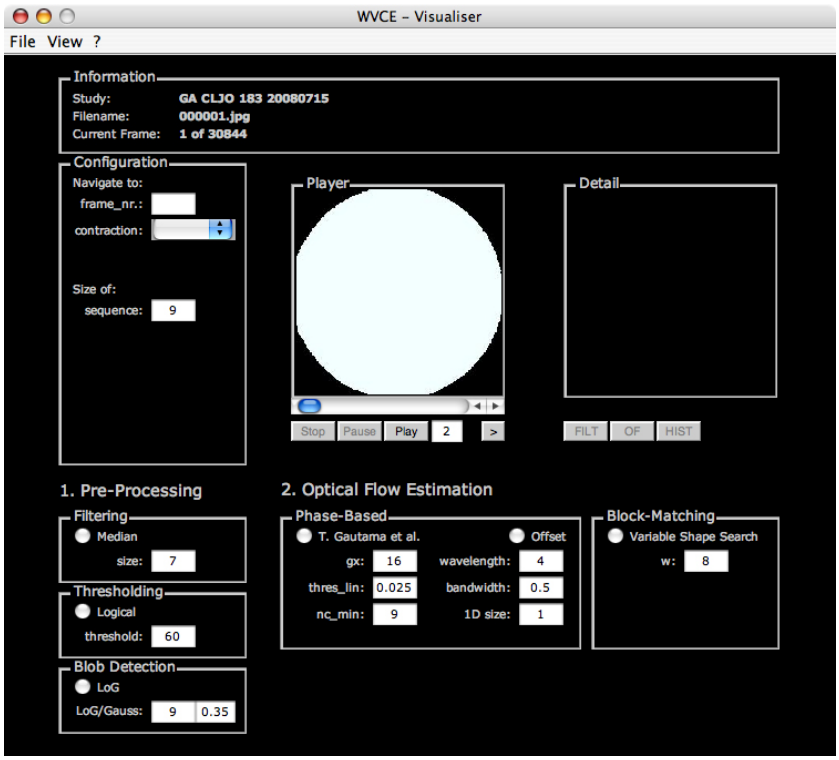


Figure 6.3 – The Visualiser after loading a patient study folder.

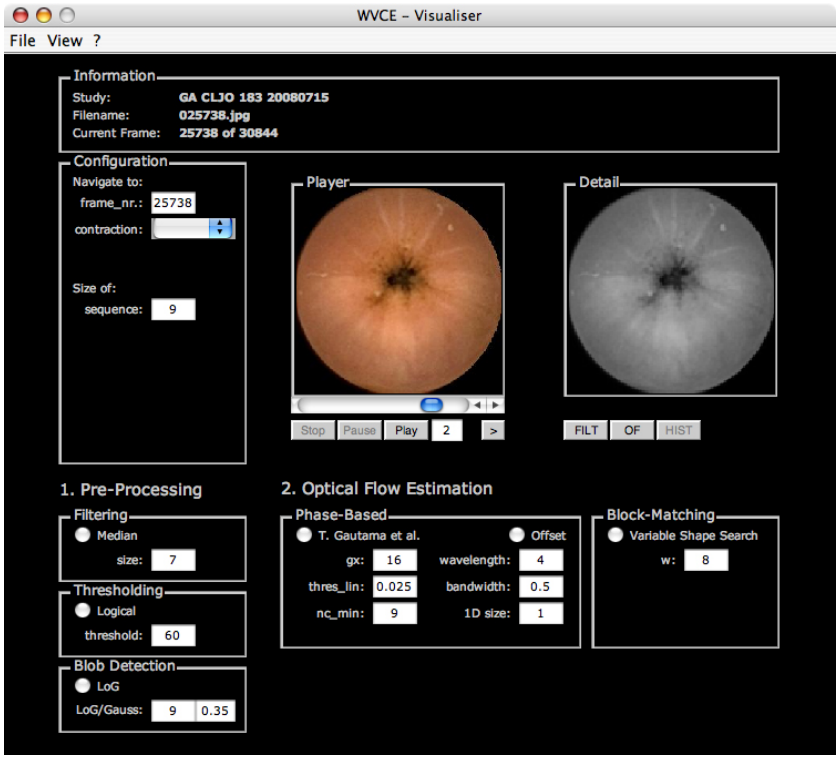


Figure 6.4 – Flipping an image over to the `Detail` box for further image processing.

