

# Objective Methods of Evaluating Colour Image Segmentation



Hassan Almuhairi

A thesis submitted for the degree of *Doctor of Philosophy*  
School of Computer Science and Electronic Engineering  
University of Essex

Date of submission: July 2010

*I would like to dedicate this thesis to my loving  
wife and parents ...*

**W**RITERS imagine that they cull stories from the world. I'm beginning to believe that vanity makes them think so. That it's actually the other way around. Stories cull writers from the world. Stories reveal themselves to us. The public narrative, the private narrative - they colonize us. They commission us. They insist on being told. Fiction and nonfiction are only different techniques of storytelling. For reasons I do not fully understand, fiction dances out of me. Nonfiction is wrenched out by the aching, broken world I wake up to every morning.

---

Arundhati Roy

## Acknowledgements

**T**HIS thesis is part of many years of research that has been performed since I came to University of Essex. By this time, I had the pleasure of knowing and working with a great number of people who contributed in many ways to the nice time I spent there and helped me with my research. And although it's hard to mention them all here, I know they feel my appreciation to them all and I would like to convey my humble gratitude: Thank you.

Several people have played a decisive role in helping me persevering through my PhD studies and not quit. None had done more than my wife, who was so supportive and understanding through out the process. My loving parents and family back home through their regular calls and comforting words. Also I don't forget my friends here in Colchester and in the university, without their laughs and and nice times I would not have been able to get through this time.

I am deeply indebted to my supervisors, Adrian Clark and Martin Fluery for their thoughtful guidance. This was not an easy journey for me and they were helpful and understanding all the way through it. They were my coaches and collaborators. I could not have wished

for better supervisors. Your contributions, detailed comments and insight have been of great value to me.

I also would like to thank Khalifa University (previously known as Etisalat College of Engineering) for their belief in me and their support all the time.

It's amazing how  
a friend I have not yet met  
can brighten my day

---

A Thank You Haiku by Unknown

## Abstract

**I**MAGE segmentation constitutes an important step in automatic object recognition. However, there is still no agreement on a mathematical model that can represent the segmentation process. As a result, a large variety of segmentation algorithms have been introduced into the image processing literature.

The lack of a single standard segmentation solution has led to further research that provides different evaluation models, frameworks and a small variety of image data-sets for testing purposes. A researcher again faces a dilemma of choice: it is hard to decide on a standard evaluation solution to choose an appropriate algorithm. If the goal is to extensively test and evaluate the segmentation algorithms, the task reported in this thesis, then there are: 1) many segmentation algorithms; 2) variety of input images; 3) different input parameters; and 4) diverse evaluation methods. Consequently, this evaluation task can prove to be computationally highly demanding and it may prove hard to approach an optimal solution.

To help research in this field, the author has firstly investigated the current range of segmentation algorithms, with the aim of composing a

‘navigation map’ of the available algorithms and evaluation methods. The thesis also proposes an evaluation methodology.

The research specifically involved using and customising a scripted evaluation framework that performed real-time colour image segmentation and as a result provided an objective assessment of the best-quality image segmentation algorithm for a given application. To enhance the computational performance of the framework: firstly, a cluster computer was used to enhance throughput; and secondly, a genetic algorithm module was added to the evaluation process to improve the evaluation’s search efficiency. Furthermore, the introduction of a time-factor into the genetic algorithm proved to be beneficial in a variety of ways explored in the thesis.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Image Segmentation . . . . .	2
1.2	Motivation of the Research . . . . .	5
1.3	Contribution of the Thesis . . . . .	7
1.4	Outline of the Thesis . . . . .	8
<b>2</b>	<b>A Review of Research on Image Segmentation and its Evaluation</b>	<b>11</b>
2.1	Introduction . . . . .	11
2.2	Taxonomy of Image segmentation methods . . . . .	13
2.2.1	Boundary Demarcation Segmentation . . . . .	18
2.2.1.1	Thresholding Segmentation . . . . .	18
2.2.1.2	Edge Detection Segmentation . . . . .	22
2.2.2	Contouring Techniques . . . . .	25
2.2.2.1	Region-growing Segmentation . . . . .	25
2.2.2.2	Level-Set Segmentation . . . . .	25
2.2.2.3	Watershed Segmentation . . . . .	28
2.2.2.4	Rain-falling Watershed approach . . . . .	28
2.2.3	Learning-Based Techniques . . . . .	29
2.2.3.1	Clustering-Based Segmentation . . . . .	29
2.2.3.2	Artificial Neural Network based Segmentation . . . . .	32
2.2.4	Physics Based Techniques . . . . .	34
2.3	Improving segmentation computational performance with parallelisation . . . . .	35



2.4	Segmentation Evaluation . . . . .	37
2.4.1	Segmentation evaluation categorisation according to the use of a priori information . . . . .	38
2.4.1.1	Unsupervised Segmentation Evaluation . . . . .	38
2.4.1.2	Supervised Segmentation Evaluation . . . . .	39
2.4.1.3	Commentary . . . . .	39
2.4.2	Zhang Segmentation Evaluation Categorisation . . . . .	41
2.4.2.1	Analytical Methods . . . . .	42
2.4.2.2	Empirical Goodness Methods . . . . .	42
2.4.2.3	Empirical Discrepancy Methods . . . . .	43
2.5	Summary . . . . .	43
<b>3</b>	<b>Analytical evaluation of image segmentation algorithms</b>	<b>47</b>
3.1	Image Segmentation Algorithms: analysis . . . . .	47
3.2	Profiling using Valgrind’s Tool Suite . . . . .	49
3.3	Analytical evaluation of thresholding algorithms . . . . .	50
3.4	Analytical evaluation of edge-detection algorithms . . . . .	53
3.5	Analytical evaluation of region-growing segmentation algorithm .	58
3.6	Analytical evaluation of watershed-based algorithms . . . . .	60
3.7	Analytical evaluation of clustering-based image segmentation al- gorithms . . . . .	66
3.8	Analytical evaluation of graph-based image segmentation algorithms	71
3.9	Analytical evaluation of physics-based image segmentation algo- rithms . . . . .	73
3.10	Commentary . . . . .	74

<b>4 Parameters' significance on the segmentation algorithms' performance</b>	<b>75</b>
4.1 Overview . . . . .	75
4.2 The choice of the algorithms and related parameters . . . . .	77
4.3 Thresholding algorithms' parameters . . . . .	78
4.4 Edge detection algorithms parameters . . . . .	80
4.5 Region growing segmentation parameters . . . . .	85
4.6 Watershed segmentation parameters . . . . .	87
4.7 Rain-falling segmentation parameters . . . . .	90
4.8 Colour Watershed segmentation parameters . . . . .	94
4.9 K-means segmentation parameters . . . . .	98
4.10 Mean-shift segmentation parameters . . . . .	101
4.11 Graph-based segmentation parameters . . . . .	104
4.12 Anisotropic-based segmentation parameters . . . . .	107
4.13 Concluding Remarks . . . . .	111
<b>5 Evaluation framework for automatic generation of tests</b>	<b>116</b>
5.1 Introduction . . . . .	116
5.2 Evaluation Environment . . . . .	118
5.3 Using and extending FATE . . . . .	119
5.3.1 Operating modes . . . . .	120
5.3.1.1 Running tests . . . . .	122
5.3.1.2 Evaluating segmentation algorithms . . . . .	122
5.3.2 Working with the harness . . . . .	123
5.3.2.1 Interfacing to the algorithm under test . . . . .	123

5.3.2.2	Specifying the tests. . . . .	124
5.4	Reducing evaluation time . . . . .	125
5.4.1	Using a cluster computer . . . . .	125
5.5	Evaluation methods . . . . .	126
5.6	The Berkeley segmentation dataset and benchmark . . . . .	127
5.6.1	Berkeley group's segment and segmentation definitions . . .	128
5.7	Testing the human hand-segmentation precision . . . . .	130
5.8	Concluding Remarks . . . . .	140
<b>6</b>	<b>Genetic algorithm optimisation for the evaluation framework</b>	<b>142</b>
6.1	Using genetic algorithms . . . . .	142
6.2	Segmentation Error Measure . . . . .	146
6.2.1	Error Measure Definitions and Equations . . . . .	146
6.3	Exhaustive parameter search testing . . . . .	148
6.4	Using a genetic algorithm . . . . .	150
6.4.1	Genetic Algorithms: Basics . . . . .	150
6.4.2	The GA implementation details . . . . .	152
6.5	GA optimisation results . . . . .	153
6.5.1	Example of GA improvement over exhaustive search . . . . .	153
6.6	Adding time as a factor . . . . .	164
6.6.1	The different models of adding the time factor . . . . .	167
6.6.2	The timing experiments image data-set and GA parameters	169
6.6.3	Application of the time-weighted parameter search experi- ment . . . . .	175
6.6.3.1	Colour edge detection evaluation . . . . .	176

6.6.3.2	Watershed segmentation evaluation . . . . .	178
6.6.3.3	Colour Watershed segmentation evaluation . . . . .	182
6.6.3.4	K-means segmentation evaluation . . . . .	185
6.6.3.5	Mean-shift segmentation evaluation . . . . .	189
6.6.3.6	Graph-based segmentation evaluation . . . . .	193
6.6.4	Applying time-weighted parameter search on objective seg- mentation evaluation . . . . .	195
6.6.4.1	Objective evaluation metrics definitions and equa- tions . . . . .	196
6.6.4.2	Results of the $F$ measure evaluation . . . . .	199
6.7	Concluding Remarks . . . . .	202
6.7.1	Summary . . . . .	202
6.7.2	Discussion . . . . .	204
<b>7</b>	<b>Conclusion</b> . . . . .	<b>209</b>
7.1	Findings . . . . .	210
7.2	Reflections . . . . .	212
7.3	Improvements and Further work suggestions . . . . .	214
<b>A</b>	<b>Publications</b> . . . . .	<b>216</b>
<b>B</b>	<b>Appendix B</b> . . . . .	<b>218</b>
B.1	Thresholding evaluation . . . . .	218
B.2	Edge Detection evaluation . . . . .	221
B.2.1	Canny Edge Detection evaluation . . . . .	223
B.3	Rain-falling Watershed segmentation evaluation . . . . .	224

## CONTENTS

---

B.4 Anisotropic-based segmentation evaluation . . . . .	224
<b>C Appendix C</b>	<b>228</b>
<b>References</b>	<b>258</b>

# List of Figures

1.1	Image understanding system flow chart . . . . .	4
2.1	Taxonomy of segmentation methods . . . . .	15
2.2	Histogram thresholding example . . . . .	19
2.3	Thresholding example . . . . .	21
2.4	Example of edges in a grey-scale image . . . . .	23
2.5	Edge detection examples . . . . .	24
2.6	Region growing example . . . . .	26
2.7	Level-Set technique illustration . . . . .	27
2.8	Watershed examples . . . . .	30
2.9	Clustering-based segmentation examples . . . . .	33
2.10	Evaluation methods categorisation . . . . .	40
3.1	Classic thresholding method flow chart . . . . .	50
3.2	Profiling result of the classic thresholding method . . . . .	51
3.3	Relative thresholding method flow chart . . . . .	52
3.4	Profiling result of relative thresholding method . . . . .	52
3.5	Optimal thresholding method flow chart . . . . .	54
3.6	Profiling result of optimal thresholding method . . . . .	54
3.7	Classic edge detector method flow chart . . . . .	55
3.8	Profiling result of classic edge detector method . . . . .	56
3.9	Canny edge detector method flow chart . . . . .	56
3.10	Profiling result of Canny edge detector method . . . . .	57
3.11	Colour edge detector method flow chart . . . . .	58

## LIST OF FIGURES

---

3.12	Profiling result of colour edge detector method . . . . .	59
3.13	Region-growing segmentation flow chart . . . . .	60
3.14	Profiling result of region-growing segmentation . . . . .	60
3.15	Watershed segmentation flow chart . . . . .	61
3.16	Profiling result of watershed segmentation . . . . .	62
3.17	Rainfalling segmentation flow chart . . . . .	63
3.18	Profiling result of rain-falling segmentation . . . . .	64
3.19	Colour Watershed segmentation flow chart . . . . .	65
3.20	Profiling result of colour watershed segmentation . . . . .	65
3.21	K-means segmentation flow chart . . . . .	66
3.22	Profiling result of K-means segmentation . . . . .	67
3.23	Local K-means segmentation flow chart . . . . .	68
3.24	Profiling result of local K-means segmentation . . . . .	68
3.25	Meanshift segmentation flow chart . . . . .	70
3.26	Profiling result of meanshift segmentation . . . . .	70
3.27	Graph-based segmentation flow chart . . . . .	71
3.28	Profiling result of graph-based segmentation . . . . .	72
3.29	Anisotropic-based segmentation flow chart . . . . .	73
4.1	Example image from Berkeley segmentation database . . . . .	77
4.2	Classic thresholding algorithm parameters' variations example. . .	79
4.3	Relative thresholding algorithm parameters' variations example. .	81
4.4	Classic edge detection algorithm parameters' variations example. .	83
4.5	Canny Edge Detection algorithm parameters' variations example. .	84
4.6	Region Growing algorithm parameters variations example. . . . .	86

## LIST OF FIGURES

---

4.7	Watershed segmentation algorithm parameters variations example.	88
4.8	Processing time example for Watershed segmentation. . . . .	89
4.9	Watershed segmentation parameters variation with no filtering. . .	91
4.10	Processing times for Watershed segmentation with no filtering . .	92
4.11	Rain-falling segmentation algorithm parameters variations example.	93
4.12	Processing time example for Rain-falling segmentation. . . . .	95
4.13	Rain-falling segmentation parameters variation with no filtering. .	96
4.14	Processing times for Rain-falling segmentation with no filtering . .	97
4.15	Colour Watershed segmentation parameters variations example. . .	99
4.16	Processing time example for Colour Watershed segmentation. . . .	100
4.17	K-means segmentation parameters variations example. . . . .	102
4.18	Processing time example for K-means segmentation. . . . .	103
4.19	Mean-shift segmentation parameters variations example. . . . .	105
4.20	Processing time example for Mean-shift segmentation. . . . .	106
4.21	Graph-based segmentation parameters variations example. . . . .	108
4.22	Processing time example for Graph-based segmentation. . . . .	109
4.23	Anisotropic-based segmentation parameters variations example. .	110
4.24	Processing time example for Anisotropic-based segmentation. . . .	111
5.1	An abstract diagram of the FATE evaluation framework . . . . .	121
5.2	Toshiba M200 displaying image number 10 for hand segmentation	132
5.3	Test Images 1 to 12 used for hand-segmentation test . . . . .	133
5.4	The mean differences calculated for each subject . . . . .	135
5.5	The mean differences calculated for each image . . . . .	137
6.1	Mean-shift segmentation extensive search of parameter space results	149



## LIST OF FIGURES

---

6.2	Genetic algorithm operations representation . . . . .	151
6.3	Mean-shift segmentation parameter convergence example. . . . .	160
6.4	Mean-shift segmentation linear trend of the cost function. . . . .	160
6.5	Watershed segmentation linear trend of the cost function. . . . .	161
6.6	Graph-based segmentation linear trend of the cost function. . . . .	162
6.7	Application of Nelder-Mead polishing to GA optimisation. . . . .	165
6.8	The Mean-shift GA evaluation with 100 generation . . . . .	172
6.9	The first 3 Generation of the Mean-shift GA evaluation . . . . .	173
6.10	The first 20 Generation of the Mean-shift GA evaluation . . . . .	174
6.11	Colour Edge Detection evaluation 20 images with and without time factor parameters . . . . .	178
6.12	Colour Edge Detection evaluation 20 images with and without time factor timing . . . . .	179
6.13	Watershed segmentation evaluation 20 images with and without time factor parameters . . . . .	180
6.14	Watershed segmentation evaluation 20 images with and without time factor timing . . . . .	182
6.15	Colour Watershed segmentation evaluation 20 images with and without time factor parameters . . . . .	184
6.16	Colour Watershed segmentation evaluation 20 images with and without time factor parameters . . . . .	185
6.17	Colour Watershed segmentation evaluation 20 images with and without time factor timing . . . . .	186
6.18	K-means segmentation evaluation 20 images with and without time factor parameters . . . . .	187

## LIST OF FIGURES

---

6.19 K-means segmentation evaluation 20 images with and without time factor parameters . . . . .	188
6.20 K-mean segmentation evaluation 20 images with and without time factor timing . . . . .	189
6.21 Mean-shift segmentation evaluation 20 images with and without time factor parameters . . . . .	190
6.22 Mean-shift segmentation evaluation 20 images with and without time factor parameters . . . . .	191
6.23 Mean-shift segmentation evaluation 20 images with and without time factor timing . . . . .	192
6.24 Graph-based segmentation evaluation 20 images with and without time factor parameters . . . . .	194
6.25 Graph-based segmentation evaluation 20 images with and without time factor parameters . . . . .	195
6.26 Graph-based segmentation evaluation 20 images with and without time factor timing . . . . .	196
6.27 Objective evaluation of Mean-shift segmentation on 20 images with and without time factor for <i>radiusS</i> and <i>radiusR</i> parameters . . .	199
6.28 The timing performance of the objective evaluation of Mean-shift segmentation on 20 images with and without time factor . . . . .	201
6.29 The three stage of the segmentation algorithm design with examples of the graph-based and the mean-shift segmentation algorithms	205
7.1 An abstract graph of an improved and combined segmentation and evaluation framework . . . . .	214

## LIST OF FIGURES

---

B.1	Thresholding evaluation 20 images with and without time factor parameters . . . . .	219
B.2	Thresholding evaluation 20 images with and without time factor timing . . . . .	220
B.3	Edge Detection evaluation 20 images with and without time factor parameters . . . . .	222
B.4	Edge Detection evaluation 20 images with and without time factor timing . . . . .	223
B.5	Canny Edge Detection evaluation 20 images with and without time factor parameters . . . . .	224
B.6	Canny Edge Detection evaluation 20 images with and without time factor timing . . . . .	225
B.7	Rain-falling Watershed segmentation evaluation 20 images with and without time factor parameters . . . . .	225
B.8	Rain-falling Watershed segmentation evaluation 20 images with and without time factor timing . . . . .	226
B.9	Anisotropic-based segmentation evaluation 20 images with and without time factor parameters . . . . .	226
B.10	Anisotropic-based segmentation evaluation 20 images with and without time factor timing . . . . .	227
C.1	Grey-scale version of Fig. 4.7 with parameter settings for Watershed segmentation . . . . .	229
C.2	Grey-scale version of Fig. 4.11 with parameter settings for Rain-falling segmentation . . . . .	230

## LIST OF FIGURES

---

C.3	Grey-scale version of Fig. 4.15 with parameter settings for Colour Watershed segmentation . . . . .	231
C.4	Grey-scale version of Fig. 4.17 with parameter settings for K-mean segmentation . . . . .	232
C.5	Grey-scale version of Fig. 4.19 with parameter settings for Mean- shift segmentation . . . . .	233



# 1

## Introduction

All human knowledge begins with intuitions, thence passes to concepts and ends with ideas.

---

Immanuel Kant, Critique of Pure Reason

### 1.1 Image Segmentation

---

Image segmentation is one of the key steps in image analysis for object recognition and scene understanding (Lucchese & Mitra, 2001a). The major target is to recognise homogeneous regions within a scene image as individual objects. Typically, identifying and labelling objects is not performed during the image segmentation phase; this can take place in a later stage. Computer vision and image processing applications range from medical analysis (Pham *et al.*, 2000; Pun *et al.*, 1994; Withey & Koles, 2007) to industrial quality control (Malamas *et al.*, 2003), remote geophysical exploration (Fan *et al.*, 2009; Liu *et al.*, 2009;

Rekik *et al.*, 2007), robot navigation (DeSouza & Kak, 2002), and last but not least military applications (Gonzalez & Woods, 2002). In all these areas, the quality of the final result depends largely on the quality of the segmentation (Paragios & Deriche, 1999).

To highlight the importance of the segmentation stage, see Figure 1.1 which illustrates the processing stages that takes place in typical image understanding/computer vision systems. Segmentation is the last stage before the system processes the data to assign ‘meaning’ to the identified ‘objects’; as such, if the segmentation results are not assigned correctly to the objects in the acquired scene then the computer model of the scene will be totally wrong and any other application that depends on this model (robot movement, mail sorting, medical examination, etc.) will be based on a wrong premise.

The feature extraction stage can encompass a number of sub-processing stages depending on the application and the input/output specification. The features can include luminance or colour features, texture feature, or even motion features for video inputs. The extraction process helps in classifying regions with similar features and as a consequence identifying homogeneous segments in the input image.

Its important to highlight here that the segmentation stage itself by definition does not assign any object meaning to the identified segments in the image, and it does not include any pre-processing filtering or feature extraction operations. However, in practice image segmentation algorithms add some of these operations, especially pre-processing filtering operations. For example, some segmentation algorithms implementations add smoothing or noise reduction filters, these filters can be classified as pre-processing stages and -in the whole image un-

## 1.1 Image Segmentation

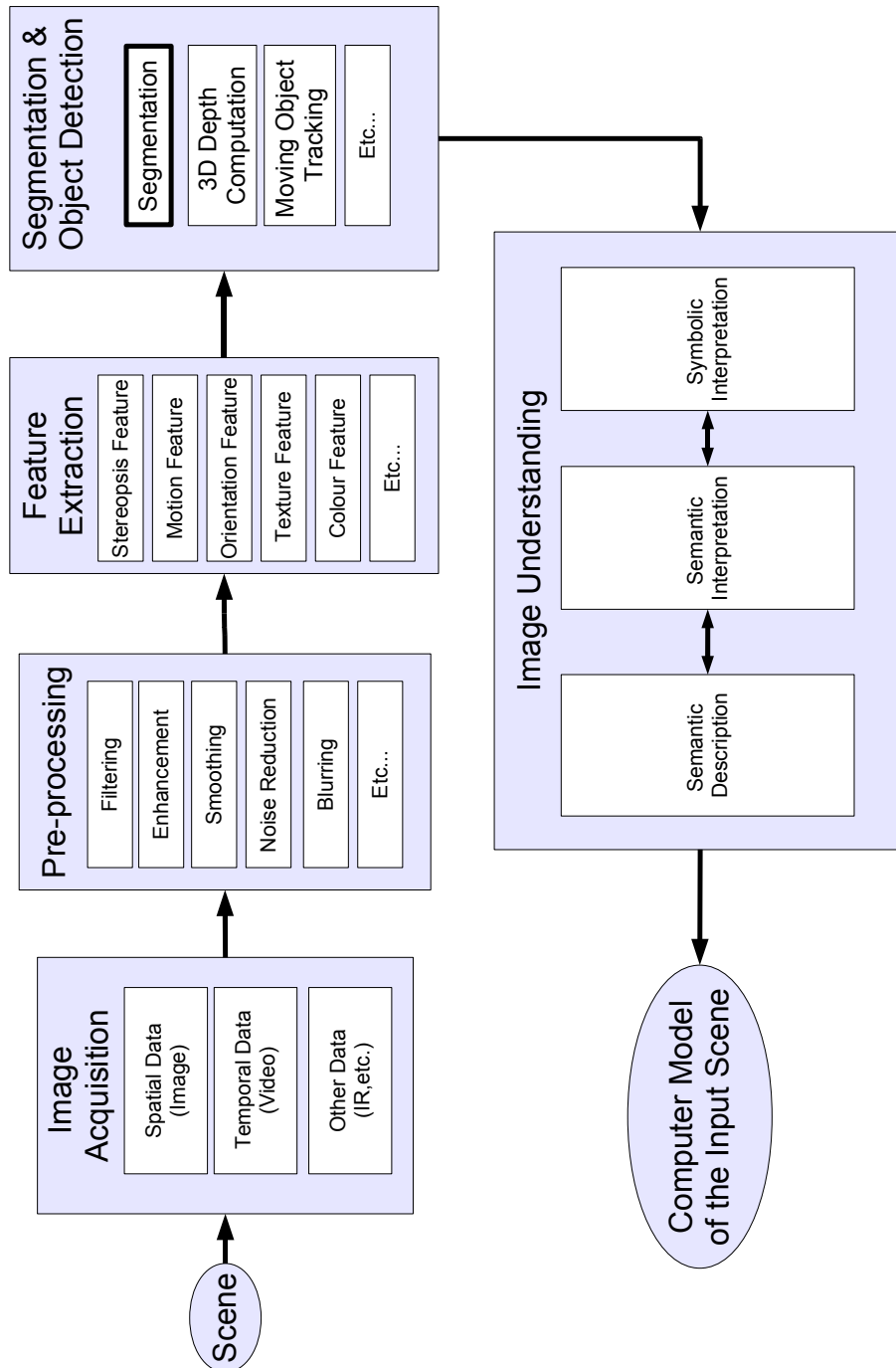


Figure 1.1: Image understanding system flow chart



## 1.2 Motivation of the Research

---

derstanding framework- can even come before the feature extraction stage. This is one particular point that will be explored.

To understand the constraints and conditions that are associated with image segmentation techniques available today, several issues must be examined. One of the common problems is choosing a suitable approach for isolating different objects from the background. A simple example is when greyscale image segmentation techniques do not achieve good results if different objects in the image possess similar grey levels. In practice, researchers employ image enhancement techniques to highlight relevant features of the original image and, hence, simplify the image segmentation process.

## 1.2 Motivation of the Research

---

This research started with the initial aim of improving the current performance of the image segmentation algorithms available in the literature. Watershed segmentation was taken as a first representative algorithm and some initial work was carried to test the performance, looking at parallelising the algorithm and also profiling the algorithm itself. It was concluded that there is a constraining factor and a bottleneck point in the message-passing characteristic of the segmentation algorithm. The segmentation algorithm can always be divided into different elements. However, the fundamental aspect of the algorithm is that it's a global process that will need to look at the image as whole at the end of the processing. Therefore, those distributed/parallelised elements, for the sake of the performance, will need to communicate in the end and overall any advantage found from the parallelisation process is lost.

## 1.2 Motivation of the Research

---

This initial outcome directed the study towards exploring the segmentation field as a whole and to also look for a standardised way of evaluating the segmentation results. Both of these fields have received in depth research in the last three decades at least (Fu & Mui, 1981; Haralick & Shapiro, 1985; Lucchese & Mitra, 2001a; Sahoo *et al.*, 1988; Weszka, 1978; Zhang *et al.*, 2008; Zhang, 1996). However, one of the main points encountered in the initial watershed profiling was exposing the processing stages that are part of the whole algorithm. The other was how the input parameters are actually related to those processing stages. Both of those points: the processing stages (e.g., pre-processing and post-processing enhancement filters in addition to the core segmenting stage) and the parameters significant was not covered by sufficient research in the author's opinion.

In the literature, it is clear that the enhancement technique used has a direct impact on the quality of the resulting segmented image. In practice, the best enhancement techniques are those that make the segmentation process easier by augmenting the object's edges and the image background. Other matters related to image segmentation comprise selecting the right segmentation algorithms for the task at hand, evaluating execution time and the quality of the result and identifying their impact on the computer vision and image analysis systems.

This research aims to provide a framework to evaluate the segmentation algorithms against each other using appropriate scientific and empirical methods, and to develop a framework for a real-time image segmentation processing for computer vision and image-analysis systems. Furthermore, the focus was to highlight the importance of the processing stages in the different segmentation algorithms and their related parameters, and also how they affect the final segmentation

result. This hopefully will lead us to have an alternative formulation of the segmentation problem emphasised by consideration of the processing stages.

## 1.3 Contribution of the Thesis

---

The thesis makes the following contributions:

- Profiling of image segmentation algorithms from different categories according to an image segmentation taxonomy (see Figure 2.1) and highlighting any similarities in the processing stages used among all the algorithms under study.
- Extensive image segmentation algorithm evaluation with different parameter settings, to determine the significance of the parameters on the final results of the segmentation. Also explored is the use of a cluster computing architecture to speed up the evaluation (Al-Muhairi *et al.*, 2007a).
- Employing a genetic algorithm to improve the performance and speed up the parameter evaluation of the image segmentation results; and also using mathematical optimisation procedures to improve the performance further (Al-Muhairi *et al.*, 2007b).
- Illustrate the significance of the parameters on the segmentation performance and quality and how they relate to the “building blocks” of the segmentation algorithms of pre-processing and post-processing filters and the main segmentation stages. Provide some advice on the stages that are more significant for improvement of the overall segmentation performance and quality across different image segmentation categories.

A list of published and submitted publications is included in Appendix [A](#).

## 1.4 Outline of the Thesis

---

The remainder of this thesis is organised as follows:

**Chapter 2** provides a literature review. The first section presents an overview of relevant previous work in the area of image segmentation in the context of a taxonomy of image segmentation algorithms. The image segmentation evaluation research is also reviewed.

**Chapter 3** describes the first step of our image segmentation evaluation. It presents research in profiling image segmentation algorithms in different categories of the image segmentation taxonomy presented. This highlights the different stages involved in image segmentation algorithms and also emphasises any similarities in the stages followed by all the algorithms, even if they belong to different categories of the taxonomy tree.

**Chapter 4** introduces the proposed testing framework, a modified Framework for Algorithm Testing and Evaluation (FATE) testing harness ([Courtney et al., 1997](#)). It also introduces the use of cluster computing to speed up our work.

**Chapter 5** extends the work by adding a Genetic Algorithm (GA) stage to the evaluation framework to optimise the performance.

**Chapter 6** illustrates further the effect of the parameters on the final segmentation results with different segmentation algorithms.

## 1.4 Outline of the Thesis

---

**Chapter 7** concludes this thesis by presenting a summary of the work. It also points to directions for future research.

.....

If a man is offered a fact which goes against his instincts, he will scrutinize it closely, and unless the evidence is overwhelming, he will refuse to believe it. If, on the other hand, he is offered something which affords a reason for acting in accordance to his instincts, he will accept it even on the slightest evidence. The origin of myths is explained in this way.

---

Bertrand Russell (1872 - 1970) — British author,  
mathematician, & philosopher

# 2

## A Review of Research on Image Segmentation and its Evaluation

### 2.1 Introduction

---

Image segmentation is the process of partitioning an image into homogeneous sections with regard to particular characteristics that hopefully relate to real objects in the actual scene (Gonzalez & Woods, 2002). The image segmentation process performance directly influences the performance of the later steps in image-analysis processing, and hence arguably is the most important step in image analysis. Please refer to Figure 1.1 back on page 4 for an illustration. In spite of its significance, segmentation still remains an unsolved problem in the general sense, as it lacks a general mathematical theory (Kurgöllüs & Sankur, 1999). There is almost certainly no one correct segmentation solution acceptable under all psychophysical conditions and to every image condition (Wang

& Suter, 2003). Difficulties in segmentation relate to two main characteristics: its under-constrained nature (LaValle & Hutchinson, 1995); and the deficiency in providing the correct definition of image segmentation (Kurgöllüs & Sankur, 1999). The problem is under-constrained because any segmentation algorithm constraints should be both general enough to satisfy a diversity of problem types, and limited enough to constrain the problem. As such, any general solution for segmentation will tend to be under-constrained to satisfy all application classes.

Furthermore, the disconnect between practical segmentation research and theoretical research that provides mathematical model of segmentation on idealised images lead to the deficiency of a general theory of image segmentation. Actually, as Ciesielski & Udupa (2007) put it:

In fact, there is even no evidence of the use of any definition formally connecting idealized images (infinite objects) with their digital representations (which are finite).

Both of these characteristics relate to the lack of a complete mathematical model to represent image segmentation for general use (although constrained models for specific applications are arguably feasible). Hence, determining the correctness and consistency of the segmentation result of a given scene becomes feasible only for specific tasks.