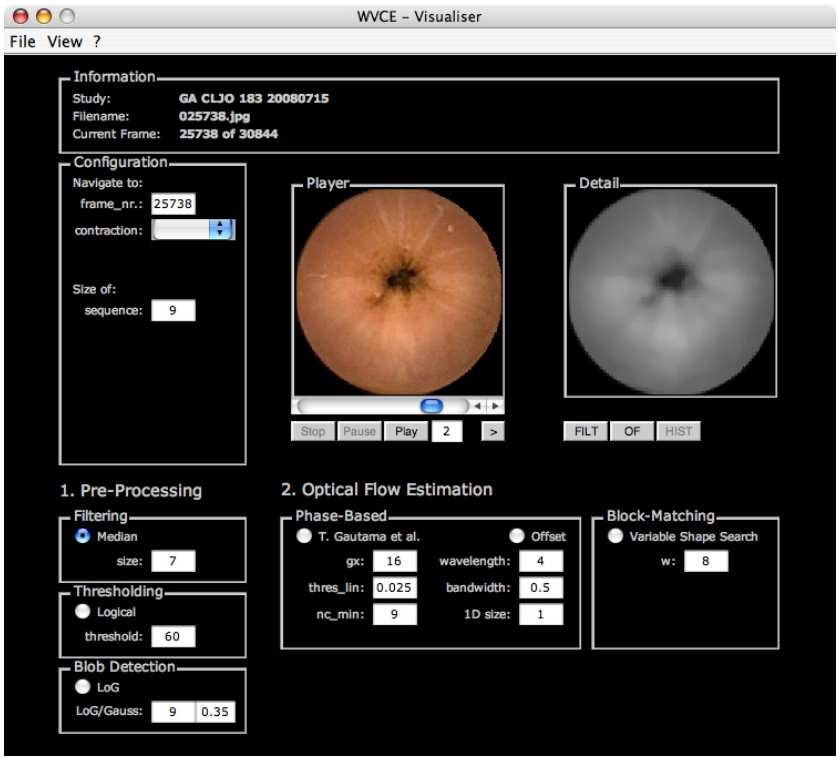


Figure 6.5 – Application of a pre-processing step to the currently selected image.

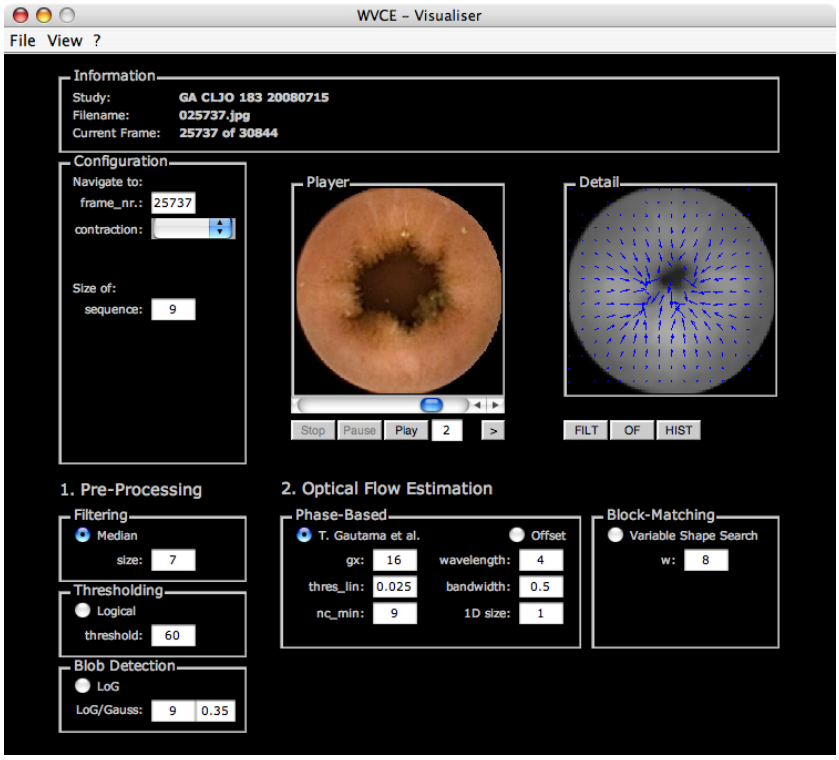


Figure 6.6 – Application of an optical flow algorithm to the currently selected image.

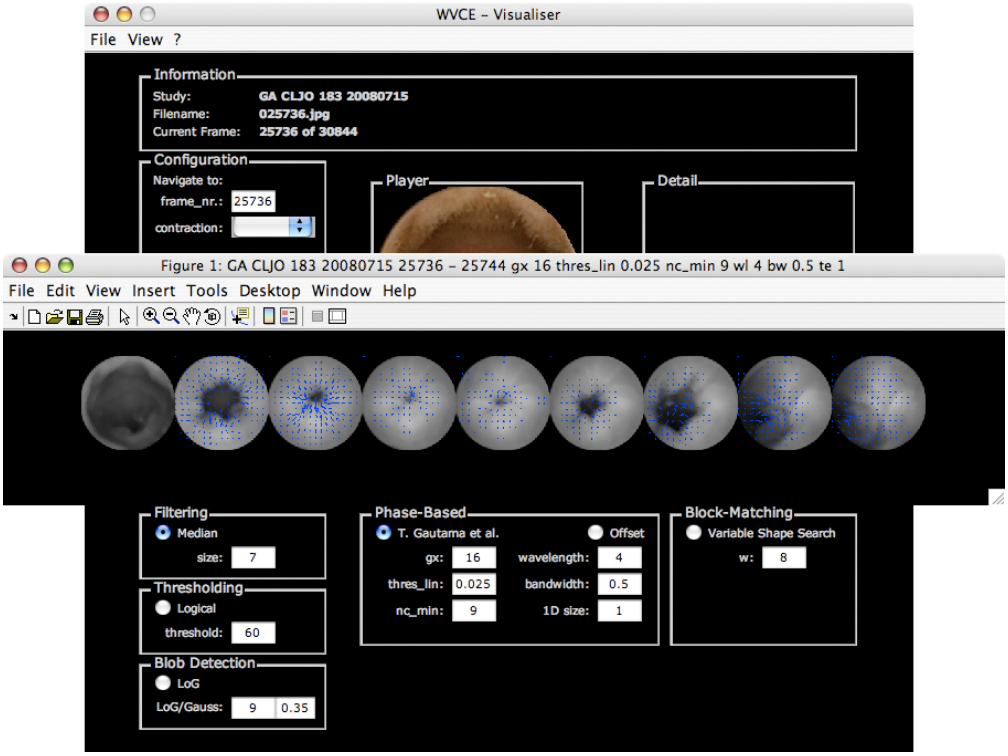


Figure 6.7 – Screenshot of using the mapping tool.