Perhaps as a result of these deficiencies, many image segmentation methods can be found in the image segmentation literature. Typically, these methods are categorised into (Cheng *et al.*, 2001; Wang & Suter, 2003; Yang & Kang, 2009): region growing (Cramariuc *et al.*, 1997; Ouyang *et al.*, 2009; Rehrmann & Priese, 1998); edge-based (Ma & Manjunath, 1997; Rotem *et al.*, 2007; Saber



---

## 2.2 Taxonomy of Image segmentation methods

---

*et al.*, 1996); clustering (Chen *et al.*, 2008; Comaniciu & Meer, 1997; Zhang & Wang, 2000); histogram thresholding (Kurugollu *et al.*, 2001); Deformable Models approaches (Singh *et al.*, 1998; Xu *et al.*, 2000) (including the Parametric Active Contours usually referred to as Snakes (Kass *et al.*, 1988), and Geometric Active Contours like Level-Set methods (Caselles *et al.*, 1997; Sethian, 1999)); physical model-based (Borràs & Lladós, 2009; Klinker *et al.*, 1990); fuzzy-based approaches (Chamorro-Martínez *et al.*, 2003; Maeda *et al.*, 2007); and neural network methods (Campbell *et al.*, 1996; Kurokawa *et al.*, 2009). These algorithms range from complex methods that use vision models for objects and images to simple and heuristic methods that are application-specific.

## 2.2 Taxonomy of Image segmentation methods

---

To illustrate the different segmentation categories and their classification and relationships, a taxonomy of image-segmentation methods is shown in Figure 2.1. This figure categorises the different algorithms based on their design and the rationale used to segment the images into regions, such as using the image intensity to find object borders and which concepts are used to search for the segments (like region-growing initial seeded pixels). The taxonomy helps us to choose some representative algorithms from each category for evaluation experiments. Not all of the categories have well-defined characteristics or are popular and widely used (which results from many reasons, such as the author’s reputation and the ease of implementation compared to other algorithms). As will be described later, even among different categories, algorithms share some “building blocks” in some of their processing stages. Below the reader will find descriptions and discussions of

## 2.2 Taxonomy of Image segmentation methods

---

each category in the taxonomy tree.

The taxonomy has five main categories: cutting techniques, contouring techniques, learning techniques, physics-based techniques, and hybrid techniques. This taxonomy's categories were adapted from segmentation methods surveys in the literature (Cheng *et al.*, 2001; Wang & Suter, 2003; Yang & Kang, 2009) and combined to cover as much as possible of the algorithms in the literature.

Cutting techniques combine the segmentation methods that are considered elemental and unsophisticated by their design. The methods basically depend on a 'discontinuity' to define the boundaries of the segmentation regions detected. The definition of the discontinuity depends on the design method and implementation. Two of the techniques that are considered cutting techniques in the proposed taxonomy are: thresholding techniques and the edge detection techniques. It's worthy of mention that those techniques are usually used as a basis for other techniques or as one of their processing stages to get the final segmentation result. So for example edge detection and region growing techniques are closely related in their design. However, the approach used is different and as a result the final segmentation results are different.

The main difference is the use of an analogy of growing/evolving seeds to define regions in the image and this is the definition that combines the techniques under the contouring techniques category. This category combines region growing techniques (Köthe, 1995) and level-set techniques (Sethian, 1999).

The most basic implementation of the region-growing techniques start with a pre-set number of seeds with their location in the input image as an input. The seeds are 'grown' on the image space by looking on the neighbouring pixels and deciding to add them to the seed region or not according to a selection criteria

## 2.2 Taxonomy of Image segmentation methods



Figure 2.1: Taxonomy of segmentation methods

## 2.2 Taxonomy of Image segmentation methods

---

different in each implementation. Watershed segmentation techniques (Vincent & Soille, 1991) can be considered as region-growing techniques with a different analogy used to define the regions and the boundaries of the segmented regions, as will be described in more detail later.

Another techniques under this category is the level-set technique (Sethian, 1999; Tsai & Osher, 2003) which use an analogy of an evolving curve models that are grown according to a force function. A force function is computed from the underlying image data. The aim is to make the model to grow towards the object boundaries in the image to produce the final segmentation. Other related techniques are active contours methods usually referred to as snakes (Kass *et al.*, 1988) and Geodesic active contours (Caselles *et al.*, 1997) which aims at connecting level-set and snakes techniques.

Learning techniques combine techniques that use some of the underlying algorithms, that are usually employed in machine-learning and data mining fields, and apply them to the image segmentation problem. The examples in the the taxonomy are: clustering-based techniques and techniques based on algorithms from the Artificial Intelligence (AI) field.

The clustering-based techniques are the most dominant and widely used from the two categories and from observation provide very good segmentation results for the defined segmentation application it's designed to solve. Clustering basically means grouping similar data points into different clusters or groups. And this analogy is used on the image data when each pixel is considered as a data points and the aim is to produce clustered segmented regions, according to a certain similarity measure that corresponds to the objects in the image. Two widely popular implementations are the k-means implementation and the mean-shift

---

## 2.2 Taxonomy of Image segmentation methods

---

implementation.

AI-based techniques apply algorithms such as artificial neural networks (ANNs) (Goldman *et al.*, 2002) and fuzzy-based algorithms (Ito *et al.*, 1995; Lim & Lee, 1990; Maeda *et al.*, 2007) based on fuzzy-set theory. However, compared to the other categories this is a less developed field. Although the main techniques are based on AI algorithms the underlying processing stages are still based on basic segmentation techniques like thresholding and region growing. Another method that can also be mentioned here is self-organising map (SOM) methods (Huang *et al.*, 2002; Yeo *et al.*, 2005) although they are even less popular in this field.

Physics-based techniques (Maxwell & Shafer, 1997) are established on using the physical characteristics of the objects as they are captured into images, and as such tries to exploit these characteristics to help it define the region boundaries of the segmented objects in the image.

The hybrid method group is composed of segmentation techniques that combine one or more of the techniques mentioned above to achieve better segmentation results. There are many techniques that can be classified under this category (and even some of the methods that are mentioned above can be considered hybrid in some of their implementations). This category was added as a completion to the proposed taxonomy and will not be discussed in detail as most of the other categories give enough techniques representation to cover this category.

Furthermore, the boundaries between the categories in this taxonomy are not hard boundaries and as mentioned above there are shared characteristics between the techniques that are classified under different categories, as all of them try to exploit the same image characteristics to get to the ultimate result of defining the region's boundary for the objects in the image. However, the taxonomy gives

---

## 2.2 Taxonomy of Image segmentation methods

a good representation and classification for the image segmentation field overall, and for the purpose of this thesis gives a good basis for further experimentation that will be applied later. Additional detailed description for each category is provided below.

### 2.2.1 Boundary Demarcation Segmentation

#### 2.2.1.1 Thresholding Segmentation

From all the segmentation techniques, thresholding is most likely to be the simplest. Nonetheless, for particular applications it could be the most effective and efficient segmentation technique. In this approach, structures in the image (e.g. objects) are segmented by comparing their intensity value to one or more intensity thresholds. An image can be segmented into two regions by using only one threshold value; but it's common to segment an image into multiple regions using multiple thresholds (Sahoo *et al.*, 1988). Furthermore, thresholds can either be global (constant throughout the image) or local, when different thresholds are chosen based on local characteristics of different areas in the image occur (Sahoo *et al.*, 1988; Weszka, 1978). Figure 2.2 on page 19 illustrates how a threshold is typically determined from the histogram shape of an image. Image histograms represent the colour-pixel distribution for the image; this histogram example represent a grey-scale image where the horizontal axis represent a grey-scale levels (0 for black, 255 for white), while the vertical axis represents the number of pixels that have the given intensity level. The idea is to find a significant discontinuity to take as a threshold; in this example the object has brighter values while the background is dark. In real images, some of the objects can have boundaries' lev-

## 2.2 Taxonomy of Image segmentation methods

---

els beyond the used threshold value, but in practice these objects' boundaries are not considered significant enough for the application at hand, and are sacrificed for practicality and performance of the algorithm.

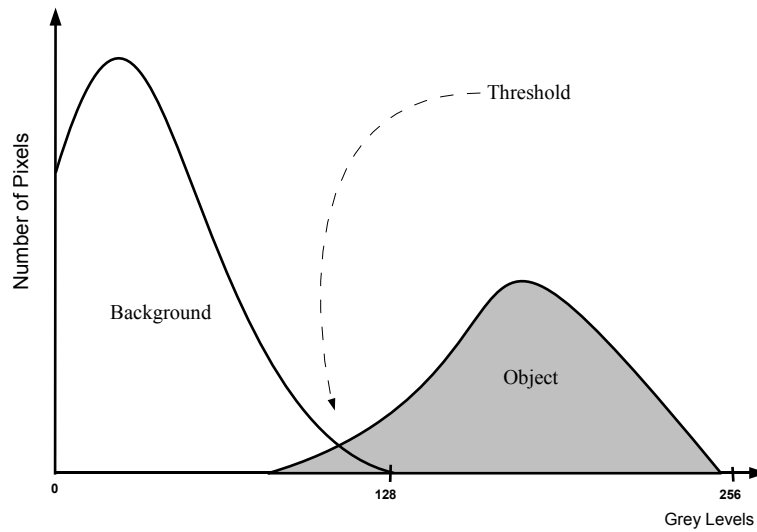


Figure 2.2: Histogram thresholding example

Sezgin and Sanker surveyed 40 different thresholding techniques for image segmentation in their paper (Sezgin & Sankur, 2004). They categorised thresholding techniques according to the image attributes used to apply the segmentation process, such as colour space clustering, histogram shape, object attributes, and spatial correlation among other classification attributes. Although they focus on techniques for grey-scale images in their survey, they conclude that their results could be similar for colour images and recommend further studies in this direction using the same performance measures they suggested. One way to apply thresholding to colour images is to split the colour space (RGB, HSV or HSL (Gonzalez & Woods, 2002)) into components, apply thresholding to each sepa-

## 2.2 Taxonomy of Image segmentation methods

---

rately and then combine them again. One such example is suggested by [Yang & Tsai \(1996\)](#). In their paper, thresholding is applied to each colour plane of the pre-partitioned input colour image using non-overlapping blocks of the image. The thresholding process produces two representative RGB colour vectors for each block. For each block, the single-spectral results are obtained by computing the difference between these two vectors. Yang & Tsai found this method can be faster than other methods such as the Karhunen-Loève expansion or vector median approaches which are also applicable. HSV and HSL have some advantage over RGB as they represent perceptual colour relationships more accurately and, as such, are closer to how humans perceive colours ([Busin \*et al.\*, 2008](#)).

In another recent example, [Pham \*et al.\* \(2007\)](#) used the CMYK colour space thresholding technique in a medical application and achieved higher performance. The benefit of CMYK in this specific application, immunohistochemistry (IHC), stems from using a Yellow channel image analysis method based on a CMYK colour model that improved sensitivity for IHC evaluation compared to other colour models. This use of different colour model to enhance the segmentation results is an improvement suggested in many other segmentation algorithms. Some have an advantage for specific applications, such as the example of using CMYK for IHC, or are suggested for use for general applications, such as the mean-shift algorithm using the LUV colour space that will be described in greater detail later.

[Sezgin & Sankur \(2004\)](#) reached a similar conclusion to most other papers in this research field: even in a single application domain, there is no single algorithm that is successful for all image types. [Figure 2.3](#) provides a few examples of thresholding techniques to illustrate that thresholding can provide some good

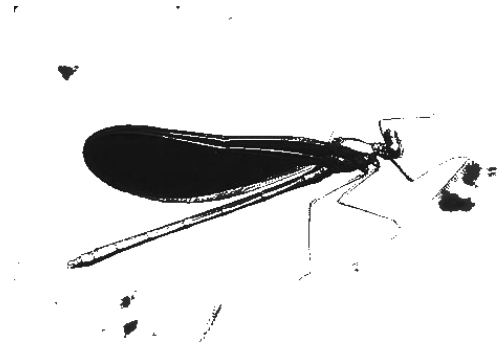


## 2.2 Taxonomy of Image segmentation methods

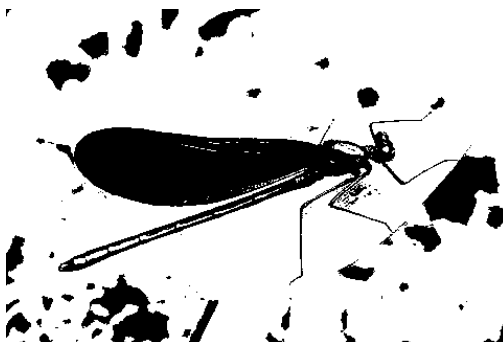
results in some cases. The relative thresholding illustrated in the Figure differ in the fact that the threshold values are used as percentages rather than absolute values to represent the pixel values like standard thresholding, which is also illustrated. Optimal thresholding aims to find a statistically optimal threshold value based on the image “structure”. The optimal thresholding represent the image histogram into an approximation of two or more normal distributions (where distributions represent object(s) or background) and iteratively calculate an optimal threshold value corresponding to the minimum probability between the maxima of the defined distributions (Ridler & Calvard, 1978; Sonka *et al.*, 1993).



(a) The original input image



(b) Output from standard thresholding method



(c) Output of relative thresholding method



(d) Output of optimal thresholding method

Figure 2.3: Thresholding example

---

## 2.2 Taxonomy of Image segmentation methods

---

However, it is important to note that thresholding (compared with the other segmentation techniques that will be discussed below) is often too simplistic in that an object does not necessarily exhibit contrast relative to its surroundings. One possible improvement, introduced in the literature, is to use multiple thresholds (Reddi *et al.*, 1984; Sahoo *et al.*, 1988). Also thresholding produces very basic segmentation results, typically a “binary” segmentation: segmented object region, and background region, where as this can be sufficient in certain applications, most of the applications will need higher discriminating segmentation results (as will be provided by examples below in other segmentation algorithms).

### 2.2.1.2 Edge Detection Segmentation

Edge-based segmentation is one of the earliest image segmentation techniques (Roberts, 1963) and still has significance. Edges are interpreted as discontinuities and as objects’ outer boundaries in the image segmentation process. A variety of edge detecting algorithms such as gradient or the Canny edge detector (Canny, 1986; Koschan, 1995). However, it’s important to note here that the resultant images cannot be used directly as a segmentation: additional processing stages are needed to join edges into edge chains, as edge chains are a better representation of the objects’ boundaries in the image. The goal is to reach the best possible object segmentation, grouping local edges to reach a resultant image, where all the edges correspond correctly to existing object boundaries in the image. Figure 2.4 illustrates an example of edges in grey-scale images; the same idea applies to colour images. However, colour features found in colour images can provide additional information compared to grey-scale images. Colour edges can be defined by two techniques: (1) computing the edges of each of the colour

## 2.2 Taxonomy of Image segmentation methods

---

spaces separately and then combining them according to a pre-defined criteria, or (2) compute the edges in the vector space where colour information of the image is defined as colour vectors. Many criteria and definitions for each technique were introduced in the literature (Koschan & Abidi, 2005; Lucchese & Mitra, 2001a).

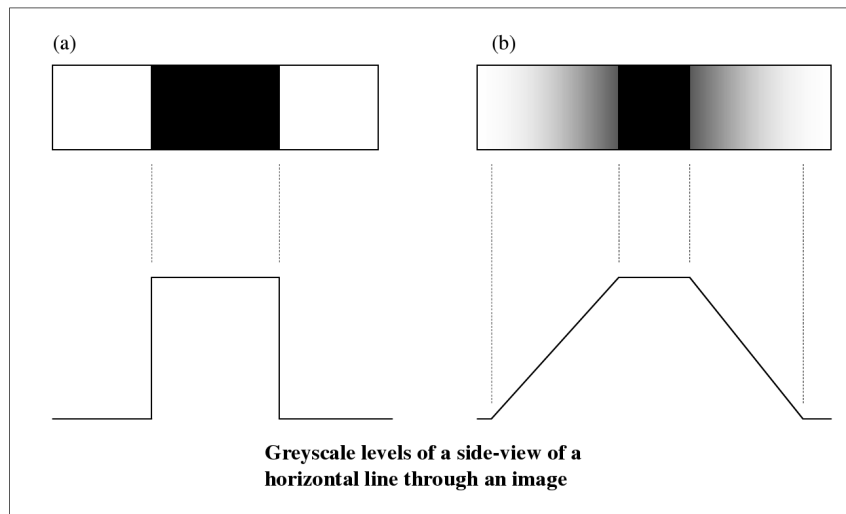


Figure 2.4: Example of edges in a grey-scale image: (a) Example of an ideal edge (b) Example of a ramp edge (adapted from Gonzalez & Woods (2002))

Classic Edge detection methods usually process the input image to create a gradient magnitude, then use this in an maxima search stage to find edges in the gradient. This can be either a thresholding method (similar to what was discussed above) or a non-maxima suppression stage, which was introduced by Canny (1986), to suppress all values along the line of the gradient that are not peak values.

An example output from a standard edge detection using Sobel operator to compute the gradient is illustrated in Figure 2.5. The Figure also illustrates other edge detection techniques, such as Canny edge detection and also a colour edge

## 2.2 Taxonomy of Image segmentation methods

---

detector suggested by [Gevers & Stokman \(2003a\)](#).

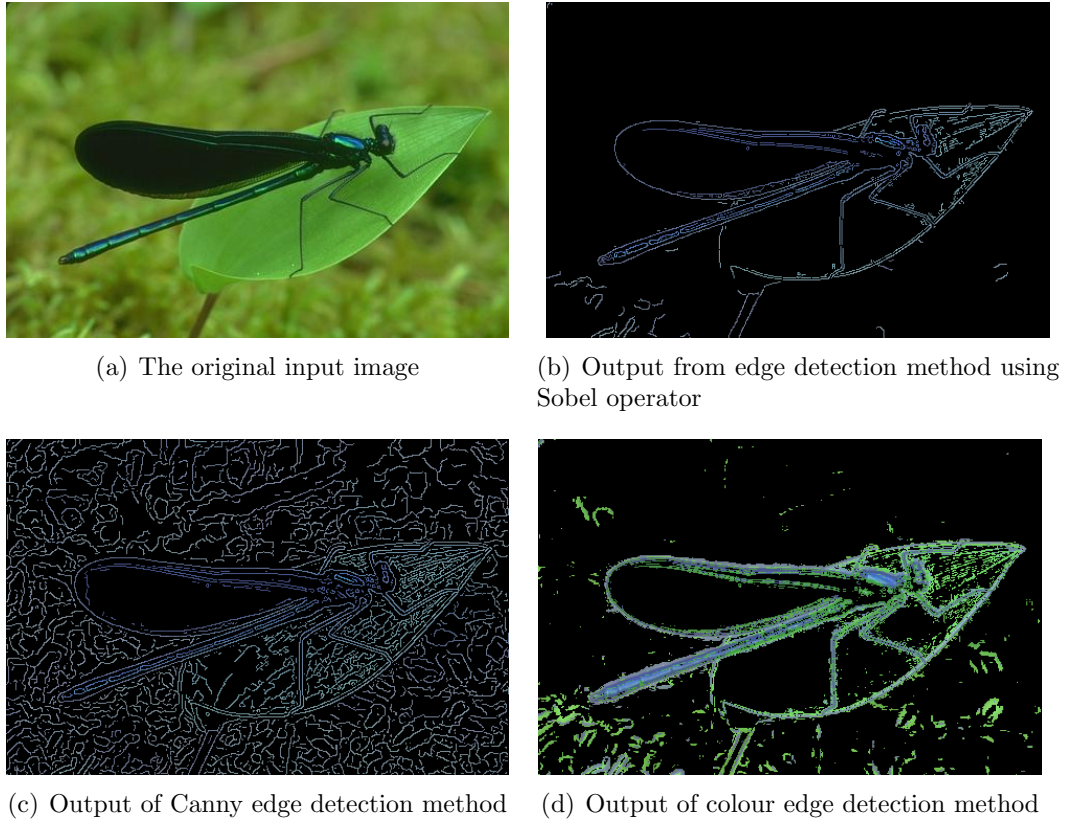


Figure 2.5: Edge detection examples

Classic edge detection usually employs a gradient function that uses Sobel kernel (other operators include Roberts, Prewitt operators to find the image gradient ([Koschan, 1995](#))). Canny edge detection, unlike classical edge detection techniques, apply an additional pre-processing noise reduction filter (e.g. smoothing filter) based on a Gaussian function then applies four gradient functions: vertical, horizontal, and two diagonal edges. And Also the algorithm uses a non-maxima suppression stage to find the edges. In other words, this stage tries to find local maxima to define the edges and suppress the other parts of the

image gradient.

### 2.2.2 Contouring Techniques

#### 2.2.2.1 Region-growing Segmentation

Region-growing techniques are widely established as an efficient approach for image segmentation (Köthe, 1995). The basic approach is to start from a region seed (typically one or more pixels) that is deemed to be inside the object to be segmented. Then the algorithm starts evaluating neighbouring pixels to resolve, according to certain criteria (like pixel intensity, colour, texture, etc.), if they should also be considered part of the object. Consequently, if pixels are found to meet these criteria, they are added to the region and the process continues as long as new pixels are added to the region.

In addition to different criteria used to decide whether a the pixel is included in a region, region-growing algorithms also vary depending on the traversing strategies for neighbouring pixels and the connectivity type used to determine neighbours. An example output from region-growing segmentation is illustrated in Figure 2.6. In this example, the seeding points are grown to the adjacent pixels according to a colour homogeneity criteria and a threshold value, and the technique uses 4-connected neighbourhoods to grow the seeding points.

#### 2.2.2.2 Level-Set Segmentation

Level-set methods, as proposed by Sethian (1999), are founded on the idea of transferring a dynamic problem into a higher dimension and consequently dealing with it as a static problem. This new dimension represents the process dynamics

## 2.2 Taxonomy of Image segmentation methods

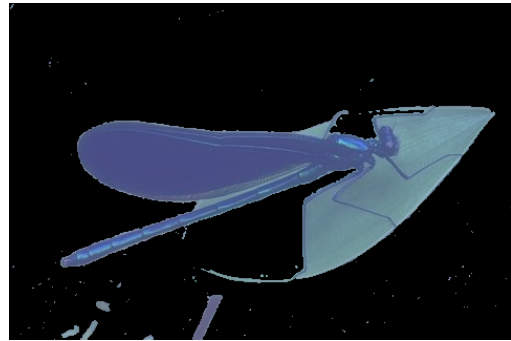
---



(a) The original input image



(b) The output of region growing segmentation method



(c) The segments overlayed on the original image to illustrate the final result

Figure 2.6: Region growing example

for the problem at hand.

Theoretically, level-set methods employ a curve in  $(x, y)$ -coordinates, representing an expanding contour that defines an object in the segmented image (see Figure 2.7 (a)). This curve is built into the  $(x, y)$ -plane of a cone-shaped surface in a three-dimensional coordinate system (see Figure 2.7 (b)). Now this surface is defined in the algorithm to cut the  $(x, y)$ -plane exactly where the curve is located. This cone-shape represents the level-set function that by its definition accepts any point in the plane and return its height as an output. The original curve is called the Zero-Level-Set and is found by cutting the surface at the  $(x, y)$ -plane (repre-

## 2.2 Taxonomy of Image segmentation methods

---

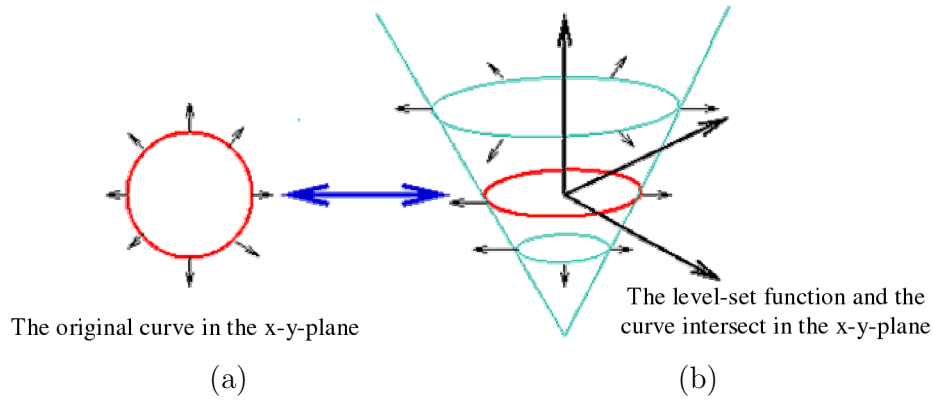


Figure 2.7: Level-Set technique illustration [Sethian \(1999\)](#)

sented in Figure 2.7 (b)) ([Sethian, 1999](#)). In simple terms, the algorithm defines an artificial closed contour that can expand or contract over time, within the input image, to conform to a pre-defined criteria (i.e. image features). Active contours and surfaces, energy-Minimisation models, and front-propagation methods based on level-sets are also related to this category ([McInerney & Terzopoulos, 1996](#); [Xu \*et al.\*, 2000](#)).

The dynamic curve described in the Level-Set function can be moved and deformed freely to fit the objects to be segmented in the input image. A function is used to regulate the speed and the deformation degree at any point of the curve. Typically, features of the curve determine the speed function (e.g. the curvature or gradient). By using these features, level sets are applied to image segmentation. [Tsai & Osher \(2003\)](#) gives a wider overview of the use of the level-set technique in the image-processing field in general and for image segmentation in particular.

---

## 2.2 Taxonomy of Image segmentation methods

### 2.2.2.3 Watershed Segmentation

The Watershed transformation is popular in the fields of image processing and morphology (Vincent & Soille, 1991). Although this technique has similarities to region-growing image segmentation, it has a different analogy to region growing that makes it reside in a class by itself.

In their seminal paper, Vincent & Soille (1991) were one of the first to propose using a watershed as a mathematical morphology tool to produce a fast and flexible algorithm that segments greyscale images. The concept of watershed for contour detection was originally introduced by Beucher & Lantuejoul (1979) but was widely popularised by Vincent & Soille (1991) with their new implementation. The algorithm is based on an immersion process analogy which views the pixels in the input image as an elevation terrain, yielding 'mountains' and 'valleys'. In practice, the algorithm is divided into a flooding phase and a sorting phase. Segmentation involves symbolic water pouring (unlike the rain-falling approach, which will be discussed below) over the elevation terrain to form catchment basins. Next the watershed process classifies areas with similar basin levels as being in the same segmentation region.

Meijster & Roerdink (1995) suggested an alternative algorithm which consists of 3 stages: (1) Transforming the image into a labelled graph; (2) Flooding (computing the watershed); (3) Transforming the graph back into an image.

### 2.2.2.4 Rain-falling Watershed approach

This approach is basically a watershed algorithm based on a rain-falling analogy rather than the immersion analogy mentioned earlier. The rule of assigning labels



---

## 2.2 Taxonomy of Image segmentation methods

to watershed regions is derived from physics: a raindrop falling on a surface will, due to gravity, move downward to the nearest neighbouring location. In (Moga & Gabbouj, 1997) and (Moga *et al.*, 1995) the segmentation takes place in two stages: (1) labelling plateaux of minima; (2) from each pixel a raindrop slides on the steepest slope towards the nearest labelled minimum.

Over-segmentation, a familiar phenomenon in watershed segmentation, occurs because every local minimum, even if tiny and insignificant, forms its own minima. The De Smet *et al.* rain-falling implementation (De Smet & Pires, 2000) tries to avoid this problem by introducing a simple pre-processing procedure in which the image is modified to remove minima that are too shallow.

De Smet & Pires (2000) illustrate clearly that their implementation is an excellent candidate for use in practical applications where rapid performance and/or efficient memory usage is needed.

Figure 2.8 gives a good idea of what to expect from the watershed algorithm. The colour watershed used here is based on Alvarado's work (Alvarado, 2004), which employs an adjacency graph to merge similar colour regions. It is adapted from a region merging algorithm (Haris *et al.*, 1998) but modified for colour images, where the merging is performed according to a vector that represents an RGB pixel.

### 2.2.3 Learning-Based Techniques

#### 2.2.3.1 Clustering-Based Segmentation

Clustering is a process of substituting sets of data by representative clusters, each cluster representing a related collection of data points. Hence, image segmenta-

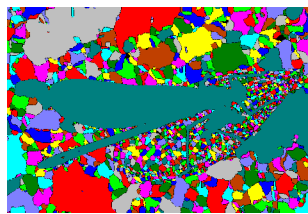
## 2.2 Taxonomy of Image segmentation methods



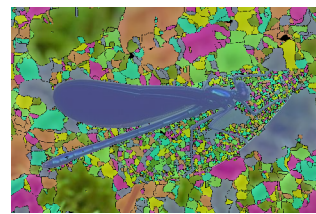
(a) the original input image



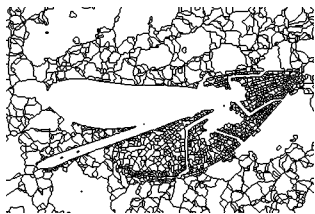
(b) Watershed Segmentation Output



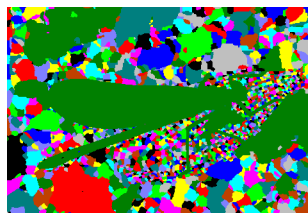
(c) Watershed regions (each in different colour)



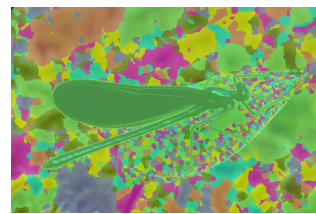
(d) Watershed regions overlaid over the original image



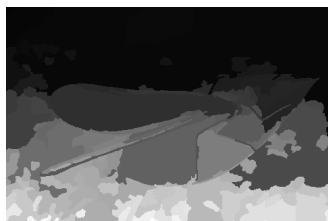
(e) Rain-falling Segmentation Output



(f) Rain-falling regions (each in different colour)



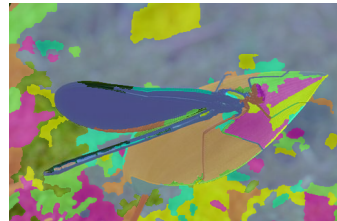
(g) Rain-falling regions overlaid over the original image



(h) Color watershed Segmentation Output



(i) Color watershed regions (each in different colour)



(j) Color watershed regions overlaid over the original image

Figure 2.8: Watershed examples

---

## 2.2 Taxonomy of Image segmentation methods

---

tion, by classifying pixels that possess similar texture and/or colour, can naturally implement clustering. Therefore, if each object in an image has a homogeneous and constant colour property and the pixels of the image are mapped using a certain colour space, then it's expected that clusters of homogenous colours will represent the objects in the image. This makes clustering-based segmentation using the feature-space, the colour-space in this case, more than the spatial-space of the image (Lucchese & Mitra, 1999). One of the requirement for clustering-based segmentation is to define the number of the clusters to be computed. This is usually left to the user as an input parameter, and as such provides a dilemma of choice for the user: to decide what the best value will be depending on the application at hand and the type of the input image. Further discussion will be provided later on the topic of the input parameters.

Clustering techniques are typically classified into two categories: hierarchy-based and partition-based clustering (Jain *et al.*, 1999); under each category we can find a vast number of sub-categories that contains different clustering algorithms. Hierarchical clustering classifies the clusters themselves into groups, either by merging or splitting clusters, repeating the process at different levels to achieve finally a tree of clusters classified according to their relation to each other. Partition-based techniques form clusters by optimising a clustering criterion, derived from a representation function that expresses the best clustering result of the dataset. The K-means algorithm is the most commonly used partition-based clustering method (Marroquin & Girosi, 1993a), in which the representation function is the squared distance of the dataset point from its nearest cluster centre. The K-means algorithm finds clusters by iteratively computing a mean intensity for each class and then classifying each pixel in the input image to the class with

---

## 2.2 Taxonomy of Image segmentation methods

---

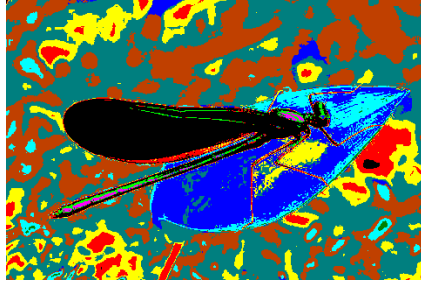
the closest mean until all pixels are classified. See Figure 2.9 for some clustering-based segmentation examples.