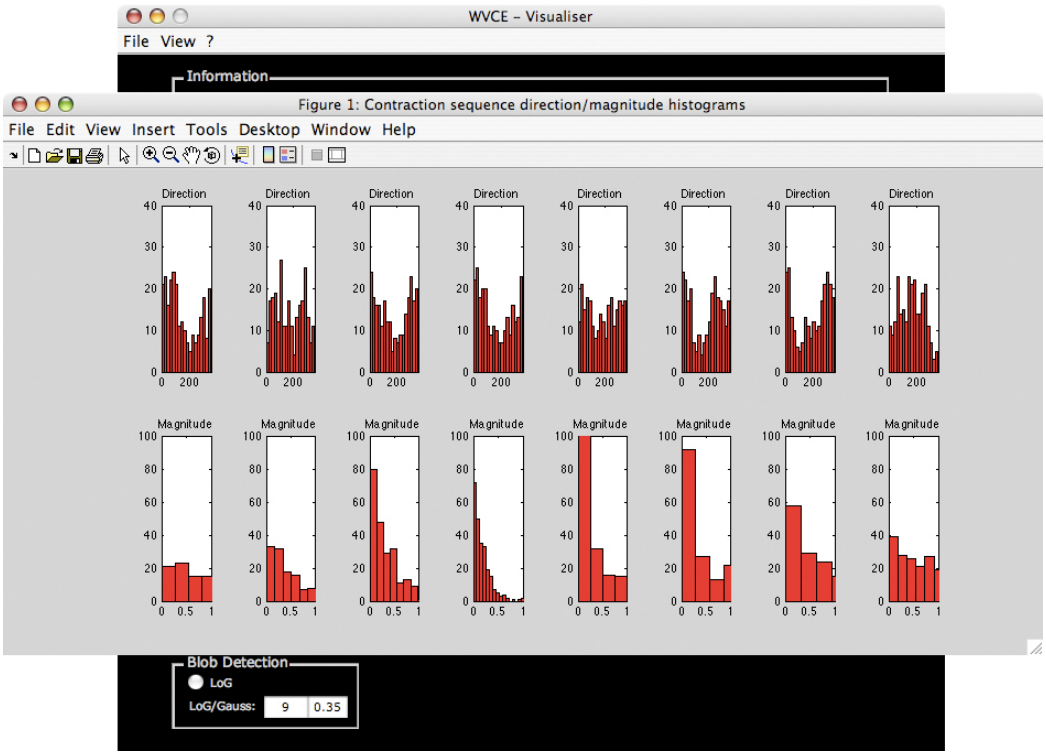


Figure 6.8 – Screenshot of using the histogram tool.

Part III

Discussion, Final Conclusions and Further Work

Chapter 7

Discussion and Final Conclusions

The presented set of tools, combining pre-processing methods, optical flow algorithms and a simple feature extraction approach, has been proposed to provide a useful source of information when studying motion in intestinal contraction images obtained by WVCE. Throughout the previous chapters, this objective has been worked out in detail. At this point, we shall now focus on discussing major findings and providing final conclusions.

All things considered, it has to be argued that the comparable poor frame rate of the capsule endoscopy video material demonstrates a major problem for applying motion analysis procedures. Due to the fact that there can be great changes between two subsequent frames, the estimation of appropriate parameters is a crucial task. Experimental results show that there can be a good estimation of the lumen motion with the provided set of tools, once appropriate parameters have been determined. Nevertheless, the same parameters may fail on another contraction sequence. For this reason, it is not only a matter of what combination of methods and algorithms to use but how those are adapted to the changeable appearance of intestinal contraction sequences obtained by WVCE. The determination of global parameters for both pre-processing and optical flow estimation can be seen as a major source of discontinuities.

The median and logical filtering methods have been quite useful in order to reduce noise and may be appropriate in many different situations. To be considered as pre-processing steps, both median and logical filtering can be highly recommended within the scope of capsule endoscopy image sequences. Especially initial median filtering has been appropriate at almost any time while working with contraction se-

quences as well as ground truth solutions. The blob detection approach through the application of the Laplacian of Gaussian operator provided a convenient separation of the lumen from the intestinal wall while it suffers from the previously mentioned frame rate problem. Nevertheless, it can be recommended as well while there is probably plenty of space for further improvement (e.g. through dynamic parameter estimation).

For estimating the optical flow in intestinal contraction sequences, the block-matching technique has not produced convenient results. The resulting motion vector field contains to much noise and due to the fact that there is not really much room for improvement except for the choice of the macro-block size, it may not be recommended for this objective. The phase-based approach on the other hand shows expected results, once it had been adapted to the current problem, of course. Maybe there is even more improvement of the final results by dynamically adapting the parameters of the included Gabor filtering approach to the current contraction sequence.

The Visualiser has been a convenient visualisation application in order to apply both pre-processing methods and optical flow algorithms in a fast and easy way. Furthermore, it has improved presentational purposes.

Chapter 8

Further Work

The final chapter of this paper yields a brief proposal of which further experimentation could be done with the proposed set of motion analysis tools and how it could be improved and extended to carry on with the objective of this project.
For further improvement of the pre-processing section, the blob detection approach could be revised. In order to minimise the drawbacks of determining certain global parameters, an approach to dynamically estimate appropriate parameters for each contraction sequence may be established. Moreover, the experimentation with different parameters for both morphological dilation and closure has not been carried out extensively and could provide room for improvement of the blob detection and final optical flow estimation results.
The optical flow estimation section may be improved by separating the motion of the lumen from the motion of the intestinal wall in order to be observed individually. This may be carried out by detecting the current position of the lumen and extracting a certain region around it (e.g. by utilising the blob detection approach). Probably, this would lead to a more focused and noise reduced experimentation environment for the study of the lumen motion. Furthermore, one could focus on utilising the motion vector field around the lumen for estimating the location of the intestinal contraction bouncing point.
For further classification, the simple feature extraction approach needs to be tested with a classifier and may be revised and improved by the development of additional features in order to extend the characteristics of the final feature vector.
Concerning the Visualiser, the previously mentioned bugs and incompleteness' may be eliminated by revising and extending the current source code.