Clustering-based techniques are considered ideal for applications where each object to be segmented in the input image has a relatively distinct homogenous colour for its surface and close objects are colour distinct from each other, as they can be easily extended to handle the complexity of the colour-space of the images. However, for other cases where the spatial information needs to be considered in addition to the feature space, adaptive clustering-based segmentation techniques were proposed in the literature with varying degree of success, although this also results in relatively longer execution times (Chen *et al.*, 1998; Pappas, 1992).

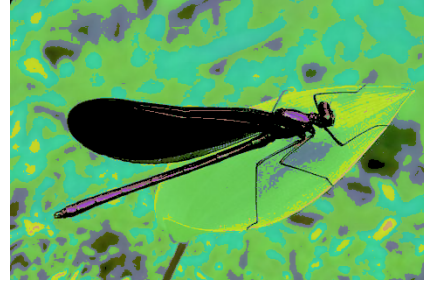
### 2.2.3.2 Artificial Neural Network based Segmentation

Artificial Neural Networks (ANNs) are a type of Artificial Intelligence (AI) algorithm (Luger, 2004; Russell & Norvig, 2002) that is commonly used in the pattern recognition research field (Ripley & Hjort, 1995). Moreover, ANNs are inherently non-linear in nature. Thus, in principle ANNs can model general and complex functions. However, ANNs also usually require a long time to train, which would lower the execution performance if the training was carried out online. The ANNs can only be used to segment images of the same type it was trained on. Additionally, it is not possible to extract the complex function it models. Therefore, this type of solution for image segmentation can become too application-centric and not applicable for general use.

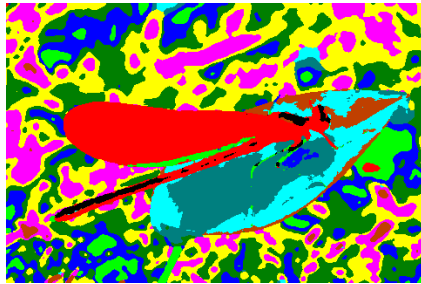
## 2.2 Taxonomy of Image segmentation methods



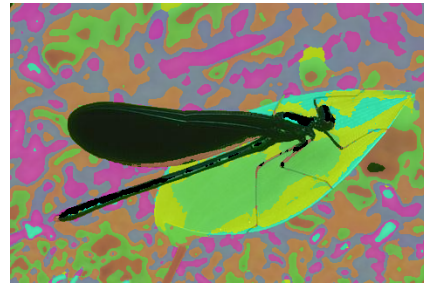
(a) Local K-mean segmentation regions (each in different colour)



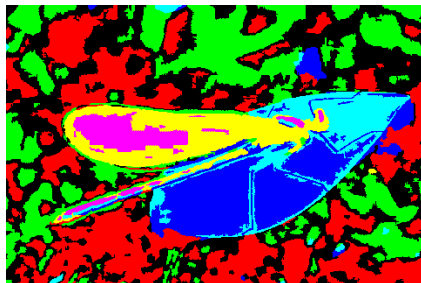
(b) Local K-mean segmentation regions overlaid over the original image



(c) K-mean segmentation regions (each in different colour)



(d) K-mean segmentation regions overlaid over the original image



(e) Classic Mean-shift segmentation regions (each in different colour)



(f) Classic Mean-shift segmentation regions overlaid over the original image



(g) Mean-shift segmentation regions (each in different colour)



(h) Mean-shift segmentation regions overlaid over the original image

Figure 2.9: Clustering-based segmentation examples

### 2.2.4 Physics Based Techniques

Natural lighting of objects in the world, such as highlights, shadowing, reflections and others, greatly affect the results of image segmentation algorithms. The reason is that consistently coloured objects in the world affected by these phenomena result in apparent changes in their coloured surfaces; hence those algorithms are expected to return over-segmented results.

One way proposed to prevent this negative aspect (Shafer, 1985; Shen *et al.*, 2008) is to apply physical models of how light interacts with objects. The underlying algorithms to find the segments do not radically differ from those of the techniques previously discussed; the main distinction is taking into consideration a physical model of the light properties affecting coloured objects.

Coloured objects may be divided into three main categories: optically homogeneous dielectrics (such as crystal and glass), optically inhomogeneous dielectrics (such as textile and paper), and metal. Klinker *et al.* (1987, 1988, 1990) use this categorisation to develop an algorithm (with either split-and-merge or region-growing as an underlying algorithm). Klinker's segmentation algorithm makes some optical assumptions relating to an object's colours, shading, and highlights and tries to justify them as the segments' shapes.

Other researchers also proposed other approaches to employ lighting and colour properties to achieve better image segmentation results (Gevers, 2002; Gevers & Stokman, 2003b; Kravtchenko & Little, 1999; Lucchese & Mitra, 1999, 2001b; Wesolkowski *et al.*, 2000).

### 2.3 Improving segmentation computational performance with parallelisation

---

In trying to improve the computational performance and improve the processing time, some image segmentation research employs parallel processing to handle the immense computation load (Bader *et al.*, 1996). Such speed-ups from parallelisation are mainly attractive for real-time segmentation needs, such as in video coding or in industrial inspection.

Meijster & Roerdink (1995) in their approach divides the segmentation into three stages: 1) transform the image into a directed valued graph, then 2) compute the watershed of the directed graph, and finally 3) transform the labeled graph back into a labeled segmented image. This makes it possible to parallelise all of the stages, though they prove that in practice it is worthwhile to parallelise only the second stage. This is because the local nature of the second stage that makes it possible to perform it in parallel, in contrast to the original algorithm. This property, as we will see in the other techniques mentioned in this thesis, is always true for this stage of the process, even though each technique tackles the problem by forming different solutions.

In the parallel implementation of Nicolescu *et al.* (1999) the whole image is distributed to all the processors. The rationale here is that if only regions of the images are distributed to each processor, then some of the processors will need some data that are stored on another processor and consequently could generate high inter-communication between the processors. Each node will compute the watershed only for the region of the image assigned to it, and then the result

### **2.3 Improving segmentation computational performance with parallelisation**

---

is sent back to a “manager” processor which combines the result. Additionally, this approach proves that, to compute the watershed correctly, the assigned sub-images should have overlapping sections that cover all the region-growing seeds in the sub-images assigned to neighbour processors.

What is interesting in this research is that the implementation is performed with two separate message-passing standards: PVM (Parallel Virtual Machine) (Geist *et al.*, 1994) and MPI (Message Passing Interface) (Snir *et al.*, 1995). PVM, developed at Oak Ridge National Lab, is a portable heterogeneous message-passing system. It offer tools for process spawning, execution on multiple architectures, and inter-process communication. MPI has come into the mainstream more recently than PVM, but it is a mature standard that has been available for several years. Argonne National Lab developed this standard as a public domain implementation and it is available for nearly all the major computing architectures. Nicolescu *et al.* (1999) conclude in their research that the speed-up is disappointing and below expectation for both implementations. The MPI implementation had slightly better speed-up result than the PVM because it was implemented on a network with higher bandwidth. They conclude that distributed memory systems may not be the best choice to implement the watershed algorithm, and that the gathering of the sub-results has to be made more efficient.

For rain-falling segmentation approach, Moga & Gabbouj (1997) (and (Moga *et al.*, 1995)) confirmed the advantage of this approach in term of anticipated efficiency. It is a local method because each raindrop follows its own way regardless of its neighbouring raindrops. This results in less message-passing between the processors in a parallel implementation compared to the immersion-approach which is highly global in nature. This was implemented with both PVM and



MPI, but the two implementations were not compared with each other by the authors. One of the reasons given is that the underlying computer hardware used by each implementation was different. Even so, the results showed PVM to be slightly faster than MPI for the same number of processing nodes. Having said that, a more exhaustive evaluation needs to be carried out as the example given was only on two images.

Inherently by their design, ANN-based segmentation processing is carried out simultaneously on different computational elements. Hence, it is practical to implement it on parallel or distributed systems, which could allow for faster execution performance and constitute an appropriate solution for real-time applications. However, the disadvantage of message passing overheads can negate the performance advantage of individual elements, which is the same disadvantage that all the parallel/distributed segmentation implementations face. A fundamental aspect is that at least one part of the segmentation process is a global process. This problem is not overcome completely by current research and further research is still needed.

## 2.4 Segmentation Evaluation

---

The evaluation of image segmentation techniques is important in determining which characteristics of which algorithms work well, and in what circumstances. Although there is a fair amount of research activity that can be found in the literature, this field has not reached maturity, in the sense of reliability and consistency. As such, there are a number of ways to categorise segmentation evaluation. One way to do this is to categorise evaluation methods into *supervised* or *unsupervised*

based on whether or not *a priori* information is available. Unsupervised methods are defined as methods that don't require *a priori* information (i.e. reference or ground-truth image), whereas supervised methods require a reference image to complete the evaluation.

### 2.4.1 Segmentation evaluation categorisation according to the use of a priori information

#### 2.4.1.1 Unsupervised Segmentation Evaluation

Haralick & Shapiro (1985) proposed some guidelines for qualitative evaluation techniques. Several researchers in the field also developed quantitative evaluation techniques: a "Busyness measure" was introduced by Weszka and Rosenfeld (Weszka & Rosenfeld, 1978; Weszka, 1978), as well as classification error to evaluate the performance of segmentation algorithms. Sahoo *et al.* (1988) combined a shape measure with a region uniformity measure.

Qualitative measures are applied by computing the feature differences between an image segmented by an algorithm and an ideally segmented (e.g. hand-segmented) image. Zhang (1996, 1997) has extensively detailed this approach in his research. This approach can use any object (e.g. shape) features, eccentricity of the shapes in the image, the texture of the shapes, and so on.

Shape features (such as orientation, elongation, area, and circularity) of segmented objects were used for comparison of segmentation algorithms by Yang *et al.* (1995) using the accuracy of the object feature measurement from the segmented image with respect to the reference image as a discrepancy measure, termed Ultimate Measure of Accuracy (UMA) as a reference to the ultimate goal

of the segmentation process.

### 2.4.1.2 Supervised Segmentation Evaluation

Supervised segmentation evaluation typically uses a reference (or a "ground-truth" image) to measure the "error" in the segmentation results. Thus, the segmentation algorithm performance is measured according to the detected dissimilarities between the segmentation output and the ground-truth. *Yang et al. (1995)* propose error probability as a straightforward measure for the quality of supervised segmentation.

A specialised methodology was proposed by *Hoover et al. (1994, 1996)* to evaluate range-image segmentation methods. This methodology was implemented into an evaluation framework to compare range-image segmentation techniques. Input images were acquired with a laser range finder and a triangulation technique. First the user of the framework takes a specially developed tool to create a reference image segmentation (ground-truth) by assigning three distinct labels to the image regions: cross edge pixels, shadow pixels and noise pixels. Then the output of the segmentation algorithm under evaluation is compared with the ground truth and a region mapping is created according to the following criteria: correct detection, under-segmentation and over-segmentation, noise and missed.

### 2.4.1.3 Commentary

Figure 2.10 illustrates a new way suggested by the author to represent the previous evaluation methods categorisation as points in a space spanned by two axes, one representing the relation between being (Quantitative - Qualitative) and the other representing (Objective - Subjective). Subjective methods use a human evaluator

## 2.4 Segmentation Evaluation

to assess the segmentation results. The objective evaluation, on the other hand, should not use any subjective opinion in the evaluation. Quantitative evaluation research depends on defining a testing data-set for evaluation and a more general evaluation measurement. As a consequence, the conclusion that arises should be applicable to a wider set of other untested results. Qualitative evaluation, on the other hand, is applied to a limited set of images and most likely is subjective in nature. The scale itself is more representative than being a measurement for the differences between the methods. So methods that lay in the same quarter of the categorisations have the same definition like methods used by [Yang \*et al.\* \(1995\)](#) being subjective and quantitative and so lying in the top left quarter of the chart. So from this we can see that both the [Yang \*et al.\* \(1995\)](#) method and methods used by [Haralick & Shapiro \(1985\)](#) are subjective methods on the scale. However [Yang \*et al.\* \(1995\)](#) is more quantitative.

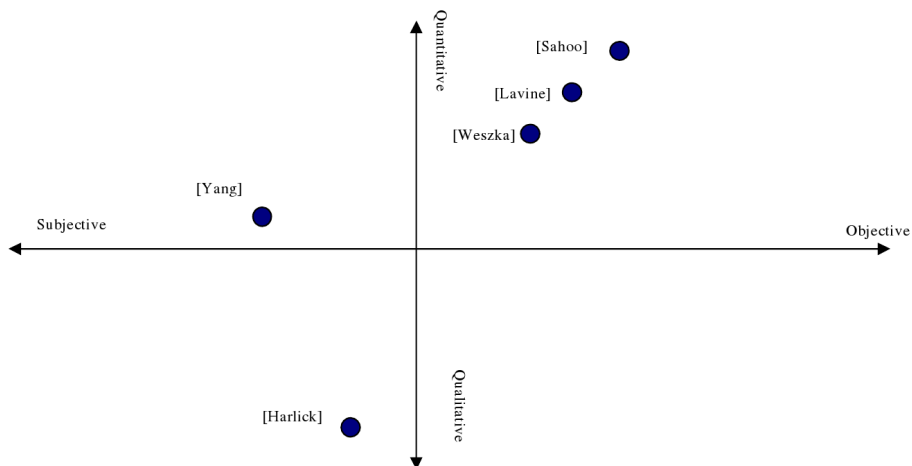


Figure 2.10: Evaluation methods categorisation using (Quantitative-Qualitative)-(Subjective-Objective) axis

On the other hand, methods surveyed by [Weszka \(1978\)](#), [Sahoo \*et al.\* \(1988\)](#), and the [Levine & Nazif \(1985\)](#) approach are quantitative evaluation methods that can also be defined as objective methods. Their location on the chart is just how they are related to each other on the scale. This type of categorisation is harder to apply to different methods as some methods can cross the categories. A method that lies in one of the quadrants can be argued to have some characteristics of methods that lie in another quadrant. It can also be noticed that some parts of this chart are not possible to fill-in such as finding a method that is both objective and qualitative. This section gives one example of segmentation evaluation categorisation. Evaluation method categories suggested by [Zhang \(1996\)](#) are more recent compared to the categorisation defined above, which gives a more streamlined and easier to define categories, as will be discussed in the next Section.

### 2.4.2 Zhang Segmentation Evaluation Categorisation

Zhang's methodology ([Zhang, 1996](#)) of categorising evaluation methods is not that much different from the categorisation explained above; the difference is in the way they are named and some slight adjustment in the categorisation process. Zhang categorises evaluation methods into: analysis methods, empirical goodness methods and empirical discrepancy methods. The analysis methods consider the segmentation algorithm directly. The empirical goodness methods evaluate the output segmented image to indirectly assess the segmentation algorithm performance. On the other hand empirical discrepancy methods compare the output segmented image to a reference image (ground-truth) and use discrepancies to

assess the segmentation algorithm's performance. The following subsections provide more information about each category.

### 2.4.2.1 Analytical Methods

Analytical evaluation methods investigate the segmentation algorithm by evaluating their properties independently of the segmentation output of each algorithm under consideration. The evaluation process can be divided into two categories: 1) Evaluating the algorithmic properties of the algorithms such as processing strategies, processing complexity, and implementation details, and 2) Evaluating the algorithms use of resources, computational performance and processing timing independent of any consideration to the segmentation quality. This type of evaluation works with some specific models or attractive properties of the algorithms. Additionally, this type of evaluation is represented by the type and amount of *a priori* knowledge that has been incorporated into different segmentation algorithms. No significant studies can be found in the literature that cover this area for a wide number of algorithms, so Chapter 3 provides such a study for a number of chosen segmentation algorithms from the thesis taxonomy.

### 2.4.2.2 Empirical Goodness Methods

This group of methods focuses on the quality of the resulting segmented images to evaluate an algorithm's performance. Most of the quality measures are established according to what conditions define an ideal segmentation by human perception (Huang & Dom, 1995; Zhang *et al.*, 2003). Intra-region uniformity, inter-region uniformity and region shape are examples of goodness measures (Zhang, 2001). These methods do not require any *a priori* information regarding

the correct segmentation reference to compute the measure and differentiate the various segmentation algorithms. Hence, these type of methods can serve as an online evaluation, as there is no need for a correctly segmented reference image. These are also known as unsupervised methods described above, [Zhang \*et al.\* \(2008\)](#) have done a very recent and rigorous survey in the area of unsupervised image segmentation evaluation methods.

### 2.4.2.3 Empirical Discrepancy Methods

These methods utilise disparity between a segmented image and an ideal segmented image (reference image) to assess the segmentation algorithm's performance. In other words, the methods measure the degree of similarity between the segmented image and the corresponding ground-truth image. This can be measured in a number of ways, like the number of mis-segmented pixels, or the location of the mis-segmented pixels to give some examples. ([Droogenbroeck & Barnich, 2005](#); [Kubassova \*et al.\*, 2006](#); [Martin \*et al.\*, 2001](#); [Polak \*et al.\*, 2008](#)).

The input image is used to obtain both the resultant segmented image and the reference image. There are two cases for the test image: natural images and synthetic images. For natural images, the reference images are manually created, while the true segmentation can be automatically generated for synthetic images.

## 2.5 Summary

---

This chapter gave a review of the current colour segmentation algorithms in the research field and related evaluation methods needed to benchmark these algorithms. The important fact, that was mentioned in the introduction, and

needs to be repeated here is: there is still no universal and widely accepted theory on colour image segmentation. As a consequence, image segmentation evaluation is in the same state as existed before. Most algorithms in the field are tailored for a particular application and assert certain hypotheses that are needed for an algorithm to work optimally. The solution should be that a segmentation method for general application should be available. However, there is still no one clear answer. It still depends on a number of factors that need to be defined before proceeding: the image type, the application and what type of output is required.

An image segmentation problem is basically one of psychophysical perception and it is essential to supplement mathematical solutions by *a priori* knowledge about the image. On the other hand, most of the algorithms in the field extend grey-scale image segmentation techniques, such as histogram thresholding, clustering, graph-based, region-growing, edge detection, and fuzzy-based approaches to colour image segmentation. Nevertheless, colour obviously will provide additional information compared to grey-scale images and as a consequence it can be made to produce more reliable image segmentation results.

When segmentation methods with comparably reliable results are introduced in the field, they are in all cases designed for a specific narrow application with pre-defined knowledge about the image context to be segmented. The specific solution usually has computationally efficient performance. On the other hand, general-purpose segmentation algorithms will not be universally reliable and at the same time can be less computationally efficient. This is most likely because the segmentation is algorithmically more complex in its implementation. Unfortunately, in almost all cases, an efficient computational performance of the



newly-introduced segmentation algorithm will be the last thing to be considered as a required feature. When the computational performance is considered, there is usually less comparison with existing algorithms, both in algorithmic build and computational efficiency. The novelty of the new implementation will always be considered over other features of the algorithms, as research publication often depends on novelty.

In addition to the efficient performance of the segmentation algorithm, in both the quality of the output and computational processing, the performance of the segmentation evaluation is as important. One method to improve the existing set of evaluation methods mentioned above in this chapter is to use machine learning techniques. This will improve the evaluation reliability by -for example- training based on accepted segmentation results, most likely hand-segmented images by a human evaluator. This will also improve the evaluation processs computational performance efficiency by both lowering the time of the evaluation and focusing on evaluating more efficient algorithms. This is one of the main focuses of this thesis that will be explored in more detail in the coming chapters.

If this proves to be efficient enough then the hope is to direct future research to implement these efficient segmentation evaluation methods in the image segmentation process itself. In essence, all evaluation framework will need to perform an image segmentation process to evaluate the final segmentation result and finally provide the quantitative evaluation score. Consequently, if the evaluation process is efficient enough then we can have an embedded evaluation framework in all segmentation methods, which can provide a continuous “feedback” to the segmentation process to improve the final segmentation results for the next set of input images. Further research will need to be done in this direction, and this

## 2.5 Summary

---

thesis will only be able to give some foundational results that can help in this direction.

# 3

## Analytical evaluation of image segmentation algorithms

Don't get involved in partial problems, but always take flight to where there is a free view over the whole single great problem, even if this view is still not a clear one.

---

Ludwig Wittgenstein (1889 - 1951)

### **3.1 Image Segmentation Algorithms: analysis**

---

In general, any study of an algorithm will depend on *a priori* knowledge or what is called analytical evaluation and *a posteriori* knowledge referring to information gained by performing empirical evaluations. The empirical evaluation is considered *a posteriori* knowledge because the analysis depends on the results of the segmentation algorithm unlike the analytical evaluation that only studies the pro-

### 3.1 Image Segmentation Algorithms: analysis

---

cess of the algorithm without considering the output. The empirical evaluation (*a posteriori* knowledge methods) include both supervised methods that depends on *a priori* information (ground truth images) and unsupervised methods that don't need any *a priori* information. This segmentation evaluation's categorisation was discussed in more detail and summarised in the previous chapter in Section 2.4.

In what now is considered the classical survey of image segmentation evaluation methods by Zhang (1996) and even in the most recent survey in 2008 (Zhang *et al.*, 2008) this evaluation categorisation is still used, and where there are other more detailed and highly refined classifications in the literature, this broader classification encompasses and covers the rest as discussed earlier. This chapter will focus on the analytical evaluation of the algorithms and we will explore the other evaluation categories in more detail in a later chapter.

In the literature there is still no consensus on a general image segmentation theory and as such it is not possible to create a benchmark image segmentation algorithm model that can be used to analyse the current segmentation algorithms against. Analytical evaluation in general assesses an algorithm's implementation independently of its input "type" or output "quality". It mainly analyses the algorithmic properties such as the implementation model, processing complexity and efficiency. Because There is no one algorithm model to refer to, and as mentioned above there is no assessment criteria to benchmark the algorithm design and implementation. As such this evaluation method is usually considered less useful in the case of image segmentation Zhang (1996), and in most image segmentation and evaluation studies, it is even mostly used as an afterthought and in other cases it is actually not considered or used at all compared to other evaluation methods. This is because almost all the studies aim at only improving

---

### 3.2 Profiling using Valgrind's Tool Suite

---

the quality of the segmentation results quality without any consideration of how this can impact on the computational performance of the segmentation process.

However, it is still useful to understand and analyse the algorithm design and implementation. This can include, but is not limited to, processing strategy, resource efficiency, and processing complexity. This can help at least to categorise the different segmentation algorithms available and also to consider their processing performance and find ways to improve the segmentation quality and computational performance, either directly (for example by modifying the algorithm) or indirectly (by choosing the best algorithm for the application under study or choosing the best parameters).

### 3.2 Profiling using Valgrind's Tool Suite

---

The Valgrind toolsuite (Nethercote & Seward, 2007) was used to profile the segmentation algorithms in the following sections. Simple algorithm analysis without any profiling tool can provide some help in analytical evaluation, but it will not provide any insight into the computational performance properties of the algorithms. This is where profiling tools like Valgrind can be essential. Valgrind consists of a number of tools that help in debugging and profiling programs to trace memory leaks, detect threading problems, and many more operations. Specifically it provides an easy tool to profile the algorithms and produce detailed call-graphs of the operations performed within the programs. Valgrind has an important advantage in being a dynamic binary instrumentation tool so it can work with any program, and there is no need to recompile or re-link the programs or have access to the source code.

### 3.3 Analytical evaluation of thresholding algorithms

---

Callgrind, the Valgrind profiling tool, was useful in providing information on where the segmentation algorithm is spending its time, how much time is spent on different processing stages, and how those stages are similar in different segmentation algorithms from different categories in the taxonomy. In particular, using similar pre-/post-processing stages in the algorithms design as detailed in the following sections.

### 3.3 Analytical evaluation of thresholding algorithms

---

The algorithm design of thresholding techniques involves examining each pixel against a threshold; see Figure 3.1 for the classic thresholding implementation. Figure 3.2 illustrates the profiling result of the algorithm, and shows basically that the thresholding operation is distributed between different iterator operations that perform the search mentioned earlier. There is no bottleneck operation or dominant operation that takes a significant part of the processing.

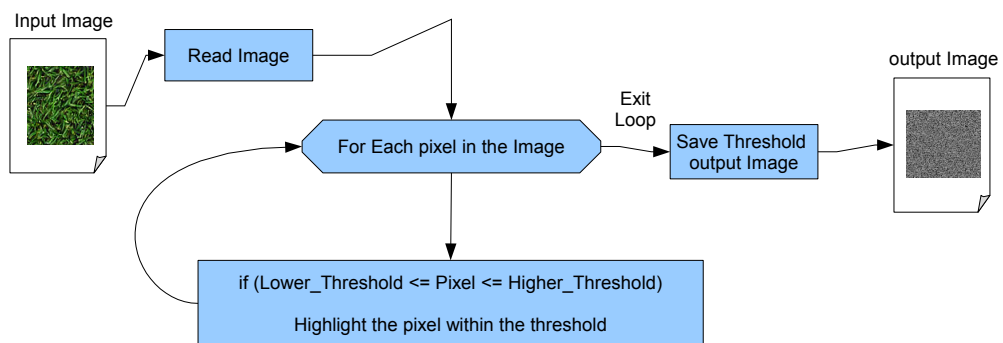


Figure 3.1: Classic thresholding method flow chart

Relative thresholding basically is a slight modification on the standard thresholding implementation, in which the input threshold parameter is interpreted as

### 3.3 Analytical evaluation of thresholding algorithms

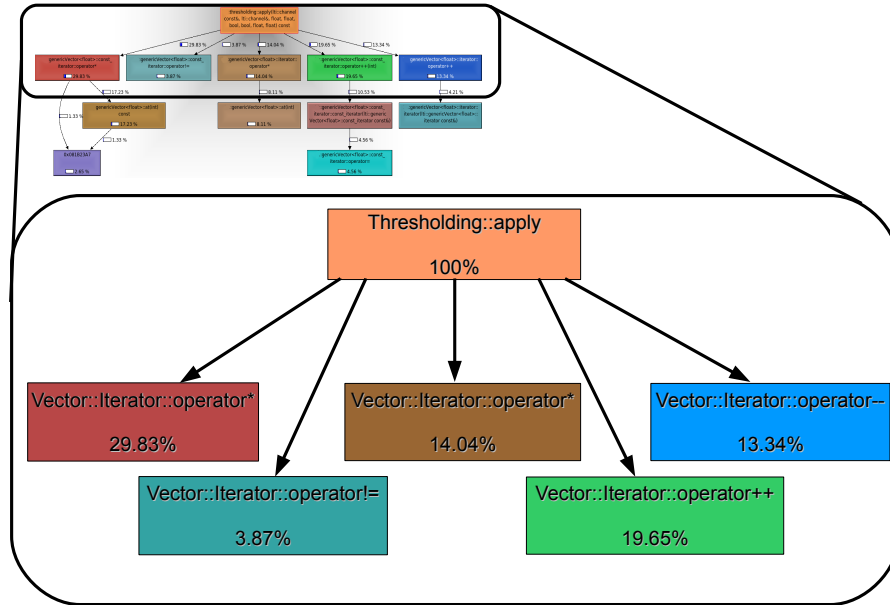


Figure 3.2: Profiling result of the classic thresholding method

a fractional rather than absolute value, as illustrated in Figure 3.3. As expected the profiling result is not different from the classic implementation as illustrated in Figure 3.4.

Optimal Thresholding (Otsu, 1979; Sonka *et al.*, 1993) is an important modification of the classic thresholding implementation. Optimal thresholding starts by adaptively calculating a threshold value for each input image that is statistically optimal, based on the contents of the image. Therefore, the algorithm defines two classes for the image pixels (e.g. foreground and background) and then computes the optimum threshold value that separates those two classes and minimises the intra-class variance. The algorithm is illustrated in Figure 3.5. This threshold value is then used in a standard thresholding implementation. About 63% of the processing time is spent on the optimal threshold search and the other 37% on

### 3.3 Analytical evaluation of thresholding algorithms

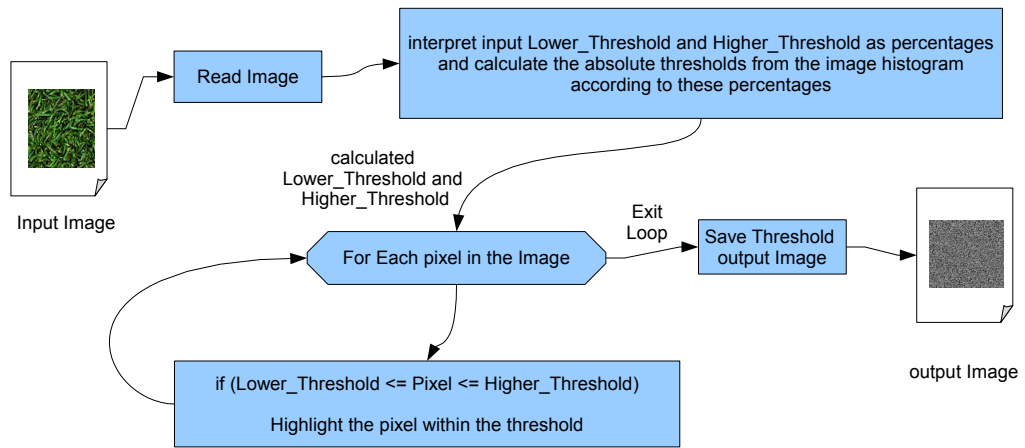


Figure 3.3: Relative thresholding method flow chart

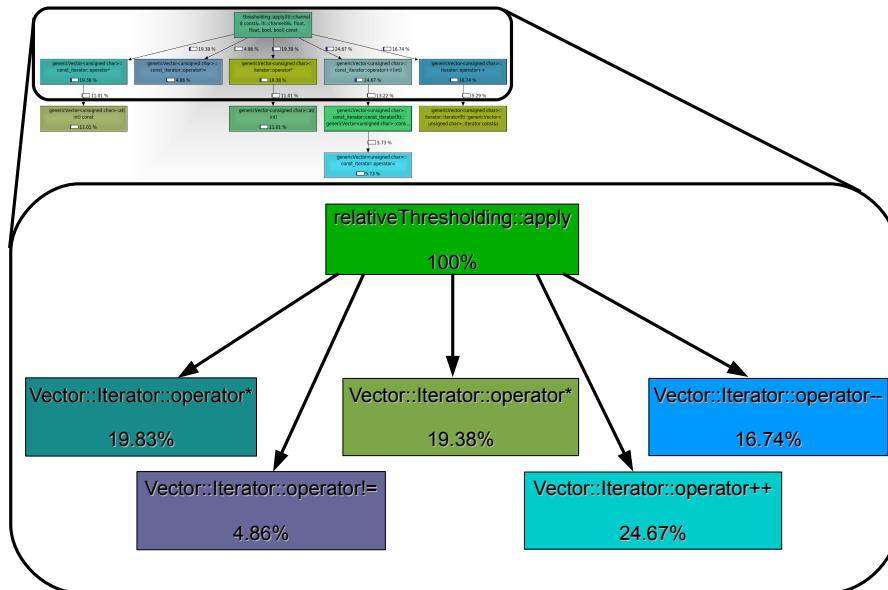


Figure 3.4: Profiling result of relative thresholding method



### 3.4 Analytical evaluation of edge-detection algorithms

---

the thresholding operation itself, see Figure 3.6. In general, more time is spent computing the optimum threshold value than on the thresholding process itself.

Although this is a good enhancement to the thresholding technique as there is no need for the user to arbitrarily determine the threshold values as input parameters, the problem is still present where, due to ambiguous discontinuity values in the image, the optimal threshold search may compute a false detection (corresponding to a local optimal solution). In addition, the threshold value is calculated as a global value on the image as whole and doesn't consider local variety in different parts of the input image.

Local adaptive thresholding was developed to solve this problem. This approach combines automatic threshold computation technique (like optimal thresholding) while looking locally at each pixel and its neighbourhood of pixels. In this approach, 50% of the processing time is spent on an optimal threshold search and the other 50% on the thresholding operation itself. One example is to choose the threshold level for the given window is to use the mean value of the pixels in the window. However, while this will be effective and provides a significant advantage in the segmentation quality for images with local variation properties (like illumination variations), it will obviously have a higher computational cost than optimal thresholding because of the extra processing stages. (Park *et al.*, 2005; Sezgin & Sankur, 2004).

### 3.4 Analytical evaluation of edge-detection algorithms

---

Edge detection algorithm design depends on a search for discontinuities in the image and almost all of the edge detection implementations search for edges

### 3.4 Analytical evaluation of edge-detection algorithms

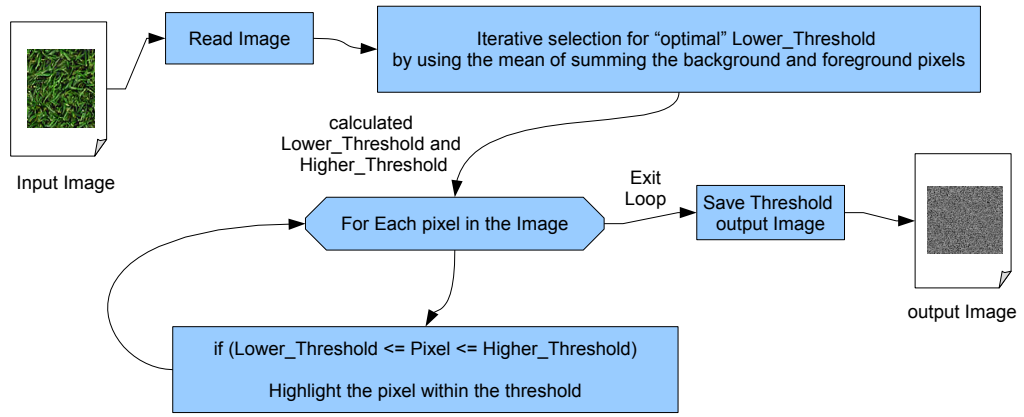


Figure 3.5: Optimal thresholding method flow chart

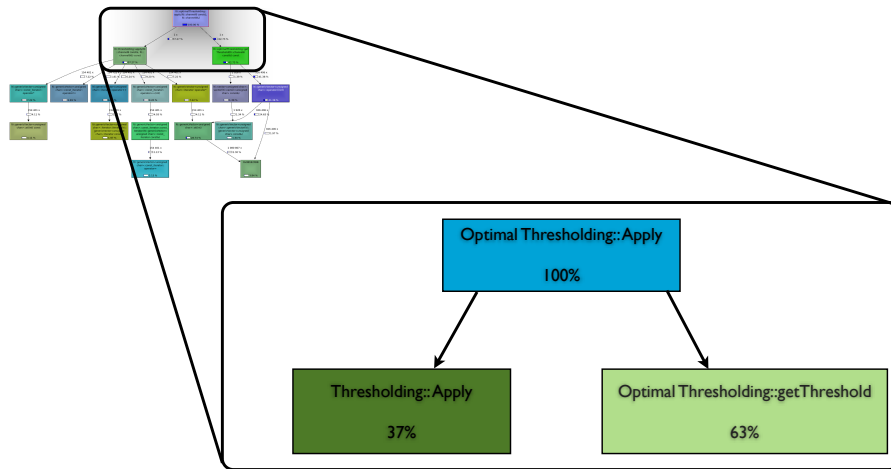


Figure 3.6: Profiling result of optimal thresholding method

(discontinuities) in the gradient of the input image.

Figure 3.7 illustrates a standard implementation of an edge detection algorithm. Basically it depends on two steps:

### 3.4 Analytical evaluation of edge-detection algorithms

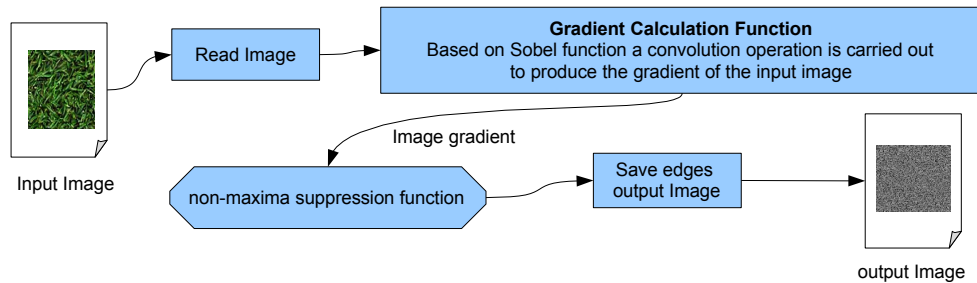


Figure 3.7: Classic edge detector method flow chart

1. Gradient Operation: this operation is basically a convolution function of the input image with a gradient kernel, in this case the Sobel kernel. This step takes approximately 90% of the whole edge detection operation processing time and of that 70% is actually the convolution operation and the rest is spent on data manipulation operations, as illustrated in Figure 3.8.
2. Non-maxima suppression (Nixon & Aguado, 2008): this operation basically tries to find all the local maxima, and suppress the rest. So edge points are detected where there is a local directional maximum in the calculated gradient space. This filter takes no more than 10% of the whole edge detection operation.

In general edge detection methods differ by using different algorithms to perform the two steps mentioned above. In addition, some of the algorithms add a smoothing filter step before the gradient computation step.

The Canny edge detector (Basu, 2002; Canny, 1986) is one of the most well-known and widely used edge detectors in the literature. The Canny edge detector adds a Gaussian smoothing filter (which is basically a Gaussian convolution oper-

### 3.4 Analytical evaluation of edge-detection algorithms

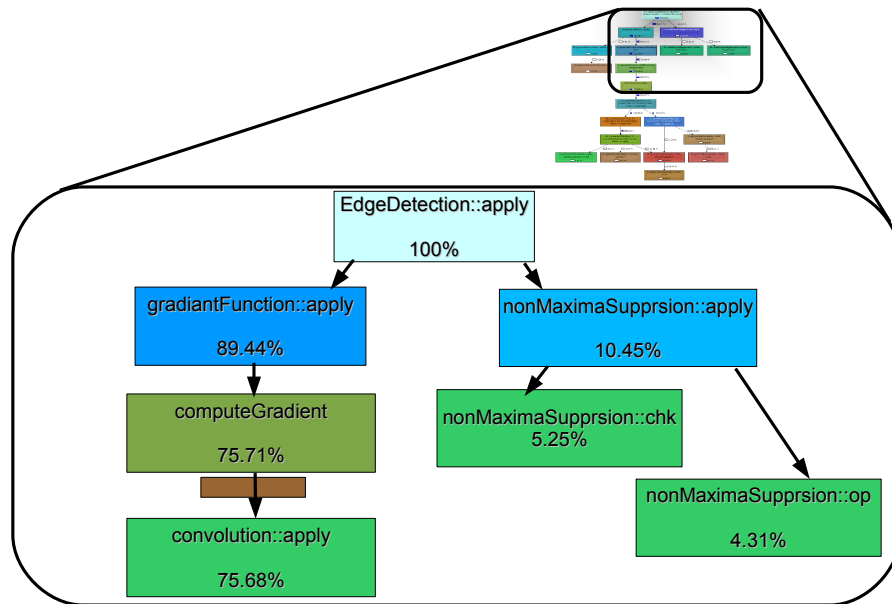


Figure 3.8: Profiling result of classic edge detector method

ation). Then it completes the steps as described with the standard edge detection method above, as illustrated in Figure 3.9.

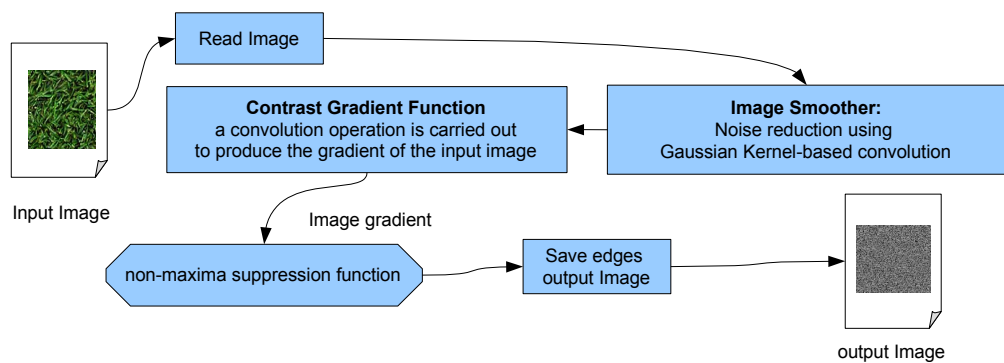


Figure 3.9: Canny edge detector method flow chart

### 3.4 Analytical evaluation of edge-detection algorithms

In this case about 75% of the processing time is taken by the smoothing filter, which is basically a convolution operation as illustrated in Figure 3.10. In later chapters, the effect of the smoothing mask size, as adjusted using an input parameter, on the computational performance will be studied in more detail. As well as the effect on the segmentation quality. For colour images, as in this case, our Canny edge implementation applies a colour contrast gradient as introduced by [Cumani \(1991\)](#). Our research focusses on colour images, so it's important to add any processing stages that give us advantage in segmenting colour images.

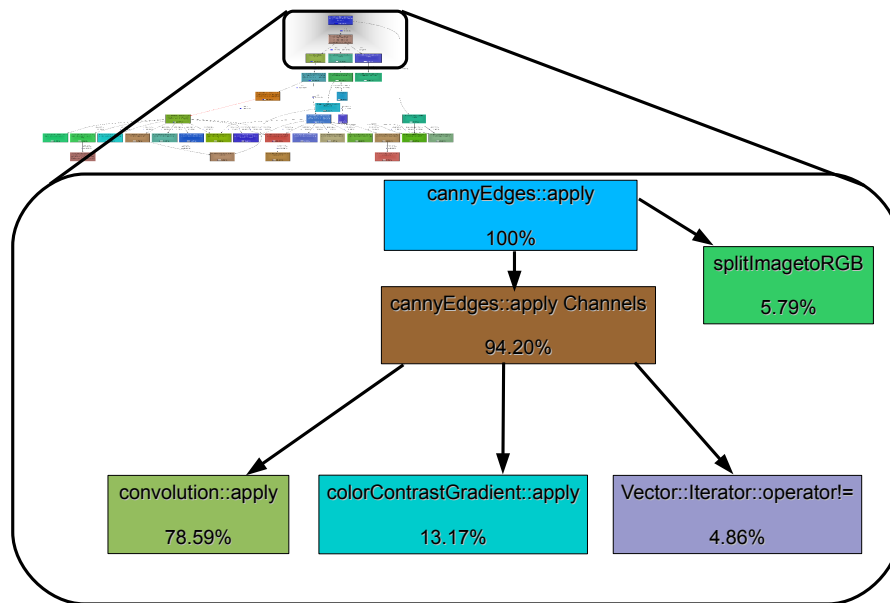


Figure 3.10: Profiling result of Canny edge detector method

Another edge detection method that provides an enhancement to process colour images is the one suggested by [Gevers & Stokman \(2003b\)](#); see Figure 3.11. This detector is optimised for colour images. As a result most likely it will not have any advantage on grey-scale images. Even if it have any advantage in

### 3.5 Analytical evaluation of region-growing segmentation algorithm

---

the segmentation quality, it will still be computationally expensive. Because it operates on all three channels in all of its steps, it's a relatively computationally expensive operation. About 90% of the processing is taken by the gradient calculation operation, which is basically a number of convolution operations. See Figure 3.12.

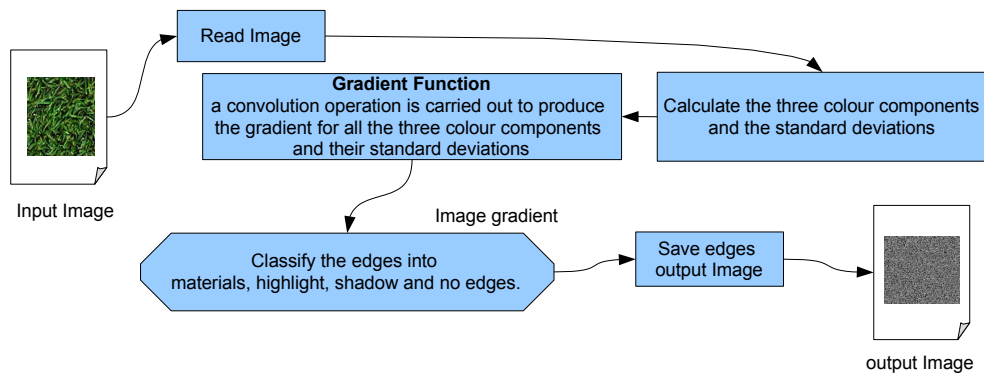


Figure 3.11: Colour edge detector method flow chart

### 3.5 Analytical evaluation of region-growing segmentation algorithm

---

The region-growing approach is based on the fact that neighbouring pixels have similar intensity values. Therefore, according to a given number of seeds within the image space (or starting by default in the image corners) the seeds are expanded to regions (Gonzalez & Woods, 2002). A threshold acts to assess the similarity between neighbouring pixels and if they are part of the seed's regions or not. In other words, each pixel value in the search window is compared with

### 3.5 Analytical evaluation of region-growing segmentation algorithm

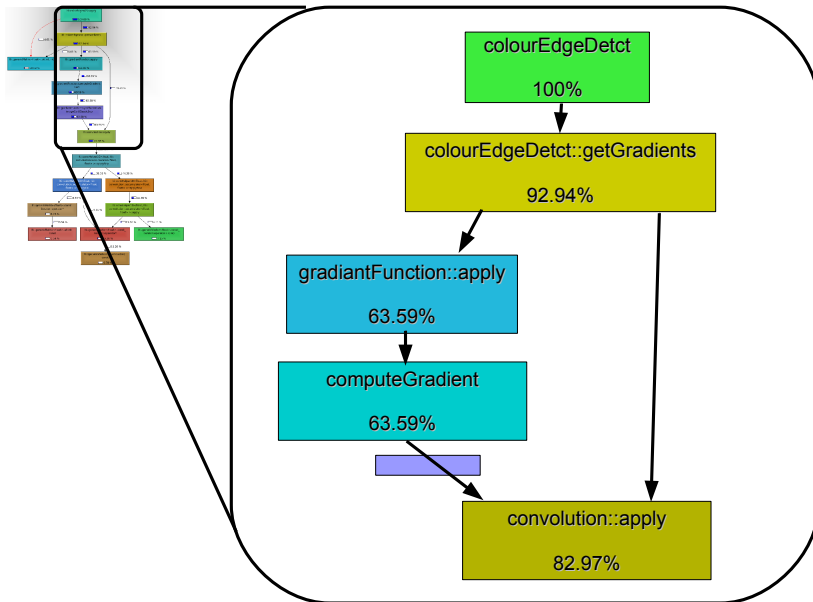


Figure 3.12: Profiling result of colour edge detector method

the average value in the seed position for the given search window. If the value is lower than the threshold then that given pixel will be considered as part of the growing region. See Figure 3.13 for an illustrative flow chart. Rejected pixels form new regions until all pixels are labelled.

The image smoothing operation takes about 55% of the processing (which is basically a Gaussian convolution), and about 5% for the starting edge detection operation (to initialise some of the regions' boundaries). The rest of the processing is spent on the region-growing, which is basically iterating through data structures. See Figure 3.14.

### 3.6 Analytical evaluation of watershed-based algorithms

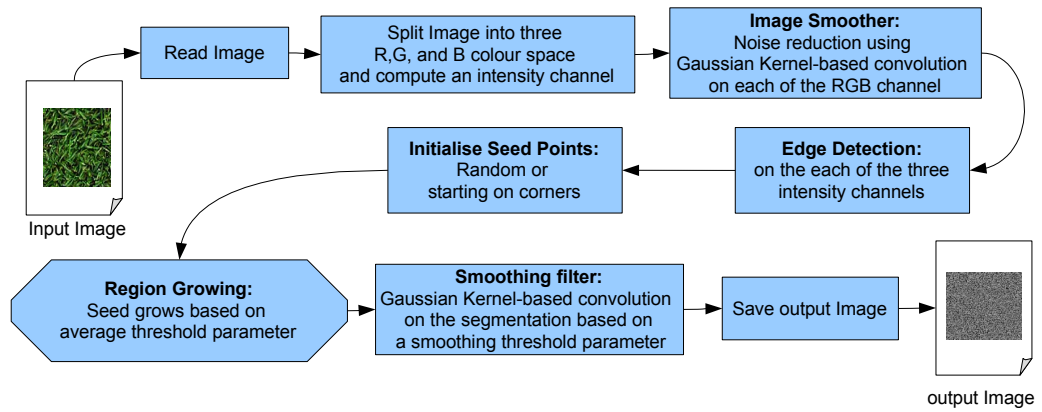


Figure 3.13: Region-growing segmentation flow chart

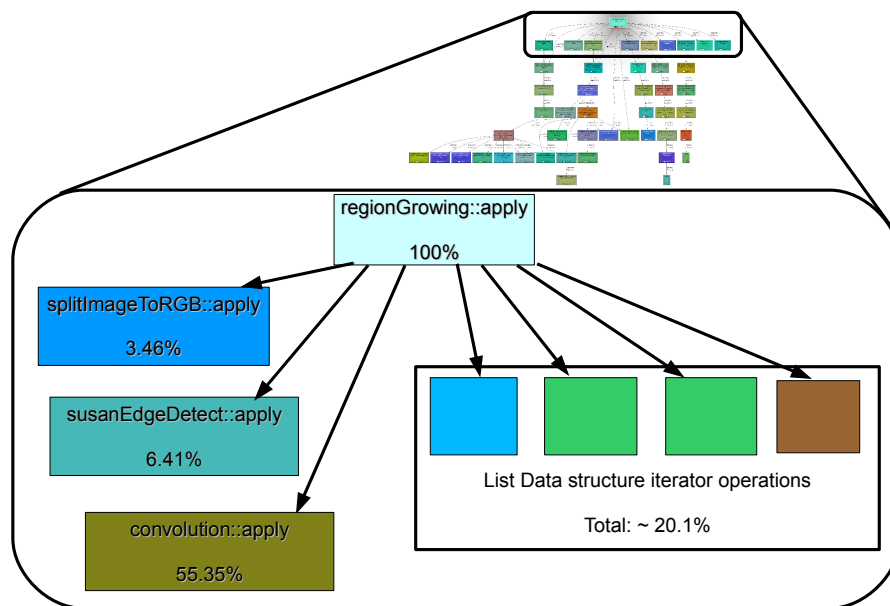


Figure 3.14: Profiling result of region-growing segmentation

## 3.6 Analytical evaluation of watershed-based algorithms

The watershed algorithm approach, inspired by Mathematical Morphology (MM) theory (Roerdink & Meijster, 2000), views the input image as a topographical



### 3.6 Analytical evaluation of watershed-based algorithms

map. This topographical map usually corresponds to a gradient map of the input image. In other words, the image gradient magnitude defines the analogy of the topographic surface. Pixels with the highest gradient magnitude values define the watershed lines, or the region boundaries. Whereas, the pixels enclosed within these boundaries define the water flowing down to the valleys with the lowest gradient magnitude values. The standard implementation only considers the intensity gradient maps (and as such colour information in coloured images is not considered; all images are processed on grey-scale space). Figure 3.15 illustrates the flow chart for a standard watershed implementation.

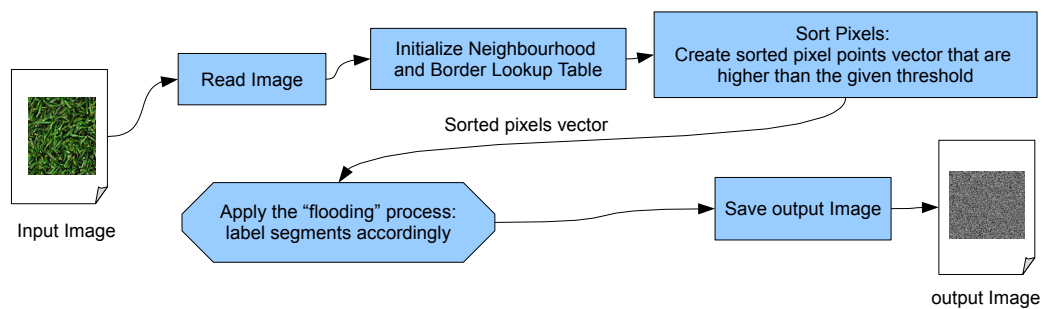


Figure 3.15: Watershed segmentation flow chart

From the Figure 3.16 you can see that about 80% of the processing is taken up by what is called the “flooding” operation. This operation consists of a number of loops that perform the segmentation operation on the image, the most important loops being: the indexing loop, and extending the basins’ loop (which is growing the regions in the image to get the segment), checking the neighbour regions loop, and finding new minima loop among others.

The loop operation (usually implemented as a number of iterators with a

### 3.6 Analytical evaluation of watershed-based algorithms

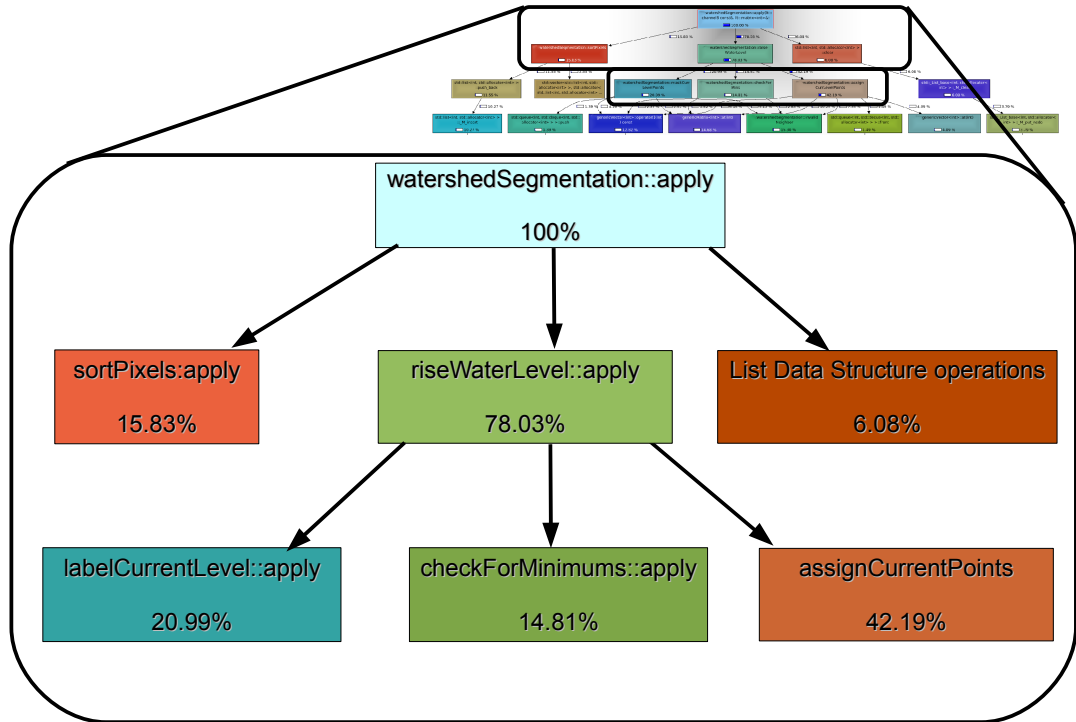


Figure 3.16: Profiling result of watershed segmentation

FIFO queue data structure) presents a potential for parallel processing. However, because there is a great deal of information passed between different parts in this operation (the watershed works on the global-level of the image) to create the region, the parallelisation advantage is overcome by the cost of high-message passing requirements between different parts of the parallelised parts of the algorithm.

The rain-falling watershed method makes a slight modification on the standard watershed method to make the whole process less affected by the over-segmentation problem. While it's true that, unlike the standard watershed im-

### 3.6 Analytical evaluation of watershed-based algorithms

plementation, the “region-growing” part of the method is less computationally intensive, the whole process is still slowed by an initial step that tries to find the lowest parts of the topographical map corresponding to the image, and this search process is computationally highly intensive search (it depends on a number of nested loops that search the image on the global level). While the following steps can be optimised this initial step will always be a highly expensive operation, see Figure 3.17 for the rain-falling processing steps.

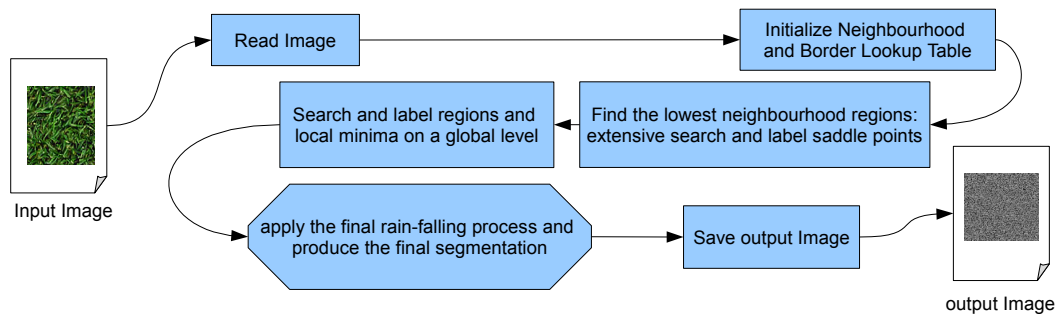


Figure 3.17: Rainfalling segmentation flow chart

And as illustrated in Figure 3.18, the “raining” operation doesn’t take more than 6% of the whole processing, while about 80% is done by the initial minima-finding step mentioned above (analogous to the “flooding operation in the standard watershed).

One approach to using the colour information in coloured images is the approach mentioned in Section 2.2.2.4. This approach basically uses the colour contrast gradient (similar to what is used by Gevers & Stokman (2003b) edge detector in Section 3.4) as the topological map equivalent for the watershed process. This approach suggests using a smoothing filter at the start to reduce

### 3.6 Analytical evaluation of watershed-based algorithms

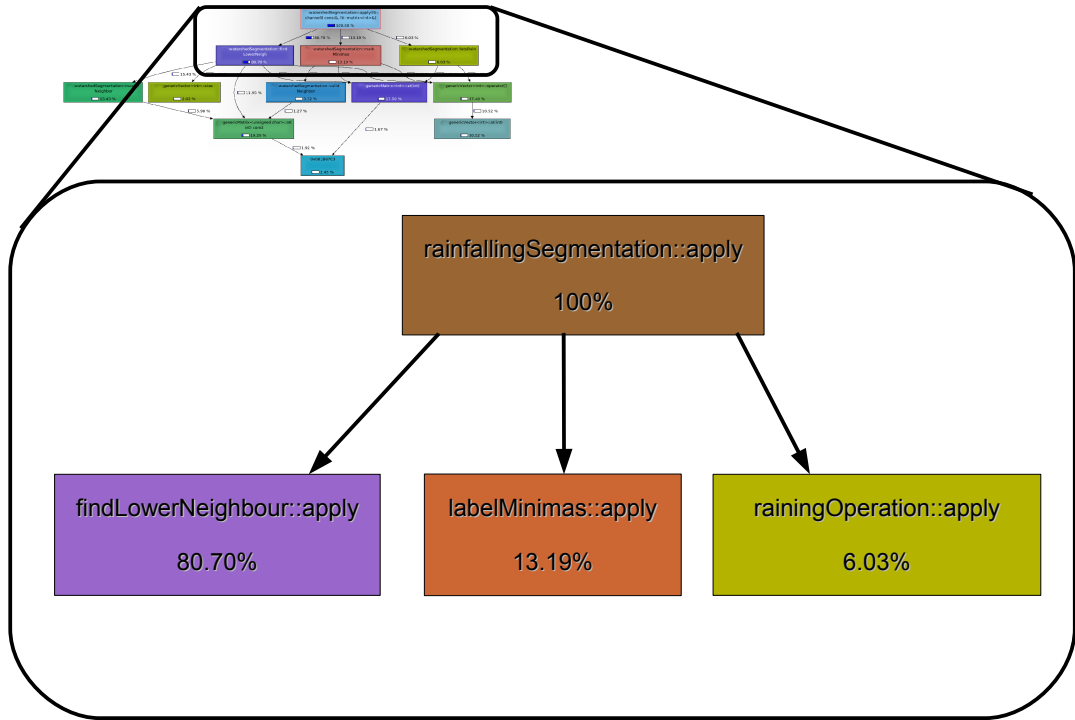


Figure 3.18: Profiling result of rain-falling segmentation

over-segmentation with good results (Alvarado, 2004). Figure 3.19 illustrates the steps taken in this approach.

The pre-processing of the image by using a smoothing filter (a median filter in this case) and the watershed process (based on rain-falling in this case) both take only about 15% of the processing each. The colour contrast gradient computation takes about 26% of the processing, while the post-processing region-merging step alone takes about 40% of the whole processing power. See Figure 3.20 for more information.

### 3.6 Analytical evaluation of watershed-based algorithms

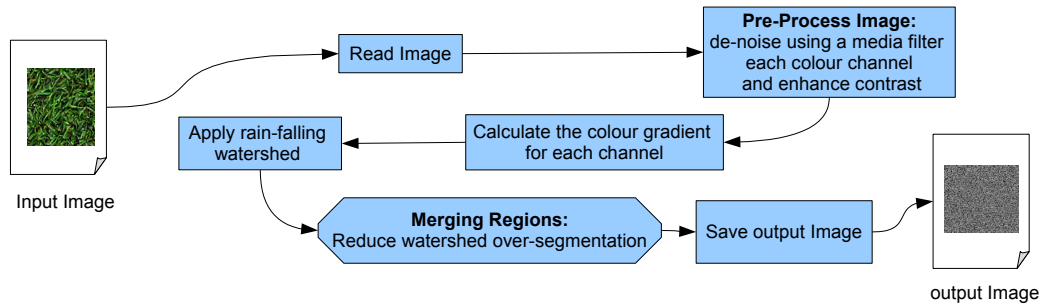


Figure 3.19: Colour Watershed segmentation flow chart

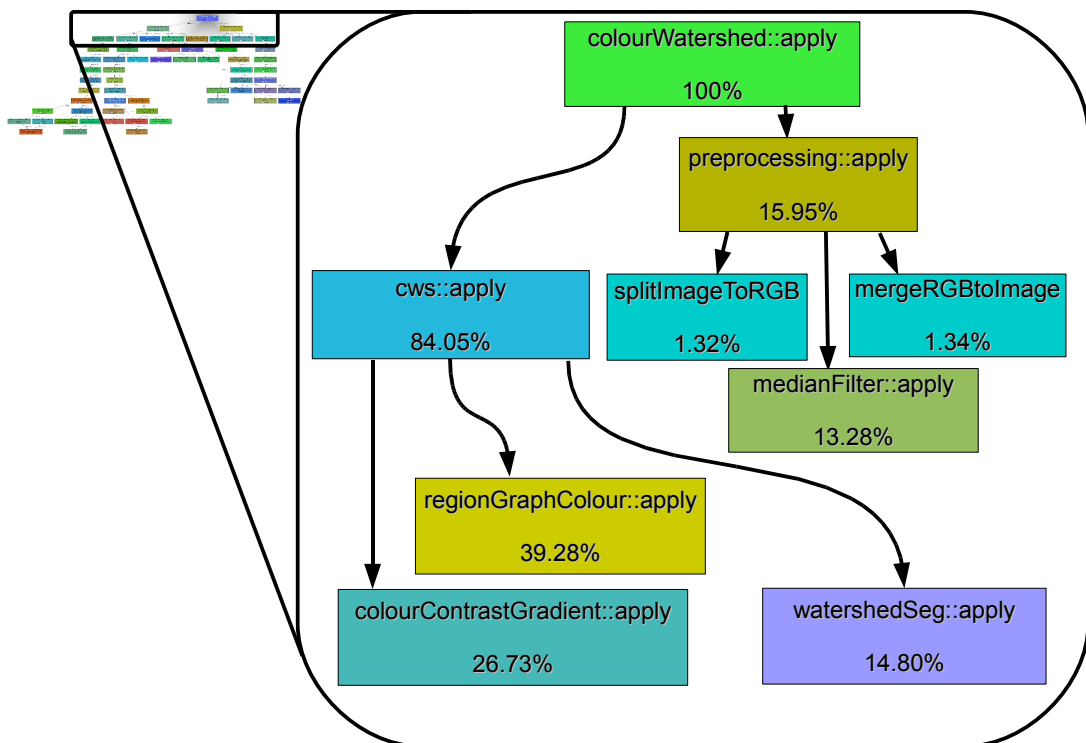


Figure 3.20: Profiling result of colour watershed segmentation

## 3.7 Analytical evaluation of clustering-based image segmentation algorithms

---

Clustering segmentation algorithms are based on partitioning an image into clusters of pixels that minimise the variance between the pixels' colour and the cluster centre. In another sense, it can be considered as an image quantisation algorithm.