Bibliography

[ASM08] K. Aires, A. Santana, and A. Medeiros. Optical flow using color information: Preliminary results. In *SAC '08: Proceedings of the 2008 ACM Symposium on Applied Computing*, pages 1607–1611, Fortaleza, Ceara, Brazil, 2008.

[BA96] M. Black and P. Anandan. The robust estimation of multiple motions parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[Bar04] A. Barjatya. Block-matching algorithms for motion estimation. *DIP 6620 Final Project Paper - Utah State University*, 2004.

[BBPW04] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *ECCV 2004: 8th European Conference on Computer Vision*, volume 4, pages 25–36, Prague, Czech Republic, 2004.

[BNB04] D. Boukerroui, J. A. Noble, and M. Brady. On the choice of band-pass quadrature filters. *Journal of Mathematical Imaging and Vision*, 21(1):53–80, 2004.

[Gau04] T. Gautama. Matlab central - file detail - phase-based optical flow. http://www.mathworks.com/matlabcentral/fileexchange/2422, 2004.

[GvH02] T. Gautama and M. M. van Hulle. A phase-based approach to the estimation of the optical flow field using spatial filtering. In *IEEE Transactions on Neural Networks*, volume 13, pages 1127–1136, 2002.

[HWJJ06] L. Hao, Z. Wen-Jun, and C. Jun. A fast block-matching algorithm based on variable shape search. *Journal of Zhejiang University - Science A*, 7(2):194–198, 2006.

[Ima08] Given Imaging. Given imaging home - gastrointestinal (gi) and digestive disease diagnosis. http://www.givenimaging.com, 2008.

[ISV+07] L. Igual, S. Segui, J. Vitria, F. Azpiroz, and P. Radeva. Eigenmotion-based detection of intestinal contractions. In *CAIP 2007: 12th International Conference on Computer Analysis of Images and Patterns*, pages 293–300, Vienna, Austria, 2007.

[Jal07] A. Jalalian. Matlab central - file detail - fast motion detection(bugs fixed). http://www.mathworks.ch/matlabcentral/fileexchange/15767, 2007.

[Mov08] J. R. Movellan. Tutorial on gabor filters. http://mplab.ucsd.edu/tutorials/pdfs/gabor.pdf, 2008.

[NMR92] K. R. Namaduri, R. Mehrotra, and N. Ranganathan. Edge detection models based on gabor filters. In *IAPR 1992: 11th International Conference on Pattern Recognition*, pages 729–732, The Hague, The Netherlands, 1992.

[OH96] Y. Okano and H. Hamada. Estimation of varying frequency by gabor filters and neural network. In *APCCAS 1992: IEEE Asia Pacific Conference on Circuits and Systems*, pages 504–507, Seoul, South Korea, 1996.

[PW08] N. Petkov and M. B. Wieling. Gabor filter for image processing and computer vision. http://matlabserver.cs.rug.nl/cgi-bin/matweb.exe, 2008.

[SIV+08] S. Segui, L. Igual, F. Vilariño, P. Radeva, C. Malagelada, F. Azpiroz, and J. Vitria. Diagnostic system for intestinal motility disfunctions using video capsule endoscopy. In *ICVS 2008: 6th International Conference on Computer Vision Systems*, pages 251–260, Santorini, Greece, 2008.

[Vil06] F. Vilariño. *A Machine Learning Approach for Intestinal Motility Assessment with Capsule Endoscopy*. PhD thesis, Computer Vision Center, Universitat Autònoma de Barcelona, Bellaterra, Spain, 2006.

[VSV+06] F. Vilariño, P. Spyridonos, J. Vitria, F. Azpiroz, and P. Radeva. Cascade analysis for intestinal contraction detection. In *CARS 2006: 20th International Congress and Exhibition Computer Assisted Radiology and Surgery*, Osaka, Japan, 2006.