The k-means segmentation (Marroquin & Girosi, 1993b) uses the k-means clustering method (MacQueen, 1967) to apply this image quantisation step. In addition, it includes a smoothing filter to smooth edges and eliminate noise from the final result. Figure 3.21 illustrate the k-means segmentation steps.

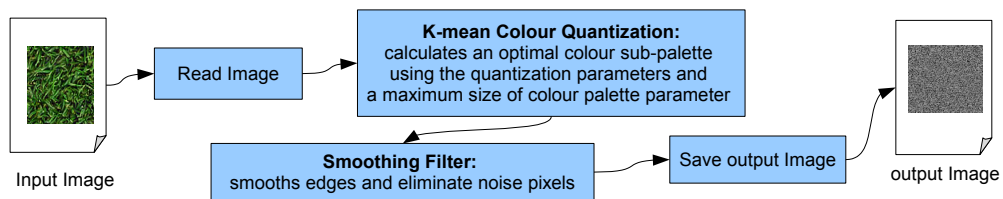


Figure 3.21: K-means segmentation flow chart

Although this segmentation includes few steps, the k-means quantisation is a computationally intensive operation (80% of processing), especially the highly iterative step that searches through the pixels to assign them to different clusters and then re-compute the clusters' centre and finally repeat the cluster labelling for pixels until no pixel changes a cluster label. As such the k-means clustering will be considered converged, see Figure 3.22.

Local k-means clustering (Verevka & Buchanan, 1995) is just a slight modification on the original k-means implementation that creates a Look-Up Table

### 3.7 Analytical evaluation of clustering-based image segmentation algorithms

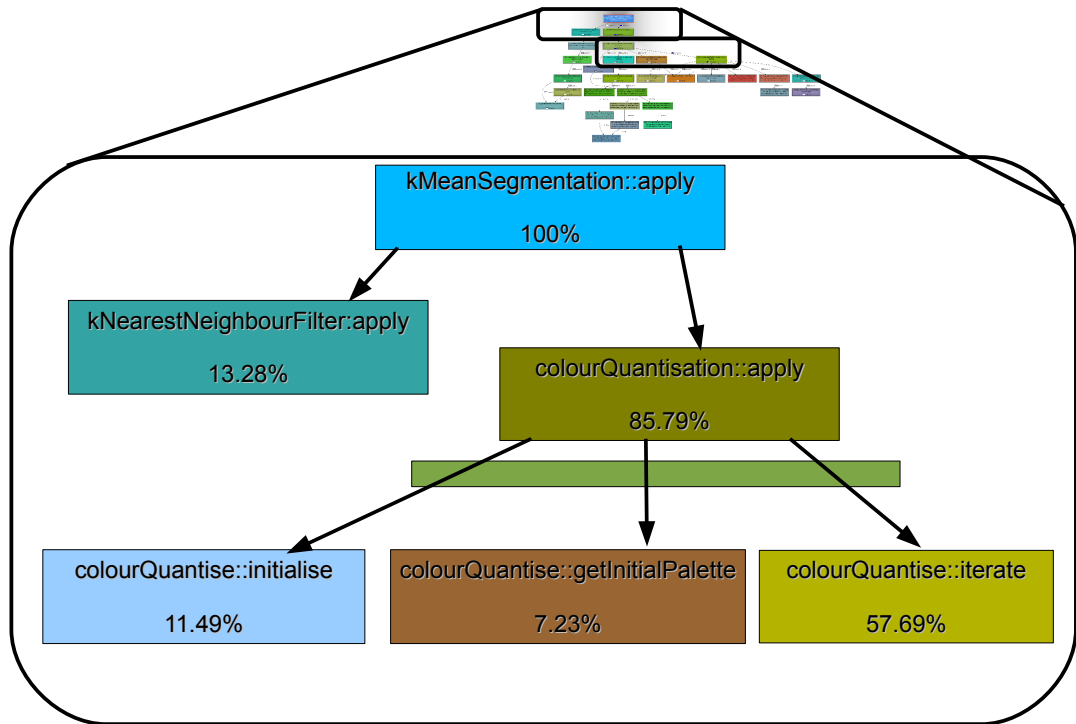


Figure 3.22: Profiling result of K-means segmentation

(LUT). The table is used to look for candidate centres that are initialised at the start. As a result, this process reduce the k-means convergence time. This implementation doesn't use a smoothing filter step as can be seen in Figure 3.23. The profiling for local k-means segmentation can be see in Figure 3.24.

One important enhancement on the clustering segmentation algorithm is introduced with the mean-shift segmentation. In addition to using a mean-shift clustering step, this clustering step itself is nonparametric and as such does not require prior knowledge of the number of clusters, and does not constrain the shape of the clusters (Comaniciu & Meer, 2002).

### 3.7 Analytical evaluation of clustering-based image segmentation algorithms

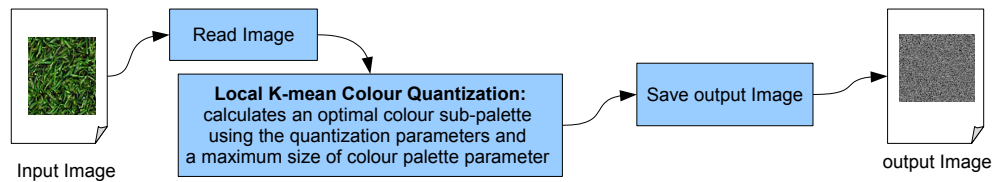


Figure 3.23: Local K-means segmentation flow chart

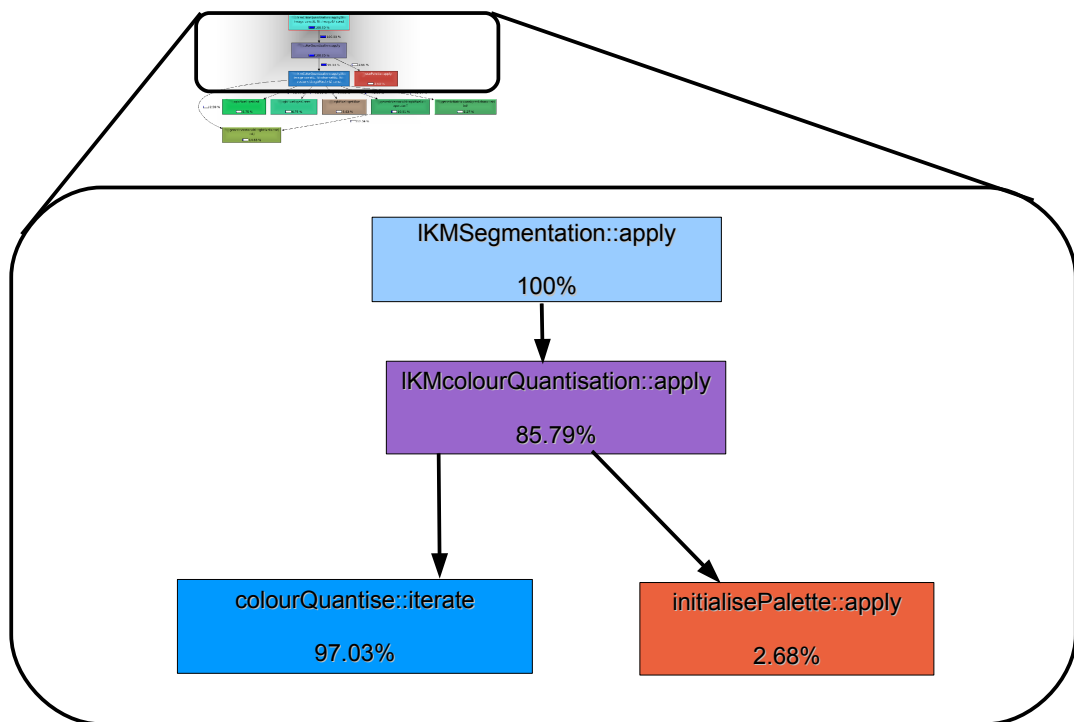


Figure 3.24: Profiling result of local K-means segmentation

Rather than beginning from initial seeding points, the mean shift algorithm begins at each pixel in the input image and estimates the local density of similar pixels (i.e., the density of nearby pixels with similar colour). In more detail,



### 3.7 Analytical evaluation of clustering-based image segmentation algorithms

---

the algorithm estimates the local density gradient of similar pixels and finds the peaks in the local density gradient. Finally, pixels are considered to be members of the same segment if they are “closer” to the same peak in the local density gradient (Wang *et al.*, 2004).

The application of mean shift to an input image consists of two stages. The first stage is to define a kernel for each pixel. This kernel defines a measure of distance between pixels, where distance is defined by the spatial, as well as, the colour distance. The second stage of the mean shift algorithm is the search stage of the algorithm. First the stage initialises a mean shift point for each pixel. Secondly, each mean shift point is iteratively moved upwards along the gradient of the density function defined by the sum of all the kernels until they reach a stationary point. The mean shift points associated with the same stationary point are considered to be members of the same segment. As a consequence, the pixels associated with this set of mean shift points defines the given segment. For a more detailed mathematical definition and implementation details refer to Comaniciu & Meer work (Comaniciu & Meer, 2002).

Neighbouring segments may then be combined in a post-processing stage. The algorithm depends on a number of post-filtering steps to reduce the noise and smooth the edges of the segments, and most importantly, a step to prune regions by their size. This stage uses input parameters to join regions whose area is: 1) less than the input threshold with its respective candidate region adjacent to the region being pruned and 2) that is closest in colour cluster, see Figure 3.25.

It will be noted later the importance of those pre-/post-processing filter steps in enhancing the quality of the segmentation results in all of the segmentation implementations and how they can also reduce the complexity and computational

### 3.7 Analytical evaluation of clustering-based image segmentation algorithms

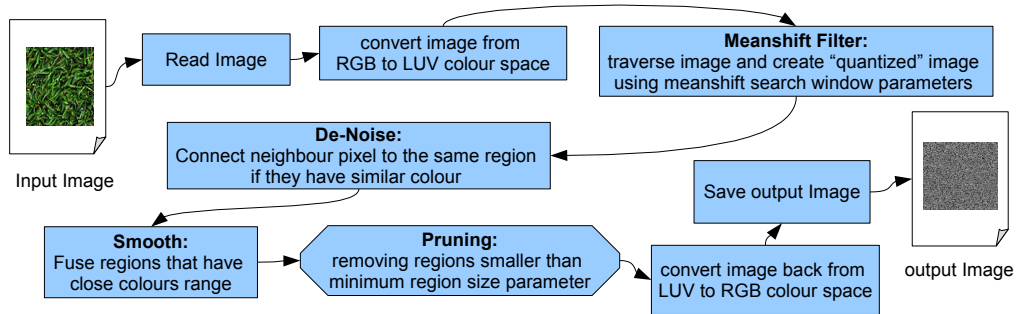


Figure 3.25: Meanshift segmentation flow chart

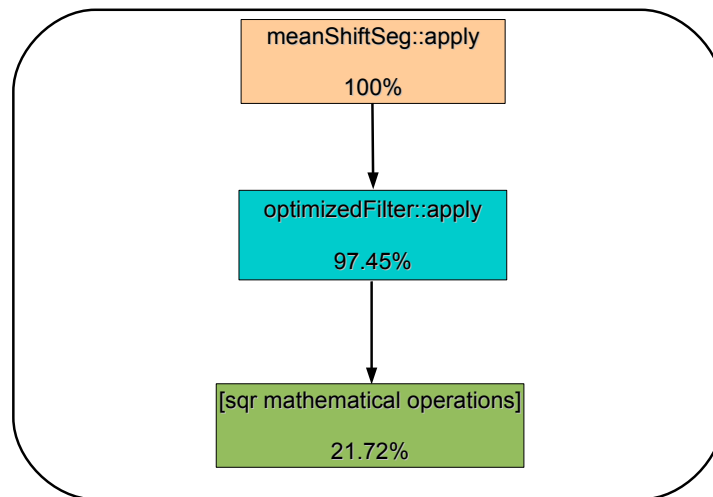


Figure 3.26: Profiling result of meanshift segmentation

power need of the core of the segmentation process itself.

## 3.8 Analytical evaluation of graph-based image segmentation algorithms

---

In general, graph-based segmentation algorithms utilise graph theory by creating a weighted graph of the pixels, where each pixel corresponds to a node in the graph, and weights on edges between the neighbouring pixels show the dissimilarity between these pixels. Then these nodes are merged into segments according to an input threshold value parameter. Figure 3.27 gives an illustration.

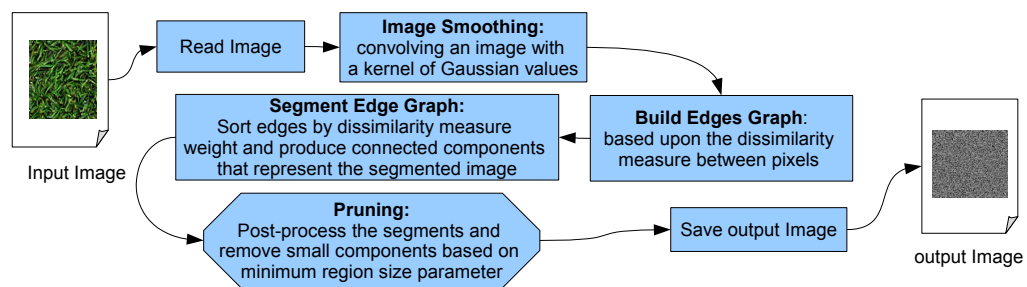


Figure 3.27: Graph-based segmentation flow chart

In other words, the algorithm defines the boundaries between regions by comparing two image properties: 1) Intensity differences across the boundary and 2) Intensity difference between neighbouring pixels within each region. This makes this method consider the intra-region properties in addition to the inter-region properties. The intensity differences across the boundary are as important if they are large enough relative to the intensity differences inside at least one of the regions. However this implementation also uses a pre-processing step to smooth the image with a Gaussian smoothing filter which take about 50% of the processing,

### 3.8 Analytical evaluation of graph-based image segmentation algorithms

and a post-processing step that prunes segments according to their size given by an input parameter, see Figure 3.28. In both cases, the processing is dependent on the input parameters' values that correspond to these stages. Further experimentation will be carried out in later chapters to determine their effect on the segmentation quality and computational performance.

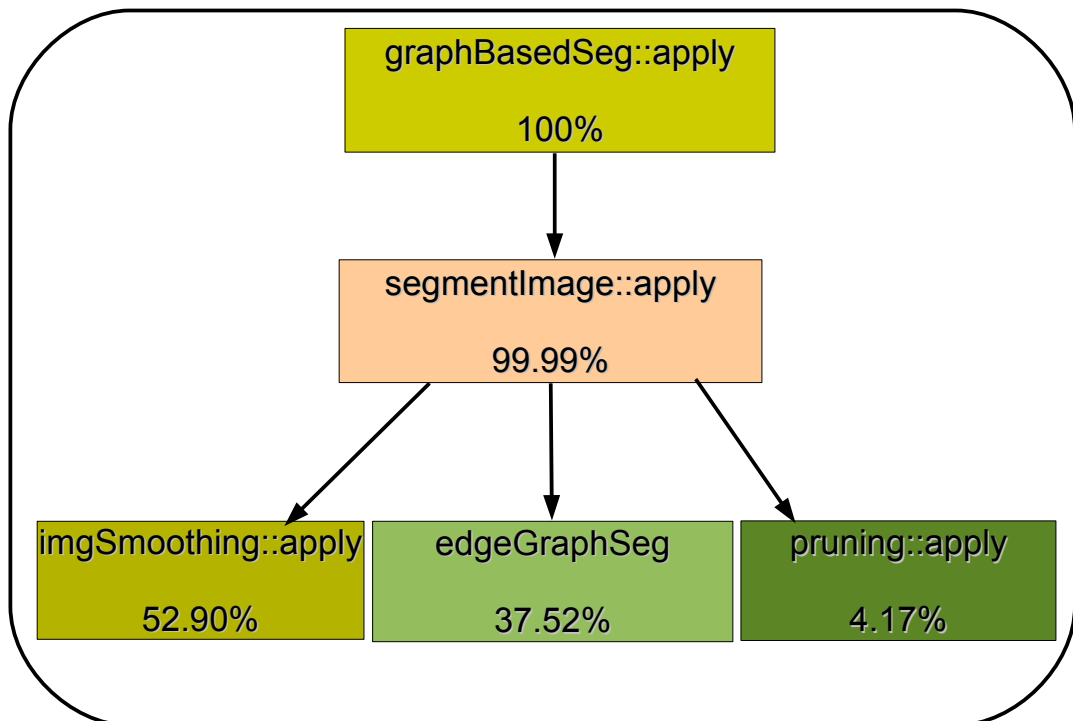


Figure 3.28: Profiling result of graph-based segmentation

## 3.9 Analytical evaluation of physics-based image segmentation algorithms

Anisotropic diffusion-based segmentation (Sumengen & Manjunath, 2005) can be considered a segmentation algorithm that depends on the physical representation of the objects in the images and is a perceptually inclined image filtering process. Anisotropic diffusion filtering has been used significantly for de-noising and enhancement of images. It aims at preserving 'object' boundaries in the image by not only considering the normal colour gradient of the image but also possible texture information in the image. It achieves this by looking for the image boundaries from different directions.

Anisotropic Diffusion filtering is applied a number of times based on an input parameter. In general it is a computationally expensive process because of its highly iterative nature but can be highly effective in providing high quality segmentation results (Sapiro & Ringach, 1996), see Figure 3.29 for illustration of the processing stages of this algorithm.

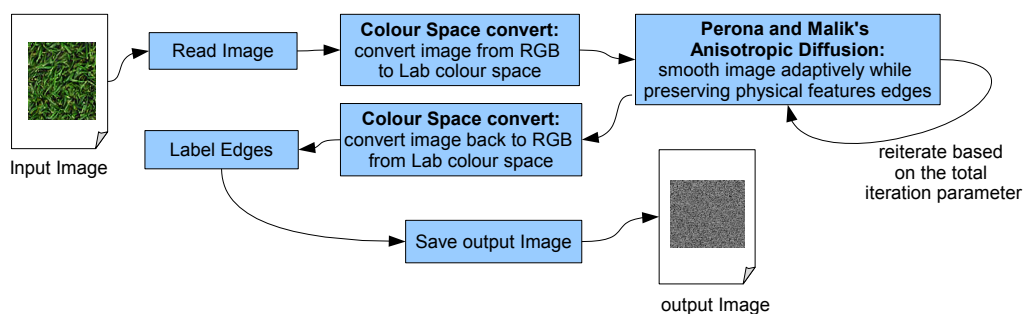


Figure 3.29: Anisotropic-based segmentation flow chart

---

## **3.10 Commentary**

---

This chapter provided a concise overview of a variety of segmentation algorithms implementations. This part of the study can be extended to provide additional detailed profiling information under multiple circumstances. The possibilities are nearly endless: parameter variations, different images, different implementation optimisation and different colour space. However what we were interested in to illustrate in this chapter was the different stages that are part of the segmentation algorithms, including pre-processing steps (like smoothing filters) and post-processing steps (like region pruning operations).

In the coming chapters we will study if variation in the input parameters can mainly affect those pre-/post-processing stages and can have an important effect on producing a high quality image segmentation results. The variation can also reduce the computational power needs of the segmentation as whole, either by optimisation or removing less effective stages in the segmentation algorithm if needed, and provide some recommendations on this front.

# 4

## Parameters' significance on the segmentation algorithms' performance

### 4.1 Overview

---

This Chapter provides a broad examination of the parameters' significance, with visual examples of the segmentation results using different parameters. Further rigorous evaluation and testing will be provided in the later chapters of the thesis to highlight how some parameters' changes play an important role in the quality of the final segmentation results and also affect significantly the performance of the computation.

The aim is to find the connection –if possible– between these parameters if possible and the “building” components of the segmentation algorithms as discussed and illustrated in Chapter 3. Not all the segmentation algorithms illustrated have results of the same significance. In other words, some methods provide simple

segmentation of one foreground object and the background and other methods produce very detailed multiple objects' segmentation results. As discussed earlier in Chapter 2, some of the algorithms only provide 'simplistic' final segmentation results (e.g., thresholding segmentation only provides a 'foreground' and 'background'). Others have only one significant parameter to vary (like the threshold parameter). However, those algorithms can also be used as one stage in a more sophisticated segmentation algorithm. This is similar to using a pre-processing smoothing filter and a post-processing pruning filter in the graph-based algorithm that have a modified thresholding technique as a core segmentation stage.

Figure 4.1 illustrates the input image that will be used in this chapter (on the left side), and also shows an example of the hand-segmentation reference image for the same input image that will be the basis of later evaluation tests as the ground truth image.

This chapter studies the influence of these parameters. Hence, the choice of input image is narrowed to one and visual representation is provided only as an illustration of the segmentation results. The aim of this chapter is to provide visual illustrations for the effect of parameter changes on the segmentation results. However, these are not rigorous testing results and hence the later chapters will provide further scientific testing on multiple images and evaluation results with a wider range of parameters.



---

## 4.2 The choice of the algorithms and related parameters

---

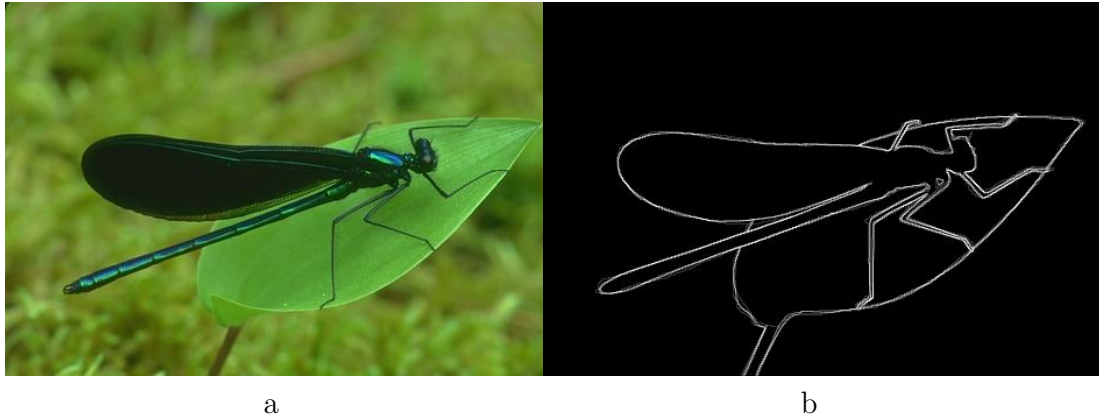


Figure 4.1: (a) Example image number 35070 from Berkeley segmentation database (b) the database human hand-segmentation for the same image

## 4.2 The choice of the algorithms and related parameters

---

The tests in this chapter are carried out on one image shown in Figure 4.1. Furthermore, the algorithms used here are used as a representative example of each branch of the taxonomy discussed earlier in Chapter 2. Some of the algorithms don't work directly with colour images but convert them to greyscale space or other colour spaces (such as the CIE Lab colour space). This requirement arises from the design and implementation decisions of the original authors of each algorithm. In this case, our research used the available implementation as long as it would work with colour images, although not all methods will use the colour information in the image to the same degree.

The parameters' choices and ranges were determined either by the suggestions and recommendations provided by the original authors' implementations, or by

---

### 4.3 Thresholding algorithms' parameters

---

the physical and scientific limits related to each parameter. For example, if the input images' colours were encoded into 256 values (between 0 and 256) then for example threshold parameter will be defined by this range and any segmentation algorithm will use this range to look for edges or segment boundaries.

### 4.3 Thresholding algorithms' parameters

---

Thresholding algorithms are simple by design, and have only two significant parameters: the high threshold, and the low threshold. Thresholding in basic terms aims at finding a discontinuity in the pixel values to detect the segments, so the upper and lower threshold values serve to narrow the algorithm's search within the colour space. Normalised values can be between 0.0 and 1.0 (corresponding to 0 to 255 in typical 8-bit grey-scale and 24-bit colour images). Of course, the high threshold parameter must always be higher than or equal to the low threshold parameter. In other words, the low and high thresholds are used in order to define the pixel-values of the regions where the thresholding operation should operate.

Figure 4.2 illustrates segmentation results from the classic thresholding algorithm. It is clear from the set of the results that varying the low threshold parameter makes for a bigger change in the number of segments detected, as well as for clarity and significance. By increasing the low threshold, the thresholding process is limited to higher pixel values (for grey-scale values from 0 to 255). As such, it will have less of a 'noise' filtering effect on the segments of the image and more segments will be included in the final segmentation result.

Relative thresholding also uses the same two parameters: the low and high

### 4.3 Thresholding algorithms' parameters

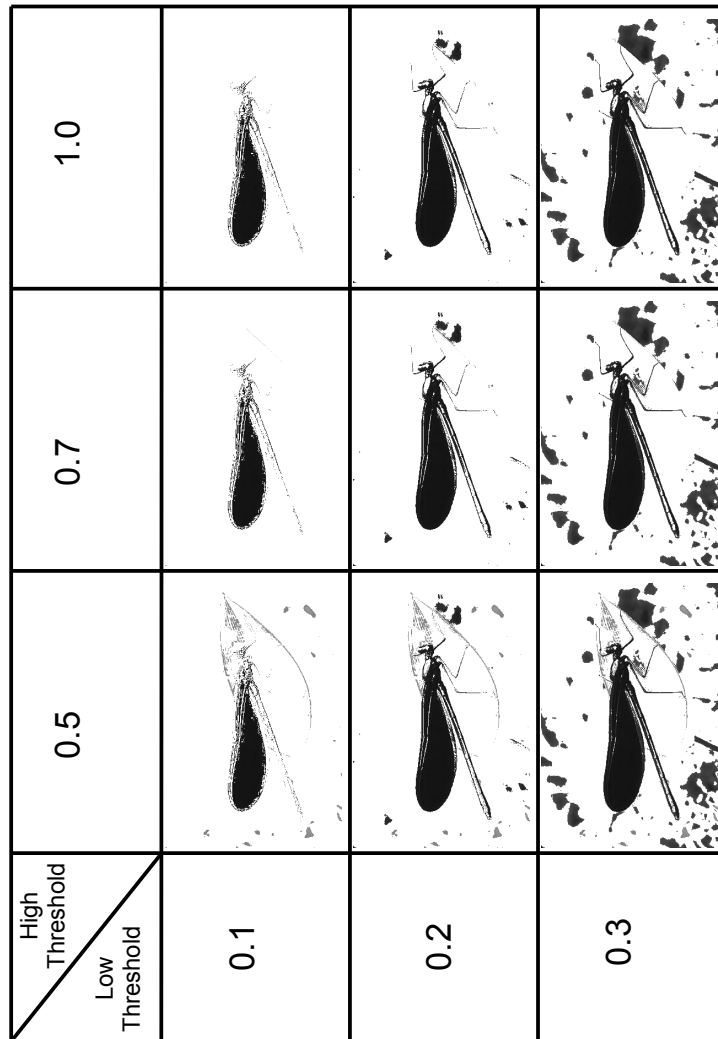


Figure 4.2: Variation of segmentation results with parameter settings for classic thresholding algorithm.

---

#### 4.4 Edge detection algorithms parameters

---

thresholds. However, it employs an alternative way of determining the thresholds: the parameters act as proportionate values. For example, a low threshold value of 0.3 will be interpreted as 30% of the pixel-values for the processed image. In other words, relative thresholding can be considered as a percentile thresholding. So for a 0.3 low threshold the lower 30% of the pixels will be cut off. An example is illustrated in Figure 4.3. The example image used shows very similar results to the results obtained with the classic thresholding algorithm above.

Overall, we can point to two main features of the thresholding technique. Firstly, it works only when the segments are clearly delineated by grey level. Secondly, the thresholding segmentation outputs are simple when part of the images are either part of the segments detected or they are part of the background (and discarded). As such, this technique is useful for relatively very quick and ‘coarse’ segmentation results or in a more practical sense are is a useful building block component as part of more ‘complex’ segmentation algorithms, like edge detection-based algorithms that will be illustrated in the next section.

#### 4.4 Edge detection algorithms parameters

---

Firstly we will look at experiments with the classic implementation of edge detection that uses a Sobel operator to compute the gradient for the image. Figure 4.4 illustrate variation of the maximal threshold parameter. The non-maxima suppression stage uses this parameter with the maximum found in the input image to calculate the real maximal threshold that will be used. This calculated maximal threshold serves to find edges such that pixels higher than this threshold will be considered as edges. The second ‘minimum threshold’ parameter is used after

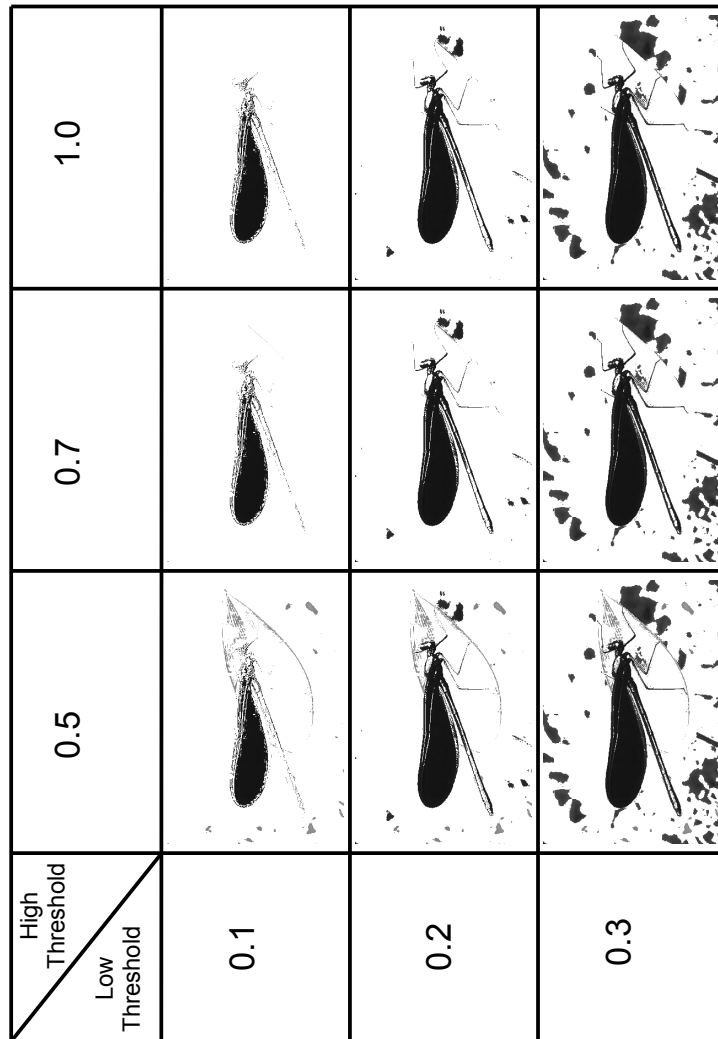


Figure 4-3: Variation of segmentation results with parameter settings for relative thresholding algorithm.

#### 4.4 Edge detection algorithms parameters

---

finding the initial set of edges with the maximal threshold parameter, by looking at neighbouring pixels to the ‘edge’-marked pixels and considering if they can be part of the edges. The maximal threshold parameter can be considered similar to the low threshold parameter in the thresholding techniques, and, likewise, the ‘minimum threshold’ parameter corresponds to the high threshold parameter in the thresholding techniques. In summary, this method first finds all local maxima in the image, and suppresses the rest. In the second stage, neighbouring pixels of the detected maxima -in the first stage- can be added to the segments if their value exceeds the lower threshold value.

Figure 4.5 illustrates the variation within another widely used edge detection technique: the Canny Edge detection algorithm. Compared to the previous classic implementation, which only considers the grey-scale magnitude of the input image, this Canny-edge implementation is enhanced to also compute the RGB colour components with a colour contrast gradient function (see earlier discussion in Chapter 3).

The same two parameters are used for the non-maxima suppression stage. It is clear that the maximal threshold parameters (like the low threshold parameter) have the greater effect on the final segmentation results, especially in reducing the ‘noise’ from the segments found. However, edge detection techniques have the very big disadvantage of actually being ‘edge detection’ techniques, rather than full segmentation techniques, and as such the final results are more like ‘unclosed segments’ (corresponding to edges). When it’s possible to add a ‘gap filling’ stage to produce a true segmented final result, that stage can prove to be complex and less-efficient. Still, edge detection techniques are useful as fast techniques for image feature detection. Additionally, further analysis of detected

#### 4.4 Edge detection algorithms parameters

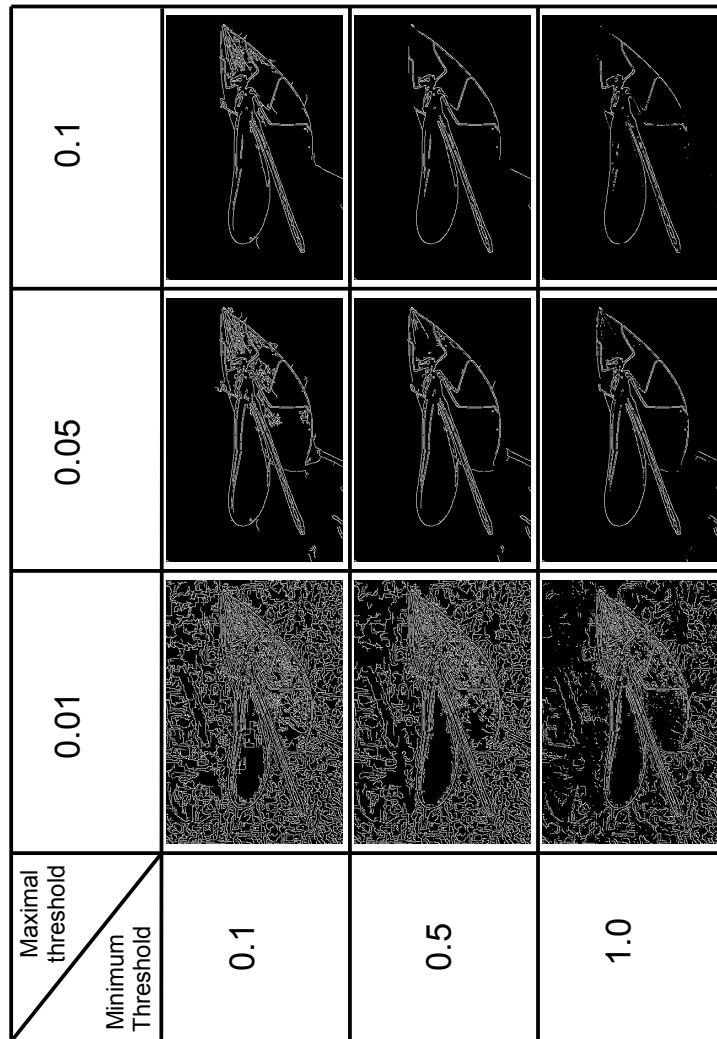


Figure 4.4: Variation of segmentation results with parameter settings for classic edge detection algorithm.

#### 4.4 Edge detection algorithms parameters

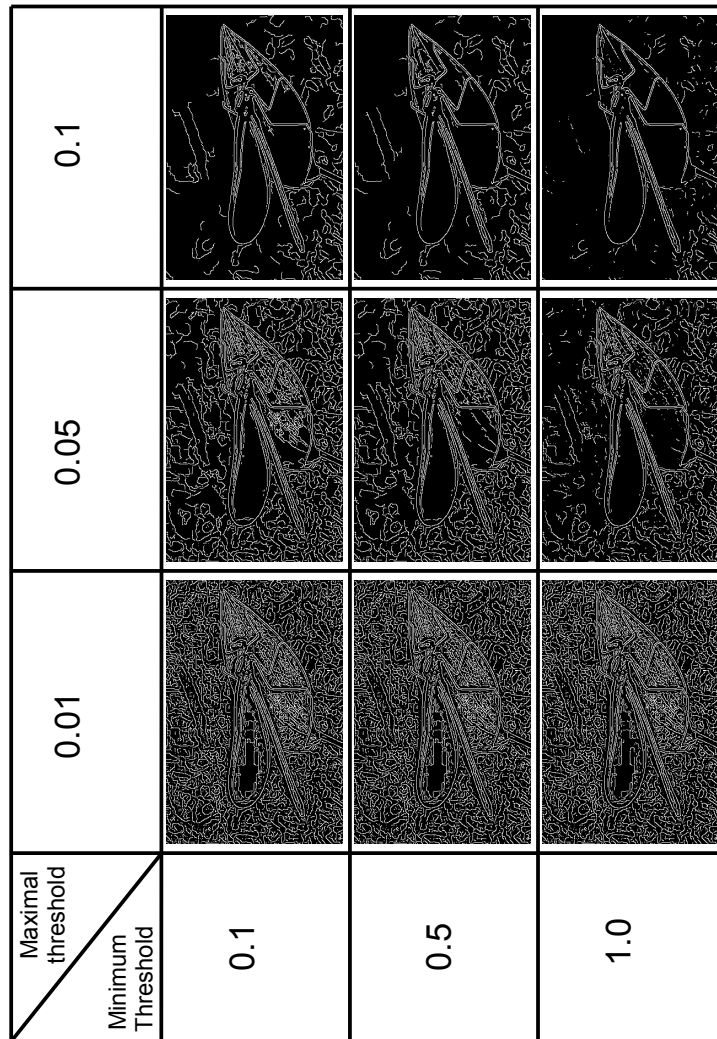


Figure 4.5: Variation of segmentation results with parameter settings for canny Edge Detection algorithm.



---

## 4.5 Region growing segmentation parameters

---

edges can help in detecting ‘corners’ which enhances feature detection. They are still widely used for embedded systems (because of their suitability for real-time implementation).

## 4.5 Region growing segmentation parameters

---

As an example of region-growing-based techniques, a basic algorithm that uses a Gaussian kernel of size  $5 \times 5$  will be illustrated, focusing on separating an object from its background. The two significant parameters for this algorithm are the average threshold and the edge threshold. For the effect of the parameters see Figure 4.6. The average threshold is used to grow the region while searching and comparing unlabelled pixels with the seed position. If the difference of values between the parameters is lower than the threshold then it is considered as part of the growing region. The edge threshold parameter is part of the edge detection stage (see Section 3.5) which produces the initial regions that will be grown.

It is clear that the significant parameter here is the growing threshold, which defines how much of the image will be considered part of the detected object. The edge detection is still important to define the starting edges for growing regions for the final results. There is also a smoothing stage used in this algorithm. However, this stage does not affect the final results as much it does in other algorithms. What is important to notice here is the use of basic image-processing filters (like smoothing filters) and basic image segmentation techniques to construct more ‘advanced’ segmentation techniques. How the parameters affect the final segmentation results actually relates to those components.

## 4.5 Region growing segmentation parameters









Edges Threshold / Avg. Threshold	0.01	0.05	0.5
	0.02		
0.05			
0.1			

Figure 4.6: Variation of segmentation results with parameter settings for classic Region Growing algorithm.

## 4.6 Watershed segmentation parameters

---

Watershed algorithms use a similar concept to the the region-growing techniques but are considered in a category by their own. Experiments with the classic Watershed algorithm (Vincent & Soille, 1991) also gave rise to a variety of results, depending on the choice of parameters. A colour quantisation or smoothing stage was added as a pre-processing step before application of the Watershed, which is usually not part for the implementation. However, it's introduced here to see how it affects the final segmentation results and it is compared with other algorithms that use a smoothing filter as a pre-processing stage. In Figure 4.7: TR is the threshold; D is the  $\delta$  (delta) threshold; I is the number of iterations; and C is the number of colours after colour quantisation.

The threshold parameter here has a significant effect on the final segmentation result. However, colour quantisation as a pre-processing stage (which can be basically considered as a smoothing filter) also has a significant effect on the final result, a result that was also clear in the previous chapter.

Figure 4.8 shows the number of detected segments and the processing time for the same parameters varied in Figure 4.7. It's clear how the pre-processing stage produces a lower number of regions and reduces the processing time significantly. When a lower number of segments are found in the end, this doesn't necessarily correspond to a 'better' segmentation overall. It's important to notice that some decision needs to be taken for some situations to balance between the final segmentation quality and the computation performance needed to complete the segmentation process. The aim is to achieve the best segmentation quality possible while still achieving this in a computationally efficient way. It's impor-

#### 4.6 Watershed segmentation parameters

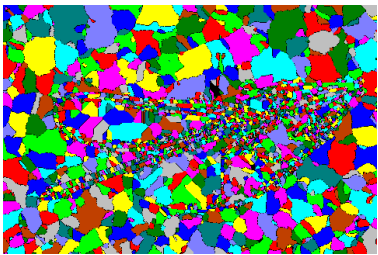
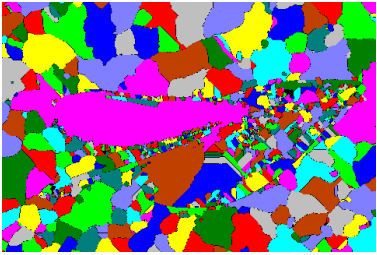
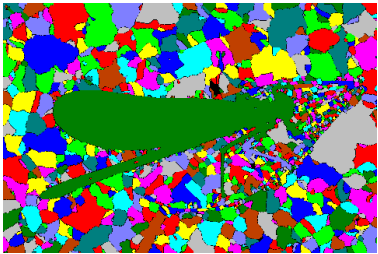
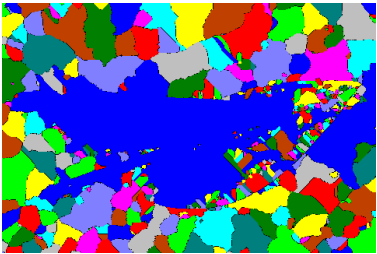
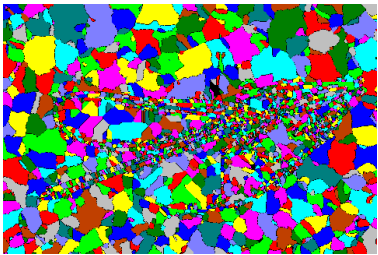
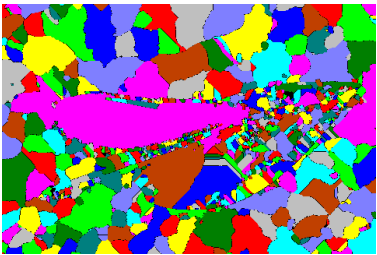
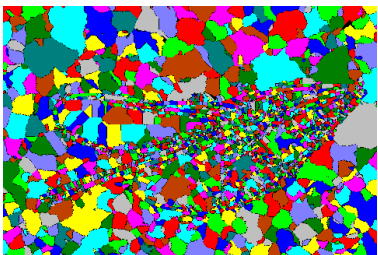
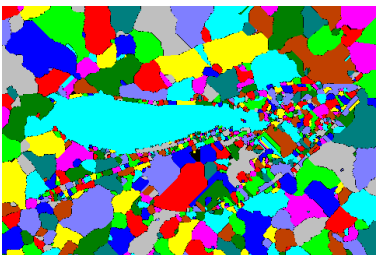
Th	No. of C		256	16
	D	I		
1	0.1	1		
60	0.1	1		
1	1	1		
1	0.1	100		

Figure 4.7: Variation of segmentations with parameter settings for Watershed segmentation (Vincent & Soille, 1991).

#### 4.6 Watershed segmentation parameters

Th	No. of C		256	16
	D	I		
1	0.1	1	Regions found: 1850 ..... Processing Time: 7257.165 ms	Regions found: 714 ..... Processing Time: 3401.433 ms
60	0.1	1	Regions found: 1071 ..... Processing Time: 6290.349 ms	Regions found: 433 ..... Processing Time: 3387.439 ms
1	1	1	Regions found: 1850 ..... Processing Time: 7205.413 ms	Regions found: 714 ..... Processing Time: 3163.131 ms
1	0.1	100	Regions found: 1905 ..... Processing Time: 60889.429 ms	Regions found: 716 ..... Processing Time: 6780.433 ms

Figure 4.8: Variation of segmentations for the same parameter's settings used in Fig. 4.7 with parameter settings for Watershed segmentation showing the number of segments found and the processing time.

---

## 4.7 Rain-falling segmentation parameters

---

tant to note that the lower number of segments found is also used as a factor in some of the objective evaluation techniques.

Figure 4.9 illustrates the results again when no smoothing filtering is used at all in pre-processing and the results with filtering with different threshold values. And Figure 4.10 illustrates the number of segments found and the processing time for the same results. Overall while the pre-processing stage has a significant effect on the result (especially computation performance wise), it's not the only factor in producing 'better' segmentation results in all situations. 'Better' here means a better segmentation quality according to the evaluation criteria used: either subjective or objective, supervised or unsupervised. However, other processing stages can be added (like post-processing stages) to improve the quality of the final segmentation results and also can improve the performance, depending on the segmentation requirement for the application.

## 4.7 Rain-falling segmentation parameters

---

Rain-falling is another type of implementation for the watershed segmentation technique. In Figure 4.11, the same set of parameters used with the Watershed algorithm are also used here with the rain-falling technique.

Similar to what was done with the Watershed technique, a smoothing filter stage was added as a pre-processing step before applying the segmentation, which is usually not part of the implementation. Again, the colour quantisation stage has a significant effect on the final result, especially in respect to performance. However, the threshold parameter plays a more significant effect on the quality of the final segmentation result. Figure 4.12 shows the number of detected seg-

#### 4.7 Rain-falling segmentation parameters

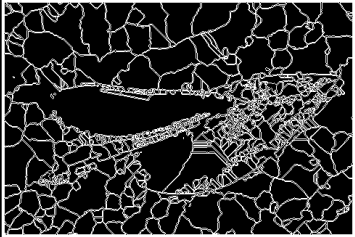
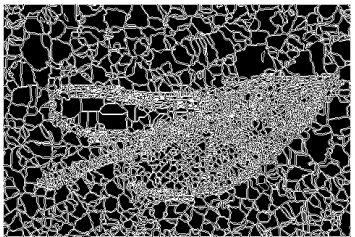
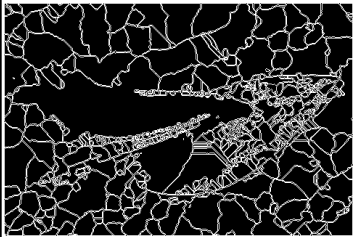
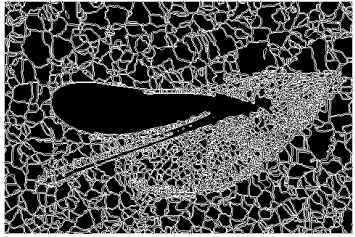
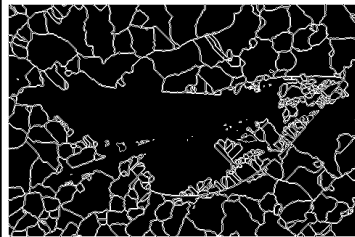
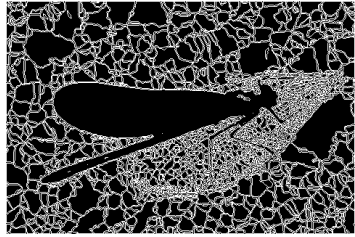
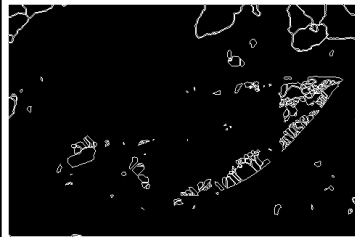
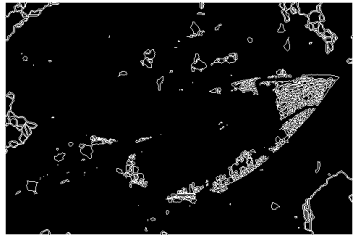
Threshold \ No. of C	16 (D = 0.1, I = 1)	Without Filtering
1		
30		
60		
100		

Figure 4.9: Variation of segmentation results similar to Fig. 4.7 with parameter settings for Watershed segmentation that show the difference between the results with and without a pre-processing stage.

#### 4.7 Rain-falling segmentation parameters

Threshold \ No. of C	16 (D = 0.1, I = 1)	Without Filtering
1	Regions found: 714 ..... Processing Time: 3276.418 ms	Regions found: 2918 ..... Processing Time: 5446.431 ms
30	Regions found: 651 ..... Processing Time: 3112.435 ms	Regions found: 2310 ..... Processing Time: 4425.426 ms
60	Regions found: 433 ..... Processing Time: 2804.423 ms	Regions found: 1902 ..... Processing Time: 4476.432 ms
100	Regions found: 208 ..... Processing Time: 2482.414 ms	Regions found: 641 ..... Processing Time: 2536.439 ms

Figure 4.10: Variation of segmentations for the same parameter's settings used in Fig. 4.9 with parameter settings for Watershed segmentation and show the number of segments found and the processing time.



#### 4.7 Rain-falling segmentation parameters

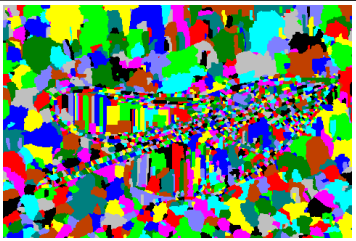
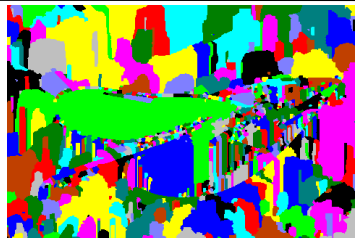



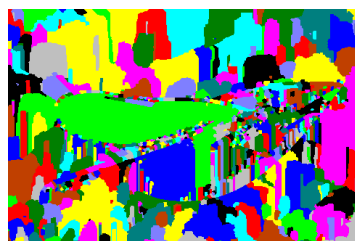
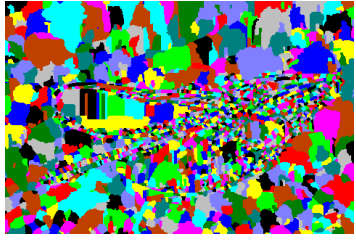
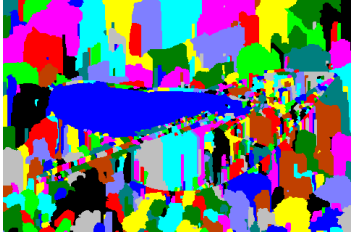
Th	No. of C		256	16
	D	I		
1	0.1	1		
60	0.1	1		
1	1	1		
1	0.1	100		

Figure 4.11: Variation of segmentations with parameter settings for Rain-falling segmentation

---

## 4.8 Colour Watershed segmentation parameters

---

ments and the processing time for the same parameters variation in Figure 4.11. Similar to the Watershed results the quantisation stage produces a lower number of regions and reduces the processing time significantly.

Figure 4.13 illustrates the results again when no filtering is used at all in pre-processing and the results with filtering with different threshold values. And Figure 4.14 illustrate the number of segments found and the processing time for the same results as in Figure 4.13. Overall, while the pre-processing stage has a significant effect on the result, it's not the only factor in producing 'better' segmentation results in all situations. However other processing stages can be added (like post-processing stages) to improve the quality of the final segmentation results and also the performance depends on the segmentation requirement for the application used.

## 4.8 Colour Watershed segmentation parameters

---

The colour watershed segmentation implementation used here (Alvarado, 2004) extends classical watershed segmentation. It uses three main stages: a pre-processing median filtering stage to smooth and reduce the noise in the colour channels, process the results with a Watershed segmentation stage to produce a segmentation result (over-segmented preferably), and then uses a post-processing region merging stage. Over-segmentation is preferred by the algorithm as the merging stage in almost all cases produces a significantly better quality final segmentation result.

The three significant parameters relate directly to the three processing stages: firstly, the Kernel size of the median filtering stage ( $Ks$ ), secondly, the threshold

#### 4.8 Colour Watershed segmentation parameters

Th	No. of C		256	16
	D	I		
1	0.1	1	Regions found: 1851 ..... Processing Time: 7737.449 ms	Regions found: 711 ..... Processing Time: 4281.337 ms
60	0.1	1	Regions found: 1074 ..... Processing Time: 5740.287 ms	Regions found: 432 ..... Processing Time: 4229.417 ms
1	1	1	Regions found: 1851 ..... Processing Time: 6301.440 ms	Regions found: 711 ..... Processing Time: 4296.431 ms
1	0.1	100	Regions found: 1904 ..... Processing Time: 63835.409 ms	Regions found: 713 ..... Processing Time: 7098.428 ms

Figure 4.12: Variation of segmentations for the same results in Fig. 4.11 with parameter settings for Rain-falling segmentation and show the number of segments found and the processing time.

#### 4.8 Colour Watershed segmentation parameters


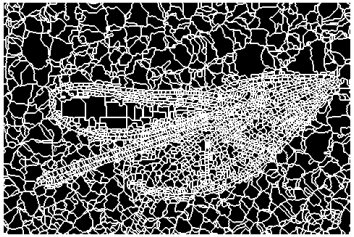

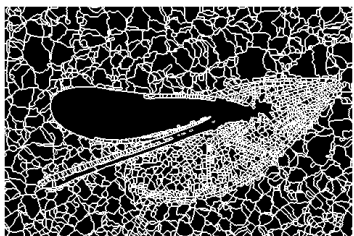
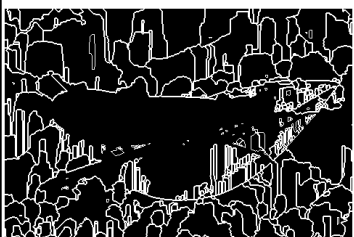
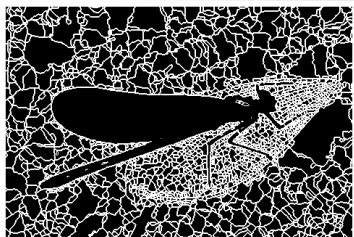


Threshold \ No. of C	16 (D = 0.1, I = 1)	Without Filtering
1		
30		
60		
100		

Figure 4.13: Variation of segmentations of the same results in Fig. 4.11 with parameter settings for Rain-falling segmentation that show the difference between the results with pre-processing stage and no filtering.

#### 4.8 Colour Watershed segmentation parameters

Threshold \ No. of C	16 (D = 0.1, I = 1)	Without Filtering
1	Regions found: 711 ..... Processing Time: 4649.417 ms	Regions found: 2903 ..... Processing Time: 4160.427 ms
30	Regions found: 648 ..... Processing Time: 4590.427 ms	Regions found: 2297 ..... Processing Time: 4068.429 ms
60	Regions found: 432 ..... Processing Time: 4474.442 ms	Regions found: 1894 ..... Processing Time: 3117.424 ms
100	Regions found: 207 ..... Processing Time: 4413.423 ms	Regions found: 643 ..... Processing Time: 2307.419 ms

Figure 4.14: Variation of segmentations of the same results in Fig. 4.13 with parameter settings for rain-falling segmentation and showing the number of segments found and the processing time.

---

## 4.9 K-means segmentation parameters

---

parameter for the watershed segmentation stage (Th), and thirdly the merge threshold parameter used in the merging stage. Figure 4.15 illustrates the effect of varying the different parameters on the final segmentation results.

A kernel size value of 1 means that no filtering was done, and it's clear that the median filtering has some effect on the final results to lower the noise and the number of detected segments. The Watershed segmentation threshold has a more significant effect on defining the segments as the objects in the input image. Overall the merging stage produces the most significant effect on the final segmentation results, taking into consideration the performance improvement and the quality of the final segmentation. See Figure 4.16 for an illustration of the processing time for each case in Figure 4.15.

## 4.9 K-means segmentation parameters

---

K-means segmentation is a clustering segmentation technique. It employs a k-mean clustering algorithm for colour quantisation that utilises the pixels' RGB colour values to measure the weighted distance measure and create an optimised colour palette (colour clusters) for the input image as a first step. Then the implementation processes the quantised result into a second stage image smoothing filter.

The implementation has four parameters related to the processing stages: first the number of colours for the quantised image result (No. of C), the threshold delta (D) and the maximum number of iteration (I) parameter, which is used by the k-means colour quantisation stage, and the kernel size parameter (K) for the smoothing filter stage. Figure 4.17 illustrates the effect of varying the different

#### 4.9 K-means segmentation parameters

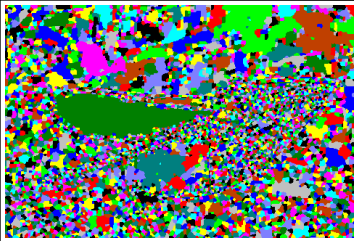
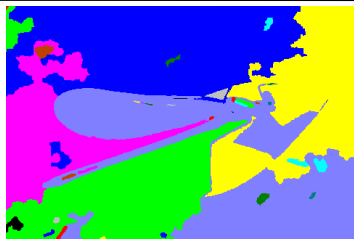
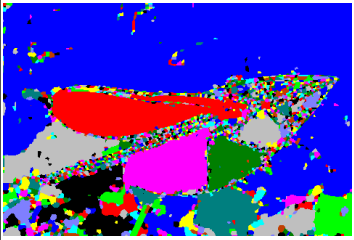
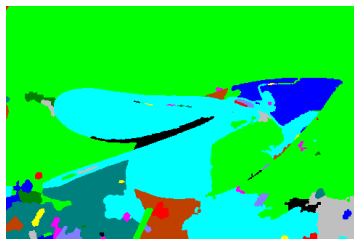
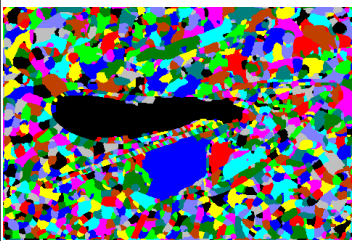


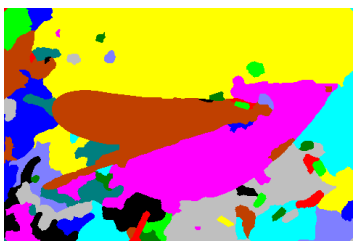
		Merge Thresh.	
		0.0	0.5
Th	Ks		
0.1	1		
0.5	1		
0.1	10		
0.5	10		

Figure 4.15: Variation of segmentations with parameter settings for Colour Watershed segmentation

#### 4.9 K-means segmentation parameters

		Merge Thresh.	
		0.0	0.5
Th	Ks		
0.1	1	<b>Regions found:</b> <b>5163</b> ..... <b>Processing Time:</b> <b>9775.426 ms</b>	<b>Regions found:</b> <b>43</b> ..... <b>Processing Time:</b> <b>3378.292 ms</b>
0.5	1	<b>Regions found:</b> <b>1938</b> ..... <b>Processing Time:</b> <b>5077.348 ms</b>	<b>Regions found:</b> <b>69</b> ..... <b>Processing Time:</b> <b>3778.433 ms</b>
0.1	10	<b>Regions found:</b> <b>1760</b> ..... <b>Processing Time:</b> <b>5838.343 ms</b>	<b>Regions found:</b> <b>56</b> ..... <b>Processing Time:</b> <b>4625.009 ms</b>
0.5	10	<b>Regions found:</b> <b>597</b> ..... <b>Processing Time:</b> <b>5056.425 ms</b>	<b>Regions found:</b> <b>70</b> ..... <b>Processing Time:</b> <b>4575.429 ms</b>

Figure 4.16: Variation of segmentations for the same results in Fig. 4.15 with parameter settings for Colour Watershed segmentation and show the number of segments found and the processing time.



---

## 4.10 Mean-shift segmentation parameters

---

parameters on the final segmentation results.

Overall, the number of colour parameters is the most significant parameter in respect to the final segmentation result. While the implementation uses the pixels' RGB values to produce the clusters other image features can be used too (like the image texture for example). The only other significant parameter is the kernel size which is used by the smoothing filter and as a result lowers significantly all the 'noise' segments found. It's clear that although the number of regions being found is lower (reduced from 2615 regions to 401 regions) this didn't affect the large segments corresponding to the objects in the image originally found by the quantisation stage (recall that a kernel size of value 1 effectively means that no filtering was performed). See Figure 4.18 for an illustration of the regions found values for each case in Figure 4.17.

## 4.10 Mean-shift segmentation parameters

---

Mean-shift is another clustering-based segmentation technique. The technique also uses a similar concept of starting with a colour quantisation stage, and then has a region fusing stage to smooth the segmentation result. Finally, there is a pruning stage that prunes segments of a given size.

Fig. 4.19 shows the results of varying the mean-shift parameters. Parameters `radiusS` and `radiusR` relate to the mean-shift search sphere used in the first filtering stage. `radiusS` is the sphere range radius in the image grid space, while `radiusR` is the range radius in the image colour space. Higher values of `radiusR` result in less regions, while higher values of `radiusS` effectively results in more computation but smoother region boundaries. The colour distance parameter is

#### 4.10 Mean-shift segmentation parameters

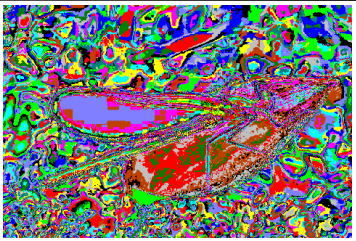
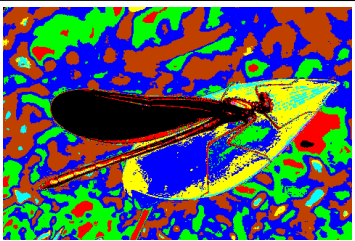
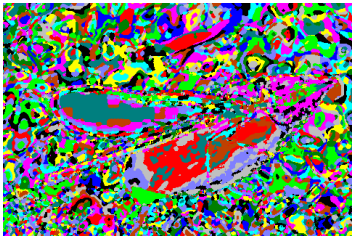
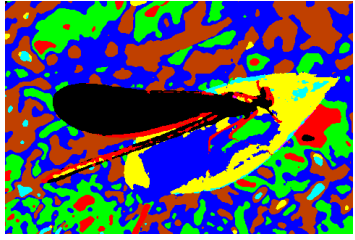
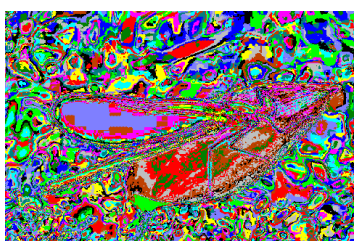
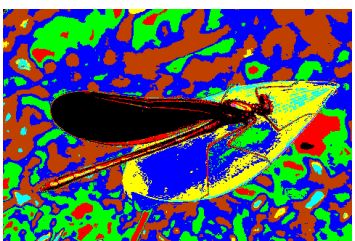
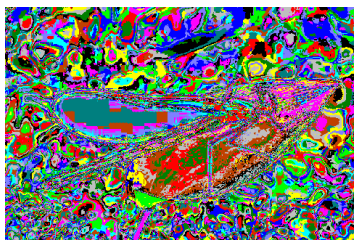
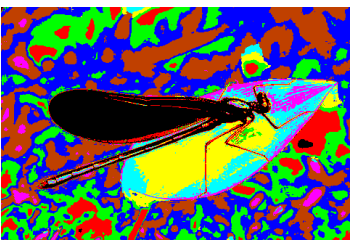
K	No. of C		256	8
	D	I		
1	0.1	1		
5	0.1	1		
1	1	1		
1	0.1	10		

Figure 4.17: Variation of the segmentation results with parameter settings for K-means segmentation

#### 4.10 Mean-shift segmentation parameters

K	No. of C		256	8
	D	I		
1	0.1	1	Regions found: 20961 ..... Processing Time: 12118.933 ms	Regions found: 2615 ..... Processing Time: 2670.287 ms
5	0.1	1	Regions found: 6591 ..... Processing Time: 9416.421 ms	Regions found: 401 ..... Processing Time: 2219.431 ms
1	1	1	Regions found: 20961 ..... Processing Time: 11297.134 ms	Regions found: 2615 ..... Processing Time: 2754.432 ms
1	0.1	10	Regions found: 21094 ..... Processing Time: 23287.266 ms	Regions found: 2453 ..... Processing Time: 3154.304 ms

Figure 4.18: Variation of segmentations for the same results in Fig. 4.17 with parameter settings for K-means segmentation and show the number of segments found and the processing time.

---

## 4.11 Graph-based segmentation parameters

---

used by the fusing stage. In the fusing stage, segments with a colour difference less than this parameter are combined. Segmented regions with colour difference less than this parameter are fused together. There is one more parameter that relates to the pruning stage for the minimum region size that in practice doesn't make a big change to the final result compared to the parameters above. However, it will become clear that pruning as a post-processing stage can be very effective in other algorithms.

Overall, the significant parameter here is the colour distance parameter, which not only 'smoothes' the resultant segmentation significantly but also reduces the computation time. See Figure 4.20 for an illustration of the regions found values for each case in Figure 4.19.

## 4.11 Graph-based segmentation parameters

---

A graph-based segmentation implementation (Felzenszwalb & Huttenlocher, 2004) also displayed a similar dependence on choice of parameters. This implementation has three main stages: firstly a smoothing filter stage, secondly a graph-building and segmenting stage based on the dissimilarity between the RGB pixel values, and finally a pruning stage according to a given minimum segmented regions' sizes. There are three input parameters that relate to the different processing stages in the algorithm: 1)  $\sigma$  (sigma), the variance of a Gaussian smoothing filter prior to segmentation; 2)  $k$ , which sets the scale of the components found through a graph-based thresholding function; and 3)  $Min$ , which is the minimum component size enforced by the post-processing pruning stage.

Figure 4.21 shows the result of varying the different parameters. The most

#### 4.11 Graph-based segmentation parameters

		Colour Distance		1	10
		rdS	rdR		
1	1				
10	1				
1	10				
10	10				

Figure 4.19: Variation of segmentations results with parameter settings for Mean-shift segmentation

#### 4.11 Graph-based segmentation parameters

		Colour Distance	
		rdS	rdR
		1	10
1	1	Regions found: 14288 ..... Processing Time: 40266.420 ms	Regions found: 496 ..... Processing Time: 3987.416 ms
10	1	Regions found: 13466 ..... Processing Time: 66843.442 ms	Regions found: 504 ..... Processing Time: 31001.428 ms
1	10	Regions found: 14288 ..... Processing Time: 44775.763 ms	Regions found: 496 ..... Processing Time: 4205.045 ms
10	10	Regions found: 5690 ..... Processing Time: 47535.416 ms	Regions found: 320 ..... Processing Time: 37388.433 ms

Figure 4.20: Variation of segmentations for the same results in Fig. 4.19 with parameter settings for Mean-shift segmentation and show the number of segments found and the processing time.

---

## 4.12 Anisotropic-based segmentation parameters

---

significant outcome to notice here is that the pruning stage variation alone controlled by the *Min* parameter, significantly reduces the over-segmentation even with low smoothing and thresholding values. Whereas it's not a complete solution on its own, depending on the final segmentation requirement of the computer vision application, the pruning stage can provide a useful and easy control point for the final segmentation quality, even with a fast segmentation algorithm that can produce over-segmented results. Additionally, a longer iteration within the core of the segmentation algorithm itself will most likely be less effective, while increasing the computation time compared to adding a post-processing stage (like the pruning stage), provided that it's integrated effectively.

See Figure 4.22 for an illustration of the regions found values for each case in Figure 4.21.

## 4.12 Anisotropic-based segmentation parameters

---

The anisotropic diffusion-based segmentation implementation used herein has a limited number of exposed parameters that can be varied by the user: 1)  $K$ , is used as the threshold for the anisotropic diffusion process that produces an image gradient result; and 2)  $I$ , which is the number of iterations to repeat the anisotropic diffusion process stage on the image. The algorithm actually has one necessary step after the diffusion that finds and labels the segments' edges using the diffused image result from the first stage (in the process using a modified non-maxima suppression algorithm and edge labelling stage). However, both of these stages are significantly dependent on the result of the input they receive from the diffusion stage and are computationally cheap compared to the diffusion

#### 4.12 Anisotropic-based segmentation parameters


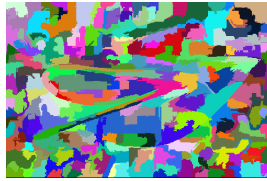


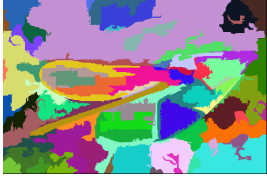






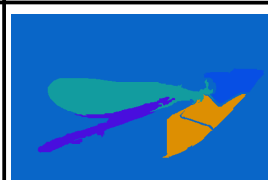
Sigma	Min	20	200	2000
	K			
0.1	50			
0.1	500			
1	50			
1	500			

Figure 4.21: Variation of segmentations' results with parameter settings for Graph-based segmentation



#### 4.12 Anisotropic-based segmentation parameters

Sigma	Min	20	200	2000
	K			
0.1	50	Regions found: 895 ..... Processing Time: 2586.234 ms	Regions found: 252 ..... Processing Time: 2554.275 ms	Regions found: 22 ..... Processing Time: 2435.431 ms
	500	Regions found: 178 ..... Processing Time: 2542.412 ms	Regions found: 75 ..... Processing Time: 2529.438 ms	Regions found: 29 ..... Processing Time: 2533.380 ms
1	50	Regions found: 351 ..... Processing Time: 3537.426 ms	Regions found: 89 ..... Processing Time: 3555.442 ms	Regions found: 17 ..... Processing Time: 3540.426 ms
	500	Regions found: 47 ..... Processing Time: 3659.556 ms	Regions found: 15 ..... Processing Time: 3546.438 ms	Regions found: 5 ..... Processing Time: 3531.339 ms

Figure 4.22: Variation of segmentations for the same results in Fig. 4.21 with parameter settings for Graph-based segmentation showing the number of segments found and the processing time.

## 4.12 Anisotropic-based segmentation parameters

stage.

Figure 4.23 shows the results of varying the two parameters. Both of the parameters shows an effective influence on the final segmentation results. However, as expected, a higher value of the iteration parameter generally results in a higher processing time to complete the segmentation. See Figure 4.24 for an illustration of the regions found values for each case in Figure 4.23.


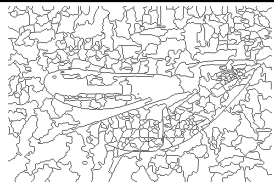
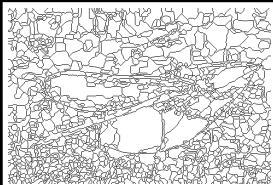
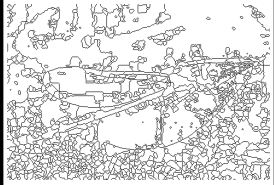

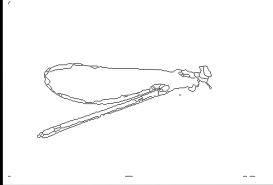
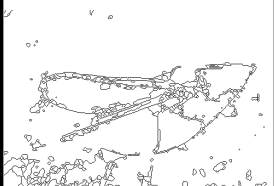
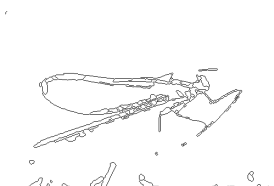

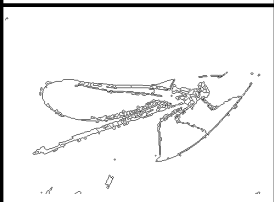
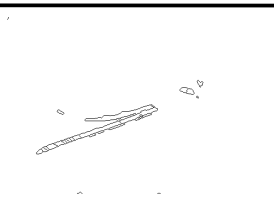

Iteration \ K	1	10	100
1			
5			
10			
20			

Figure 4.23: Variation of segmentations' results with parameter settings for Anisotropic-based segmentation

## 4.13 Concluding Remarks

Iteration k	1	10	100
1	Regions found: 1117 ..... Processing Time: 43331.529 ms	Regions found: 1170 ..... Processing Time: 27897.224 ms	Regions found: 1065 ..... Processing Time: 131905.464 ms
5	Regions found: 1133 ..... Processing Time: 22039.141 ms	Regions found: 668 ..... Processing Time: 23807.031 ms	Regions found: 75 ..... Processing Time: 121175.934 ms
10	Regions found: 447 ..... Processing Time: 11963.444 ms	Regions found: 251 ..... Processing Time: 20304.131 ms	Regions found: 4 ..... Processing Time: 121265.342 ms
20	Regions found: 177 ..... Processing Time: 7697.458 ms	Regions found: 52 ..... Processing Time: 17619.539 ms	Regions found: 0 ..... Processing Time: 126647.921 ms

Figure 4.24: Variation of segmentations for the same results in Fig. 4.23 with parameter settings for Anisotropic-based segmentation and show the number of segments found and the processing time.

## 4.13 Concluding Remarks

Several researchers have highlighted the use of parameters to find better quality segmentation results in the last 10 years: Everingham et al. in (Everingham *et al.*,

2002a,b), Chabrier et al. in (Chabrier *et al.*, 2005a,b, 2008), and the Zhang et al. group (Zhang *et al.*, 2005, 2006). This will be explored in more detail in Section 6.1.

All of these researchers touched on using segmentation evaluation with the help of machine-learning/artificial intelligence techniques (with a focus on genetic algorithms). The focus in their work was on improving the evaluation performance overall, which we also highlighted in our work. In some of the work, tests were carried out on parameters (Chabrier *et al.*, 2005a; Everingham *et al.*, 2002a) to achieve the best parameters relating to the best segmentation results.

However, in all of the research the importance of linking the parameters to what they relate to the inner working of the segmentation algorithms, and what they affect in the results of the segmentation results is not discussed. Moreover, how the parameters actually relate to the basic pre/post-processing image filtering components that makes the typical segmentation results is not highlighted. The author has endeavoured to highlight these processing stages and how they are related to the input parameters.

The research carried out uncovered two distinctive set of parameters that affect the segmentation algorithm output quality and computation performance: a) pre-processing stage(s) parameters and b) post-processing stage(s) parameters. Although almost all the algorithms researched in this work had not more than one stage at each stage there is nothing that prevents the design from adding multiple number of stages before or after the core processing stage.

The act of adding pre/post-processing stages is by itself a step into improving the quality/time performance of the segmentation algorithm even with parameter-less processing stages. In addition to being an improvement, the stages with pa-

## 4.13 Concluding Remarks

---

rameters can help to act as a controlling stage for the quality of the segmentation results, so for example a segment-pruning stage after any segmentation algorithm can provide a useful stage to control the over-segmentation for example. So the core processing stage that defines the segments' boundaries need only to provide a very fast and coarse segmentation results that are highly over-segmented and the pruning-stage can be used to determine how much segments are in the final result according to the input parameters. In this case there is no need to spend a high computation time on the core processing stage and the pruning stage is relatively cheap computationally. The pruning stage itself can be as simple as an algorithm that rejects certain segments according to their size to a more complex algorithm that creates a representative tree of the segments and order/combines them according to their features similarity (colour, texture, or others) (Salembier & Garrido, 2000).

Even with a complex tree-based pruning stage, its easier to computationally optimise these stages using methods like parallel/distributed solutions either with cluster computers or on recent advanced multi-core processors-based computers, whereas there is always a bottle-neck with the core boundary-defining stage of the typical segmentation algorithm even if the image parts are distributed to different processing nodes (one to each cluster computer or processor core), the bottle-neck will always partially be in the message-passing. Message-passing in the boundary-defining stage will always be present because in the instance that the image is divided into parts for processing then there will be a stage where the nodes processing these parts will need to communicate to at least define the boundaries at the edges of the image parts. Despite the recent and continuous development and improvement in the computers hardware buses (on the computer

## 4.13 Concluding Remarks

---

motherboards and between the processors cores) and connections (between the different hardware parts and over the networks), there is always a need to process even higher numbers of images (or video frames) in higher quality and as such this message-passing bottle-neck will always be a disadvantage.

If what is needed is some suggestion of the parameter values for segmentation, then there is no definite answer that works for all applications, as might be anticipated. However, as this research used extensively the hand-segmented images as ground truths, then some suggestions can be provided for applications with similar features. If the application uses natural images as input and need to decrease the over-segmentation in the output as much as possible then smoothing filters at the pre-processing stages and pruning as an example of post-processing stages will provide a good measure to achieve the required results.

This research focused on colour-based segmentation algorithms and in this case most of the smoothing stages depend on the colour of the input results, that reduces the noise in the image while preserving the boundaries of the objects. This can enhance the image structure at different scales before finding the segments' boundaries. So a good value for the smoothing filter was always lower than 25 colours for colour-rich images, and even lower from 16 to 8 for images with a limited palette. Similarly, the pruning stage needs to lower the number of stages in the end so typically the parameter values need to emphasise the goal of decreasing the over-segmentation depending on the type of the pruning stage used. For example, if the pruning stage eliminates the segments according to their size, then the parameter should reflect that segments with low sizes should be eliminated. The results in this chapter were only illustrative and further empirical evaluation on larger image datasets and parameters range will be provided in the

coming chapters.

Nevertheless, the results don't justify choosing a single or a range of parameters' values that work in all the cases. In conclusion, the author thinks that a segmentation framework that continuously evaluates the segmentation results and adjusts the segmentation results using a machine-learning techniques is a best solution as discussed earlier. This can be achieved by combining supervised and unsupervised segmentation evaluation methods in the evaluation framework. The evaluation should not only cover the parameters but also can include the processing stages used, including adjusting the number and type of these stages as needed. Further discussion on this will be done in the concluding chapter.

# 5

## Evaluation framework for automatic generation of tests

The sciences do not try to explain, they hardly even try to interpret, they mainly make models.

---

Johann Von Neumann

### 5.1 Introduction

---

The image-processing and computer vision communities have developed both *de jure* (e.g. , IPI-PIKS (Pratt, 2001)) and *de facto* (e.g. , IUE, Target Jr, VXL<sup>1</sup>, RAVL, Tina, ITK<sup>2</sup> (Ibanez *et al.*, 2005), and the Intel OpenCV Library<sup>3</sup>) facilities, containing collections of algorithms. In particular, ITK hosts the code for

---

<sup>1</sup>VXL (the Vision-something-Libraries): <http://vxl.sourceforge.net>

<sup>2</sup>The Insight Segmentation and Registration Toolkit: <http://www.itk.org>

<sup>3</sup>OpenCV Open Source Computer Vision Library: <http://opencvlibrary.sourceforge.net/>



numerous image segmentation algorithms intended for medical applications.

The value of these would be enhanced if there existed a direct route to selecting one or more algorithms for a particular application, as it is difficult to assess the suitability of a segmentation algorithm without detailed comparison through testing. Contrasting image segmentation to recognition tasks such as the use of handwriting, and face databases, the authors of (Martin *et al.*, 2001) remark “Typically [in segmentation] researchers will show their results on a few images and point out why the results ‘look good’”. Part of the problem is the difficulty of establishing metrics, whether pixel-based figures of merit (Yasnoff *et al.*, 1977) or rankings against a set of criteria (Everingham *et al.*, 2001) or through precision-recall curves (Martin *et al.*, 2004). However, part of the problem may also be the logistics of performing a large number of tests. Additionally, a user needs guidance in the form of structure, to navigate an evaluation framework.

There has been an emphasis in the past on algorithmic novelty, whereas increasingly the importance of systematic validation on sufficient datasets is now stressed. For a given input, the corresponding correct output is also known, either because the input is simulated or by virtue of either expert opinion or the presence of a ground truth such as hand segmented images. The developer aspires to make the algorithm agree with the correct output for every corresponding input. Implicit in this procedure is that the finite set of test data is representative of variation in the real world and that the number of samples is large enough (Guyon *et al.*, 1998).

---

## 5.2 Evaluation Environment

---

To provide the systematic validation mentioned in the introduction, the empirical evaluations need three foundational bases to be accepted by the research community, and subsequently required by all future research in the field to act as a reference point. The three bases are: 1) standard databases 2) evaluation protocols 3) scoring methods. At the moment there are many examples for each of these points in the computer vision research. In the case of image segmentation, which is the focus of this research, there are not many examples but good progress has already occurred that can be used to build on further research.

For example, the Berkeley group image segmentation dataset (BSDS) (Martin *et al.*, 2001) is a good example of a standard database that can be used for evaluation, other publicly available dataset examples include MSRC and PASCAL 2008. The MSRC dataset (Shotton *et al.*, 2006) is designed as an object recognition database but can also be used for image segmentation evaluation. MSRC is composed of 591 natural images with objects belonging to 21 classes. PASCAL 2008 (Everingham *et al.*, 2008) is another well-known object recognition database used in the PASCAL yearly competition. Part of this database has image datasets for segmentation evaluation which are composed of 1023 images. This can be considered one of the most difficult and varied datasets for recognition. However, the BSDS remains the most complete dataset available for our purposes. It has been used in several publications, and has the advantage of providing several human-labeled segmentations per image. Further details of the Berkeley dataset will be given later in the chapter.

The two other bases: the evaluation protocols and the scoring methods define

the evaluation framework and the related evaluation methods respectively. The evaluation methods were explored in more detail in section 2.4. This section will focus on the evaluation protocol or as could be better be defined as an evaluation framework. There have been many research endeavours that advanced the area of comparative evaluation of algorithms. Bowyer & Phillips in the introduction to their book (Bowyer & Phillips, 1998a) argue that the benefits of such approach would include: 1) providing a solid scientific and experimental basis for computer vision research, (2) help researchers to develop engineering solutions to practical problems, (3) obtaining an accurate assessment of the state-of-the art research, and (4) give enough convincing evidence to the potential users to help them distinguish a practical solution to their problems from the evaluated computer vision research.

In our case, and in light of the benefits that can be gained as stated above, the evaluation framework that will be explored in the following sections will try to advance the research in the colour image segmentation field. The aim is to explore further both the segmentation quality performance and the computational performance of the representative algorithms under the evaluation. Moreover, the aim is to explore the evaluation from a parameter-oriented viewpoint.

### 5.3 Using and extending FATE

---

The first component of the evaluation environment is a scripting language harness or structure for running tests called FATE (Framework for Algorithm Testing and Evaluation). FATE is an newer version of HATE: Harness for Algorithm Testing and Evaluation developed and enhanced by the original developers Courtney *et al.*

(1997). Figure 5.1 illustrates the evaluation framework in a very abstract way. All the framework needs as an input is the standard image dataset to test, the segmentation algorithm, the evaluation method and the the parameter range to test. Our work extended the original FATE framework by adding the needed parameter searching module, the GA search optimisation module, applying it to a cluster throughput engine, and adding an additional polishing stage to improve the processing performance of the evaluation. All of these extensions will be explored in more detail in the following sections of this chapter.

The harness is written entirely in Tcl, a high-level scripting language (Ousterhout, 1998)(Welch & Hobbs, 2003), which means it runs essentially unchanged on all flavours of Unix (including Linux and MacOS X) and on Windows operating systems. This section outlines how the harness is typically used, and then explains how the program works. The harness has the potential for entirely decentralised testing: tests and datasets are downloaded over the Internet as required and executed locally and securely.

### 5.3.1 Operating modes

There are two major ‘modes’ of operation, the first of which runs the tests against the program under test while the second analyses the results. This separation is deliberate as it allows different analyses to be performed without necessitating time-consuming re-execution of the tests; and comparison is greatly facilitated, as there is a test-by-test record of the results.

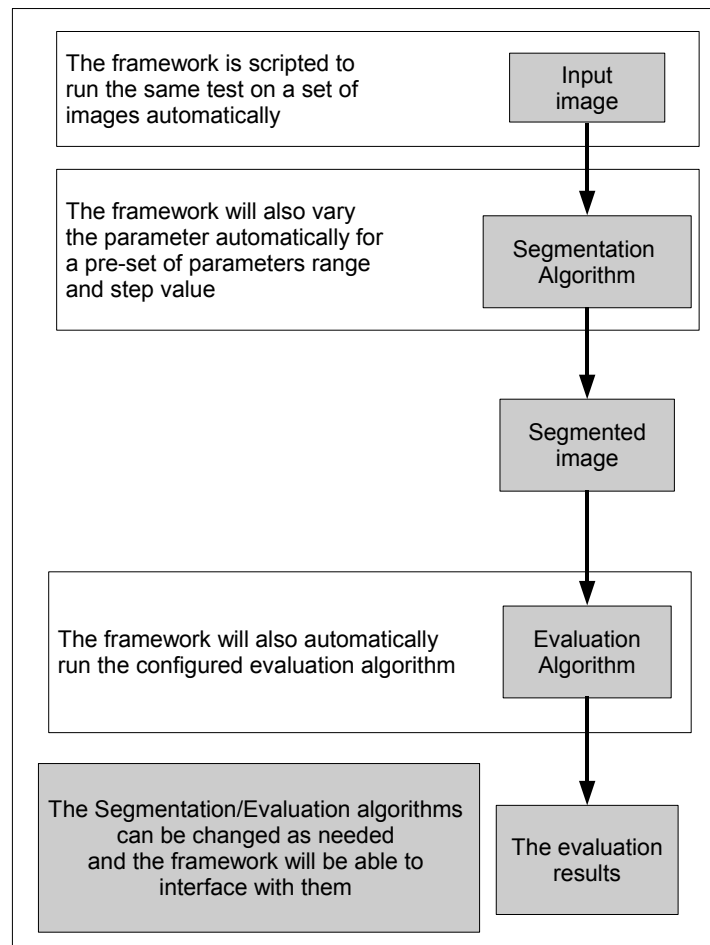


Figure 5.1: An abstract diagram of the FATE evaluation framework

### 5.3.1.1 Running tests

In ‘run mode,’ one invokes the harness by specifying a test script on the command line, capturing its output into a file, as in:

```
fate -run hdig.harn >hdig.out
```

This may be a local file or the URL of a remote test script (only HTTP is supported). If the test script is not specified precisely, the harness will try to find it in a series of pre-defined locations; this includes a directory on the local machine and the script repository on the harness web-site. Files downloaded over the network are cached locally; by default, scripts are retained for a week and data files for a year.

The harness outputs a “transcript” which concisely summarises each test and the output of the program under test. The transcript also includes the name and version of the test script, the cost function (discussed below), and the elapsed time for performing the tests; the latter is not used by the harness but is often of interest to algorithm developers. The subsequent analysis stage works solely with transcript files.

### 5.3.1.2 Evaluating segmentation algorithms

Given a transcript file, one assesses performance simply by running the harness in its default ‘evaluate’ mode:

```
fate hdig.out
```

When used this way, the harness reports evaluation statistics, which are dependent on the cost function.

An alternative is to invoke the harness with more than one transcript:

```
fate hdig.out otheralg.out http://loc/hdig.best
```

In this case, the harness compares each algorithm with each other and reports the probability that the algorithms achieve statistically significant differing performances. The receiver operating characteristic (ROC) compare True Positive and False Positive counts (Cohen, 1995) (Precision-Recall curves (Martin *et al.*, 2004) are a ROC variant.) As ROC curves give no assessment as to the statistical significance of any reported differences, another method such as McNemar's test, which is a modified  $\chi^2$  test, may be preferred. Again there is a logistical implication, as typically more than thirty tests are needed before the results approach a normal distribution. In other words, in the case of evaluating image segmentation, if the aim is to evaluate different parameter sets then the test will need to be applied with the same segmentation algorithm and the same image with thirty different parameter sets for example.

### 5.3.2 Working with the harness

#### 5.3.2.1 Interfacing to the algorithm under test

The algorithm developer writes a short piece of Tcl code, called the *interface script*, which interfaces the segmentation algorithm under test to the harness. Generally speaking, this code will invoke an external program that implements the algorithm with specific input files and/or parameters and perform any 'massaging' of the program's outputs to get them into the required form. The interface script is loaded by the harness when it is executed, and it is invoked once for each test performed.

### 5.3.2.2 Specifying the tests.

The test script is also written in Tcl as it may be desirable to perform computation in generating either the actual tests or in specifying the locations of any data files it may use. Downloading and executing code is, of course, a dangerous thing to do with the current Internet, so the test script is not executed by the harness itself but rather passed to a slave Tcl interpreter in which all operations that might affect the local system have been removed; this is similar to the ‘sandbox’ environment used by Java interpreters in web browsers. The code executing in the slave interpreter is able to perform tests by a callback to a single routine in the main interpreter.

The test script should define two Tcl procedures, named as follows:

**fate\_TestInfo:** This is invoked with one of a set of keywords and is expected to return the corresponding value. The harness uses this to obtain ancillary information about the tests, such as the number of tests and the cost function to be used.

**fate\_RunTests:** This performs the actual tests and should invoke the procedure **fate\_Test** for each test.

The user must also specify a cost function (or equivalently evaluation method) that forms the basis of the comparison. Though some cost functions are built in, there is also a mechanism for providing alternative cost functions.



---

## **5.4 Reducing evaluation time**

---

### **5.4.1 Using a cluster computer**

A cluster computer, the second component of the evaluation environment represents an accessible resource upon which algorithmic testing takes place in batch mode. Testing is a laborious operation if only a desktop computer is available. A distributed version of the harness is able to reduce the turnaround time when testing an algorithm. The distributed version runs on a cluster, using static load-balancing and process spawning through the `rsh` utility. The harness acts as a ‘throughput engine’ delivering jobs on demand, on a per job basis. A job is defined as a set of one or more tests or applications of an algorithm to one image.

The cluster computer employed by us consists of thirty-seven processing nodes connected with two Ethernet switches. Each node is a small form factor Shuttle box (Model XPC SN41G2) with an AMD Athlon XP 2800+ Barton core (CPU frequency 2.1 GHz), with 512 KB level 2 cache and dual channel 1 GB DDR333 RAM. The nodes are connected via two 24 port unmanaged Gigabit (Gb) Ethernet switches manufactured by D-Link (model DGS-1024T). Each switch is non-blocking and allows full-duplex Gb bandwidth between any pair of ports simultaneously.

The cluster computer software environment is as important as the physical hardware. To make maintenance and cluster-wide propagation of configuration changes easier, at boot time all nodes transfer their root file system from a file server to the local disk, while other file systems are accessed via the Network Filing System. The development branch of Debian sid (unstable but more current) is

used to manage upgrades to the Linux operating system. To (re)build nodes `tftp` and `udpcast` (both Linux utilities) are employed in a manner akin to a simplified `SystemImager` (software that automates Linux installs, software distribution, and production deployment). A customized script is available to synchronize packages and configuration with the cluster's master template whenever it is inconvenient to reboot and rebuild. This operation is simply accomplished through an Advanced Packaging Tool cache that is shared between nodes and a shared `debconf` `db` (a configuration database), and runs in parallel across the cluster by means of `dsh` (an implementation of a wrapper for executing multiple remote shells, such as the `rsh` or `remsh` or `ssh` commands).

Nodes rarely need to be added or removed, though there is a script to do that too. The cluster is monitored with the scalable, distributed system `Ganglia` (refer to <http://ganglia.info>).

## 5.5 Evaluation methods

---

Evaluation methods for segmentation can be divided into those that are independent of ground truth and those that are not. An early survey of independent segmentation evaluation methods was completed by [Zhang \(1996\)](#). Apart from a categorisation into five types of evaluation, [Zhang \(1996\)](#) also investigated the evaluation methods were compared by testing them on image segmentation results using threshold values. An interesting feature of this comparison was that, while no reference was made to ground truth, the methods did agree on which of the alternative segmentations was preferable. An entropy-based method ([Zhang et al., 2003](#)) was introduced after Zhang's survey; its authors advocate its use in

## 5.6 The Berkeley segmentation dataset and benchmark

---

preference to the other methods from the quantitative school, because the latter tend to be based on ad-hoc empirical evaluations with no theoretical basis.

The global and local consistency measures introduced in (Martin *et al.*, 2001) produce high scores for segmentations by humans and, hence, will rate a computer-generated segmentation highly if it corresponds to a human segmentation. Because these measures are tolerant to refinements, over and under segmentation is tolerated in the sense that the measures do not vary greatly in judging these, if the same algorithm has been used with the different parameters. Over segmentations may not be a problem if subsequent processing is able to aggregate regions but if segmentation is regarded as the end product it is an issue. Another measure based on Hamming distance (Huang & Dom, 1995) may be used as a corrective.

Both (Martin *et al.*, 2001) and (Huang & Dom, 1995) are intended for the quantitative approach considered in this chapter. However, the main purpose herein is not to compare evaluations, though in the next Section the evaluation methods appear as cost functions which are inserted into an evaluation harness.

## 5.6 The Berkeley segmentation dataset and benchmark

---

The Berkeley dataset (Martin *et al.*, 2001) is a scientific effort to bring together 30 human subjects to segment a set of 1,000 natural images which formed 12,000 segmentations. Greyscale images were used to obtain half of the segmentations; the other half were obtained from presenting the subjects with colour images. This approach was followed to provide ‘ground truth’ for learning what is involved in

---

## 5.6 The Berkeley segmentation dataset and benchmark

---

the segmentation activity and for benchmarking segmentation algorithms. The Berkeley group aim to use the data collected to exploit the “relative” perfection the human visual system has reached in segmenting objects. The human visual system, they theorise, is so good at segmenting because it uses “natural statistics” to its advantage. Additionally, the research group think that computational segmentation algorithms should be evaluated quantitatively. Thus, their experiment will be able to scientifically evaluate image segmentation for both cases: a) the human visual system, and b) the computer vision.

To achieve this goal the Berkeley research group developed segmentation comparison measures that are used to validate the consistency of the human subjects’ data and to provide approaches for evaluating segmentation algorithms. The ultimate aim is to use these performance measures to systematically improve segmentation algorithms by considering the human ground truth as the “ground truth”.

### 5.6.1 Berkeley group’s segment and segmentation definitions

The Berkeley group uses a “high-level” definition of segmentation rather than the “low-level” definition which is widely used in the computer vision literature.

The research group aimed from the start to achieve segmentation algorithms that will decompose images in a manner as similar to human beings as possible, and thus they aimed to study how human beings decompose images they receive using their visual systems. So the question that needs to be answered is: What does it mean to “decompose” an image from the human perspective.

## 5.6 The Berkeley segmentation dataset and benchmark

---

The term segmentation is mostly used in computer vision literature to refer to a low-level process of creating a group of pixels from pixels that are spatially coherent. Pixels are a good representation for computational processing but are arguably an inconvenient representation for visual coherence and understanding of the natural objects in the images.

Natural world images are in general made up of physically disjoint objects whose concurrence in a scene leads to an image consisting of spatially coherent groups of pixels. So pixel groups are segments, and the process of dividing an image into segments is segmentation. However, the low-level definition of segmentation is imprecise because it depends on the current state-of-the-art in defining uniformity and coherence. Pixel coherence based on photometric features is limited and subjective.

The Berkeley group's stable definition of segmentation is also the statement of the ultimate goal: finding regions of semantic coherence. In other words, the regions are related to objects as defined by the human logical reasoning, for example, lion or a tree rather than regions that only have spatial similarity. To ensure the general applicability of a segmentation dataset, the group therefore used the high-level definition of segmentation based on objects as seen by generic humans rather than on features as seen by vision scientists.

This high-level definition also helped define their segmentation representation as a region-based representation rather than a boundary-based representation. A region-based representation is concerned with representing objects in the segmented image as closed boundaries. Whereas a boundary-based representation doesn't require closed boundaries to represent objects. Objects that are not closed may arise from two sources: individual objects that have a folded topol-

## 5.7 Testing the human hand-segmentation precision

---

ogy, and objects blended into each other because of shading. Neither situation is relevant to the Berkeley group's standard as they are concerned with low-level representations from the Berkeley group's definitions perspective.

With their hand-segmented dataset, the Berkeley group aims for a "gold-standard" image segmentation that contains high-level information from a human visual system. Ultimately, the aim is to incorporate this high-level information into the computer vision algorithms. To use their own words, the Berkeley group defines their region-based representation as follows: "segmentation is a partition of the pixels of an image into disjoint sets. The sets need not be contiguous in the image plane."

## 5.7 Testing the human hand-segmentation precision

---

The Berkeley data-set's group, after considering different options, used a custom segmentation tool that allows the user to highlight the segments by dividing the image pixels. The images are shown on the computer monitor for the user and the basic skill of using the computer mouse is expected. Some additional instructions are given to help them operate the custom application and choose the segments correctly. Now the Berkeley group, at the time they did their research, identified Wacom LCD tablet as the ideal for the task of hand segmentation. More basic options like simple pen and paper, Adobe Photoshop (or similar) application, and also a non-LCD pen and tablet connected a computer all have their shortcomings. Using a pen to trace segments on semi-transparent to transparent paper overlaid over the image, and then scan the segments digitally would have added additional steps where errors could be introduced in each step. Adobe Photoshop provides

## 5.7 Testing the human hand-segmentation precision

---

some tools to help in this task. However, it does not provide everything because the input will still need to be taken by an external device: a mouse and keyboard, non-LCD tablet, and LCD tablet, all have their shortcomings. However, the LCD tablet is the least susceptible to the hand-eye co-ordination problem and the best solution is to provide a familiar user interface for the human performing hand-segmentation and at the same time allow direct digital input.

The group provided a good description of the custom segmentation tool. However, the tool is not available for download (like the data-set) and the description does not answer all the questions. For example, they indicate that there is no difference in principle between using a mouse for segmentation and a non-LCD tablet from the test they carried out. However, they then mention that a pen-tablet seems more natural and user-intuitive for the task. They mention hand-tremors as a disadvantage for the pen-tablet solution. However, this can be also taken as a similar disadvantage for the mouse use. The mouse user might have additional hand-eye co-ordination errors compared to the pen-tablet solution unless the software tool actually provides some ‘help’ or error-correction. Where they clearly state that no additional algorithmic aid was provided by the tool beyond the user input, we were not able to test this as the software tool was not available.

We were fortunate in having an LCD-tablet solution in the form of a Toshiba M200 Tablet PC notebook, as this notebook had a built in Wacom tablet. It was not considered practical or scientifically useful to redo the whole process of hand-segmentation that was carried by the Berkeley group again just for the case of using an LCD-tablet as an input solution. Our aim was to illustrate that even with the most ideal solution for acquiring hand-segmentation, there is a possibility for errors to be introduced in the hand-segmentation for many reasons, and as such

## 5.7 Testing the human hand-segmentation precision

---

it's not useful to expect the algorithms' segmentation to provide 100% identical segmentation to the hand-segmentation. Figure 5.2 illustrate the M200 notebook displaying an image of a hand-segmentation example in full screen.



Figure 5.2: Toshiba M200 displaying image number 10 for hand segmentation

To achieve this we created a sample of 12 images that can be considered as basic objects and features and one of the most basic requirement was that each image can act as its ground truth for segmentation. In other words, each image is actually its own segmentation result. All the images are black and white images with 1024 by 768 pixel resolution. The Notebook was configured to the same resolution and the images were shown in full screen mode for each test to have the native resolution displayed for the user. The images with their numbers are shown in Figure 5.3.

The numbers will be used later to illustrate the different results acquired for each image. We aimed at providing different basic features in the images and



## 5.7 Testing the human hand-segmentation precision

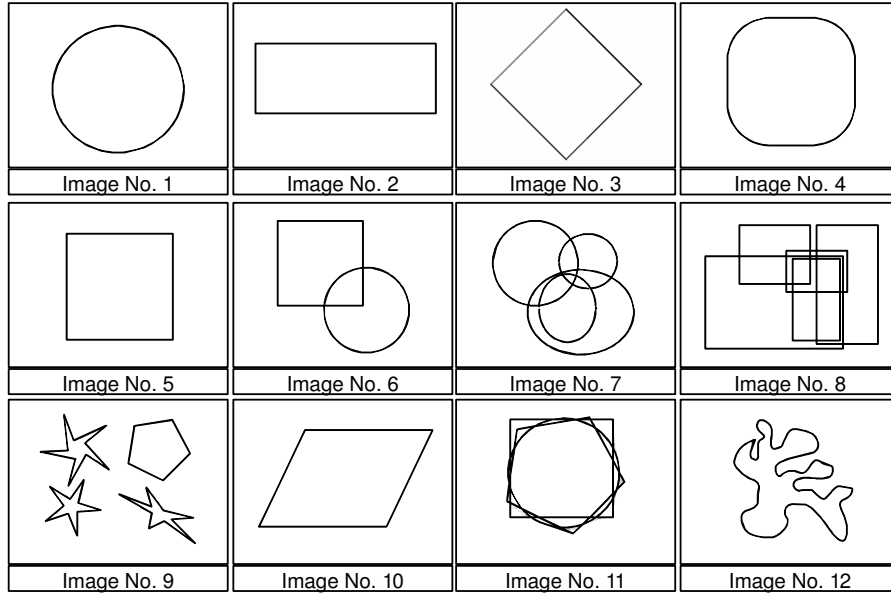


Figure 5.3: Test Images 1 to 12 used for hand-segmentation test

to give different segmentation ‘difficulties’ in different images to test the tracing ability of the human hand segmentation as much as possible. However, it’s good to point out here that natural colour images, like the one provided by the Coral database and used by the Berkeley group, will most likely have more ‘difficult’ features to trace than those provided here. Where there is only one option to trace in our sample images, the natural images will provide multiple options to trace segment lines depending on each human perception of what constitutes a segment.

So the hand-segmentation results are usually susceptible to two types of errors: errors from the human perception of the segments, which actually differ from a human to human, and errors in acquiring the hand-segmentation in digital format for use as a ground truth for algorithm segmentation results. We aimed with our

## 5.7 Testing the human hand-segmentation precision

---

sample to minimise the effect of both on our hand-segmentation test to minimise their errors.

For our test we used Inkscape<sup>1</sup> software as the capturing tool as we didn't find a reason to create our own customised software. Inkscape, or any image editing software that provides layers for editing would have done the same job. Inkscape is an open-source and free software to download, and provides the facility to display the image under editing in full-screen without any other interface of the software displayed (refer back to Figure 5.2 which shows an illustration of the program ready to be used for tracing). The user segmentation is saved in a different layer than the image to be segmented and because each user is required to segment each image five times (to average any errors) and each trial can be saved easily in a separate layer and exported at a later stage for evaluation.

For the testing we had 15 subjects, each segmented the sample of 12 images five times. After that the segmentation results are evaluated with the original image to find the mean difference between the segmentation and the original image. Figure 5.4 show the highest and lowest mean differences recorded for each subject.

From the figure it's clear that there was no perfect segmentation from any subject. And although the minimum differences recorded are low for some segmentation results, it's still significant enough to record a difference in the evaluation process.

These errors can be identified as the tremors mentioned above or even as the whole digitising process of the pen-tablet solution. The important point here is we can't expect the hand-segmentation to give us an exact segmentation results

---

<sup>1</sup>Inkscape (open-source vector graphics editor): <http://www.inkscape.org/>

## 5.7 Testing the human hand-segmentation precision

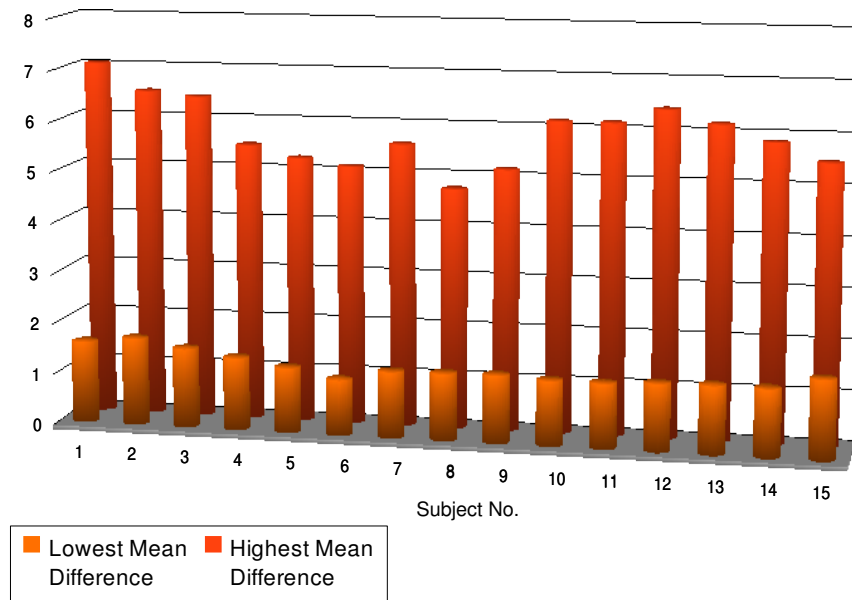


Figure 5.4: The highest and the lowest mean differences calculated for each subject

## 5.7 Testing the human hand-segmentation precision

---

even with the best results and set-up available.

The highest mean differences recorded are different from one subject to another and those high errors corresponds to test images that have ‘difficult’ features to trace. Some subjects perform better with these images than other subjects.

In general, there is no significant difference between the subjects. They become comfortable with using the segmentation solution after few trials. However, some of them are more proficient or keen to spend as much time as necessary to complete the whole process.

Although great care was taken to instruct the subjects to be careful with their segmentation and they were not given any time limits to complete the task, some subjects became irritated and tried to complete the whole set as fast as they could, which could affect the quality of their segmentation. However, overall the results show no difference between the subjects results, especially with the combination of the five trials results which helps in minimising any errors.

We are not aware if the Berkeley segmentation subjects were given a number of trials to complete or get used to the system. However, they state that 1-2 hours was given to each subject to get used to their customised segmentation tool or were taken by each subject to complete the task.

Looking at the results arranged by each image, however, gives us a better understanding on how the image decomposition and the skill to identify and trace the segments in the image can introduce errors into the segmentation. The highest and lowest mean difference recorded for each image is illustrated in Figure 5.5.

It’s clear here that the errors increase as we get to image six, which has overlapping objects in the image to segment. The highest errors are acquired with

## 5.7 Testing the human hand-segmentation precision

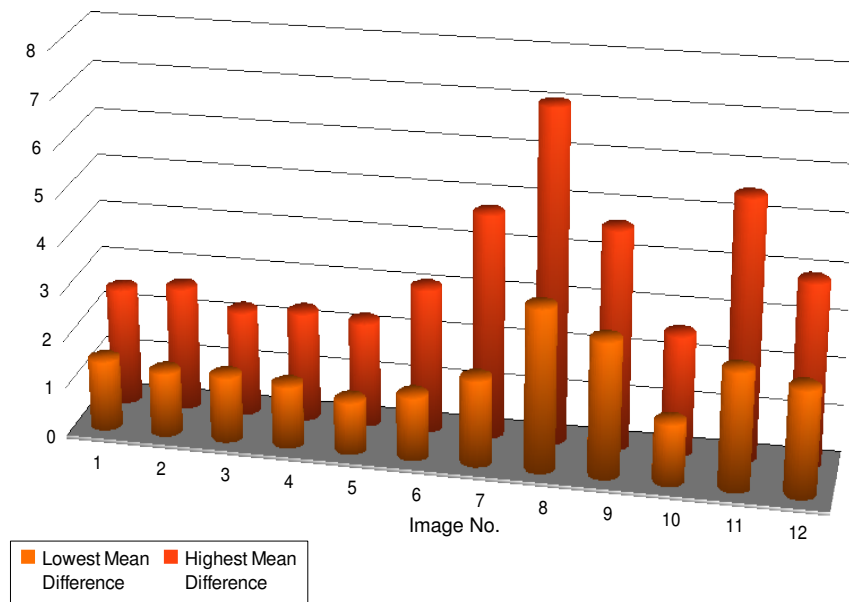


Figure 5.5: The highest and the lowest mean differences calculated for each image

image 8 which has many overlapping objects with adjacent boundaries. However, the overlapping segments and adjacent boundaries are not the only features in the images that can increase the errors. Segment boundaries with hard and irregular corners can also be affected by the high errors in hand-segmentation. This is to be expected, as those boundaries are harder to trace and errors will be compounded from both the tremor errors and the pen-tablet digitisation process.

All of these mentioned features: overlapping segments, adjacent segment boundaries, hard segment corners, and irregular shape boundaries, are typical traits of objects in natural colour images like the ones in the Coral image database. So it is clear that the hand-segmentations obtained are affected by the errors associated with this method. It's important to point out here that the images in the test performed here are 'easy' to segment compared to the difficulty faced by

## 5.7 Testing the human hand-segmentation precision

---

hand segmenting natural colour images. The difficulty can be two-fold: firstly while identifying the objects and the related segments to each object, and secondly tracing those segments. And such errors added to the hand-segmentation are certainly higher than what was found in our results.

It's also worth mentioning here that although the Berkeley segmentation dataset had employed 30 subjects for the hand-segmentation task, not all the subjects have segmented all the segments under the dataset. Specifically one subject completed only one segmentation and another completed segmentations for only two images. And 10 subjects of the 30 completed segmentations on about 10% or less of the whole image set (104 of the 1020 images).

Only two subjects completed the whole set. See Table 5.2 for the details. By definition the Berkeley group, for some reason, made the restriction that each image is not segmented more than 6 times. So looking through the images available from the downloaded dataset, on average images are segmented by 5 or 6 subjects.

In spite of this, the Berkeley group segmentation dataset is a groundbreaking effort that unfortunately was not used as much or built upon by other image segmentation research as the authors would like it to be. If an effort like this can be adopted and standardised as the testing dataset of choice then a good measure for image segmentation improvement can be defined over time.

For our work the Berkeley dataset is the only one that provided a significantly large dataset, easily accessible and with a good variety of colour images for testing.

## 5.7 Testing the human hand-segmentation precision

---

Subject No.	ID	Images segmented
1	1131	1
2	1125	2
3	1111	13
4	1118	14
5	1110	46
6	1128	50
7	1104	54
8	1102	82
9	1106	84
10	1129	104
11	1126	166
12	1117	207
13	1127	248
14	1122	249
15	1119	290
16	1113	308
17	1114	313
18	1112	344
19	1116	358
20	1132	381
21	1121	432
22	1103	509
23	1130	678
24	1124	771
25	1107	920
26	1115	953
27	1105	978
28	1108	1000
29	1109	1020
30	1123	1020
Totals		11595

Table 5.2: Table of Berkeley dataset Human subjects showing all the subjects their IDs and number of images segmented by each subject.

---

## 5.8 Concluding Remarks

---

This chapter sets the stage for the coming chapter by introducing the evaluation framework that will be used in the coming chapter to test the segmentation results. This will occur after introducing the research field of the available programming libraries and explaining the need for an evaluation framework for the computer vision in general and the image segmentation in particular.

Firstly the research aims to cover three basic points needed by any empirical evaluation: 1) a standard database, 2) an evaluation protocol, and 3) scoring methods. The options chosen are detailed and specifically the evaluation protocol is given which depends on the FATE evaluation framework.

The aim is to achieve the benefits of providing a solid scientific and experimental basis for image segmentation to help researchers develop engineering solutions to practical problems. Furthermore, this will help to obtain an accurate assessment of the state-of-the-art research, and a convincing evidence for the potential users to help them distinguish a practical solution to their problems from the evaluated image segmentation research.

After exploring the operating details of the evaluation framework, Section 5.7 extended the tests performed on the Berkeley dataset by testing if there is any advantages to the hand segmentation performed on an LCD tablet. The Berkeley database testing procedure didnt have access to a LCD tablet to complete the hand-segmentation. However the authors of the test argued that the LCD tablet can be considered the best method to obtaining the human hand-segmentation images, and this author thinks that this still has not changed at the present time. The outcome shows that even with the LCD tablet errors can be introduced



## 5.8 Concluding Remarks

---

during the hand-segmentation process, and it can be arguably higher with the method used in the Berkeley dataset. However, the Berkeley dataset remains as a very useful standard dataset for the segmentation evaluation. The author only points out that, while using the Berkeley dataset, the aim should not be only to fit the segmentation methods results to the specific hand-segmentation results perfectly, but to use it as an evaluation benchmark taken as a whole set. Furthermore, other objective evaluation methods can be used as additional benchmark that do not need any reference image to complete the evaluation. This will be explored in more detail in the coming chapter.

# 6

## Genetic algorithm optimisation for the evaluation framework

The whole is equal to the sum of its parts.

---

Euclid in The Element: Book II

The whole is greater than the sum of its parts.

---

Max Wertheimer in Gestalt theory

### 6.1 Using genetic algorithms

---

Segmentation algorithms, as mentioned in the previous chapters, are actually made of a series of sub-processing stages that require different sets of parameters. Segmentation algorithms' designers either don't expose the underlying processing components, or expose some of the parameters for the processing stages while suggesting "default" parameters values that, in their opinion, should provide

the best results in the images under test compared to other algorithms' designs. As such, the common practice was that parameters will be adjusted by hand to arrive at the best combination for the application under study according to the pre-defined ground truth. This practice is neither effective nor efficient, and mostly close to impossible to achieve.

It is important to note that the literature in this field is filled with new segmentation algorithms that introduce different approaches and try to fulfil different needs, as was summarised in Chapter 2. However, testing the quality/performance effectiveness of a suggested implementation is often only performed on a few images (Everingham *et al.*, 2001). Although as suggested by Everingham *et al.* in (Everingham *et al.*, 2002a,b), looking at the parameters is an important point in segmentation evaluation, suggested focusing on the parameters while evaluating the segmentation algorithms. However, they do not discuss how those parameters affect the final segmentation results and what each parameter corresponds to in the processing stages in the segmentation algorithms. Looking at this will provide us with the important stages that affect the final segmentation quality and overall performance and we can already see that most of the segmentation algorithms explored in the previous chapters used similar processing stages, even across different categories of the classification taxonomy introduced in Chapter 3.

Parameters have been identified by the author as an effective factor in determining the final segmentation output and have been used in two ways. The first way is automated segmentation algorithms that will automatically determine the parameters settings “on-line” while processing inputs (which was suggested early in segmentation research by Bhanu *et al.* (1989) and continues to be introduced

and used in recent years (Chabrier *et al.*, 2008; Melkemi *et al.*, 2006; Pignalberi *et al.*, 2003) in different types of segmentation applications). For example Pignalberi *et al.* used a genetic algorithm in the segmentation of range images. The other approach uses evaluation methods that test “off-line” a set of images for a certain application against different parameters settings to determine the optimal parameter set for the application under study. Both approaches use some type of machine learning technique to accomplish this task (Chabrier *et al.*, 2005b; Zhang *et al.*, 2005, 2006) and are also similar to the Everingham *et al.* approach mentioned earlier (Everingham *et al.*, 2001, 2002a,b).

GAs (and other machine learning techniques) have been used extensively in computer vision and image processing (Chojnacki *et al.*, 2004; Hall, 2006; Olague, 2007) in general and in particular image segmentation (Chabrier *et al.*, 2005a). The distinction between machine learning and GA is not always large. However, the author’s research takes into consideration that machine learning can mostly be considered as an optimisation problem (Burjorjee, 2007). Bennett & Parrado-Hernández (2006) remarks:

“Optimization lies at the heart of machine learning. Most machine learning problems reduce to optimization problems.”

Additionally, there have been many studies on formulating image segmentation as an optimisation problem (S. Levachkine, 2000; Schoenemann & Cremers, 2007a,b). Intuitively, the case of the parameter-fitting for image segmentation also lends itself inherently to being an optimisation problem. GAs by definition are designed to achieve good results in these type of problems that have wide solution spaces.

## 6.1 Using genetic algorithms

---

All of the research work in parameter fitting in segmentation mentioned above only tries to improve the parameters without trying to understand what each parameter corresponds to, and usually the parameters under test are not mentioned; in general the research introduces a suggested evaluation framework that is able to evaluate parameters. However the parameters are not identified and no further parameter significance or parameter-specific test performance is discussed. Furthermore, usually few tests are carried on few chosen segmentation algorithms. The author's approach in segmentation evaluation also uses a GA to optimise the parameter-fitting solutions as an alternative to performing exhaustive searches on all combination of the parameters. The aim is to optimise significantly the computation speed and achieve better performance results (Al-Muhairi *et al.*, 2007a,b). However, unlike other research which uses parameters, we try to reveal the significance of the parameters under test and link them to the underlying sub-processing stages in the complete segmentation process.

This chapter will show the improvement gained by using GA in optimising the performance and in reaching a final solution. It will introduce the concepts used to improve and speed up the segmentation evaluation process. This will be achieved by firstly using a GA module to reduce the search time significantly, then introducing a polishing stage to improve the GA results and reducing the number of generations needed to reach the GA optimal solution. Furthermore, the work introduces the concept of time factor and using a time-weighted cost function for the GA to optimise the segmentation results on the basis of time in addition to the segmentation quality.

---

## 6.2 Segmentation Error Measure

---

A segmentation error/difference measure is needed for the purpose of evaluating image segmentation against a reference image (i.e. unsupervised evaluation). Although the reference images can be subjective, such as the hand-segmented images from the Berkeley Segmentation Dataset that we aim to use, the segmentation error measure allows the evaluation itself to be completed in an objective manner. The problem in calculating the difference between two different segmentation is that there is no one definitive segmentation solution. Furthermore, a problem arises in finding a way to penalise the small differences between the different segmentation results.

Moreover, the error measure needs to be independent of the segments' boundary pixelisations as much as possible, resilient to any noise along those boundaries, and deal with segmentation results with different number of regions. All of the former points are expected occurrences that can be found while experimenting with different segmentation algorithms or with different parameter sets with the same algorithm. The Berkeley group suggested a measure that we think sufficiently satisfies the points mentioned earlier as was demonstrated in [Martin \*et al.\* \(2001\)](#). The measure will be defined below.

### 6.2.1 Error Measure Definitions and Equations

The image in digital form is a set of pixels. Hence, a segmentation can be defined as grouping the pixels into a set of regions. Let  $S$  be a segmentation result for a single image. The error measure takes two distinct segmentation results:  $S_1$  and  $S_2$  for the same image as input and outputs a real value output in the

## 6.2 Segmentation Error Measure

---

range of [0..1] where value zero means that there is no difference between the two segmentation results; in other words, they are identical.

For each pixel  $p_i$ , the error measure can be defined as follow:

$$E(S_1, S_2, p_i) = \frac{|R(S_1, p_i) \setminus R(S_2, p_i)|}{|R(S_1, p_i)|} \quad (6.1)$$

where  $R(S_j, p_i)$  is the region in segmentation  $j$  that contains pixel  $p_i$ ,  $\setminus$  denotes the set difference, and  $|x|$  denotes the cardinality of  $x$ . This error measure defines the difference in one direction.  $E(S_1, S_2, p_i)$  is equal to zero (i.e. the two segmentation are identical) if all the pixels in  $S_1$  are also in  $S_2$  but not *vice versa*. So it has to be computed twice to achieve the complete error measure. This can only be achieved for the whole image by using two additional measures. Let  $n$  be the number of pixels in the image:

$$GCE(S_1, S_2) = \frac{1}{n} \min \left( \sum_i E(S_1, S_2, p_i), \sum_i E(S_2, S_1, p_i) \right) \quad (6.2)$$

$$LCE(S_1, S_2) = \frac{1}{n} \sum_i \min (E(S_1, S_2, p_i), E(S_2, S_1, p_i)) \quad (6.3)$$

The Global Consistency Error (GCE) is formulated to take any local changes in the same direction. On the other hand, the Local Consistency Error (LCE) allows for changes occurring in segmentation results at different locations. In other words, GCE is a tougher measure than LCE. [Martin \*et al.\*](#) illustrates that

---

### 6.3 Exhaustive parameter search testing

when two distinct human hand-segmentations for the same image are compared, both the GCE and the LCE measures produce a very low value; on the other hand, when two random human hand-segmentations for different images are compared, both the GCE and the LCE measures as expected to produce a very high value. Furthermore, the measures are created not to evaluate extreme examples, where for example the segmentation have one region that encompasses all the pixels in the image or the other extreme where each pixel is a region by its own. In both of these cases, the measure will not provide a correct validation. However, when using it to validate image segmentation results of different algorithms against the human hand-segmentation then it is suitable. In our research, the LCE measure was used because it achieved a good error measure while still providing an acceptable tolerance to differences.

### 6.3 Exhaustive parameter search testing

---

This section describes briefly some of the first results from the exhaustive search segmentation evaluation that led this research to use the genetic algorithm to enhance the performance of the evaluation and lower the time needed to find the best parameter sets. The research started with running the evaluation with all the parameter sets possible and then compared and analysed the final results to find any relation that can link between any single parameter and the enhanced quality of the final segmentation results.

During early exhaustive search testing by the author it was noticed that clearly certain parameters affect the quality of the segmentation results. Interesting findings were found while testing was carried out on the segment sizes



### 6.3 Exhaustive parameter search testing

found by different combinations of parameters. For example for mean-shift segmentation, Figure 6.1 shows the results of an exhaustive search across a set of images from the Berkeley database to explore further the parameter space of the mean-shift algorithm. The harness was configured to output segmented regions according to their size in pixels. For fixed value of `radiusS=1` and `radiusR=1`, the `colorDistance` setting governs the size of segmented region. In this example, the single parameter governs a wide variety of possible segmentations.

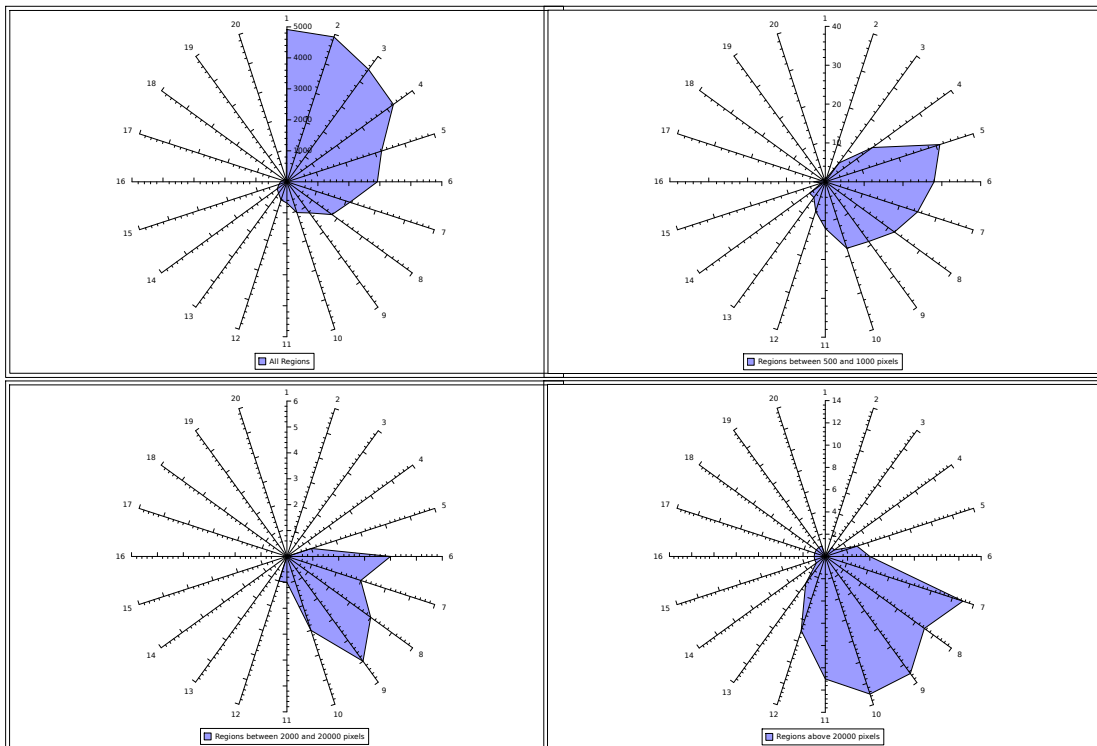


Figure 6.1: The results of an extensive search of parameter space for the mean-shift algorithm grouped by segmented region size. The `colorDistance` parameter is varied between 1 to 20 (in the circular direction), while `radiusR` and `radiusS` are fixed with value 1. The spokes of the circle represent the frequency of regions at a particular `colorDistance` parameter setting.

The results in the figure show that a certain range of parameters affect the

number of objects in the segmentation results. So regions between 500 and 1000 pixel size reach the peak values when the colour distance parameter is between 5 and 10 in value. This is also true for an overlapping range of the parameter value for regions sizes illustrated in the figure. The results here only gave an early suggestion to the parameters significance and cant be taken in isolation. As a consequence, further tests were carried out to find the parameter values significance, as will be detailed in the coming sections.

## 6.4 Using a genetic algorithm

---

### 6.4.1 Genetic Algorithms: Basics

Genetic Algorithms (GAs) form an important area of evolutionary computation, and have been applied to many problems in computer science and engineering (Goldberg, 1989; Holland, 1975). Natural selection rules discovered by Darwin are the basis for GAs; this basis is used to formulate for each problem a population of individual solutions that are evolved to find the best solution.

In implementation, a binary digit string can be used to represent each individual solution. This encoding includes all of the information needed to build an actual solution to a problem. This is termed the 'genotype' in GA terminology (mirroring biological terminology). The genotype is then used to construct an actual solution for a problem, known as the 'phenotype'. The genotype and phenotype can be seen collectively as an individual. In natural selection biology, DNA encodes the genotype, while the actual organism is the phenotype. Of course, evolution simulation in the computer is abstracted and simplified. The

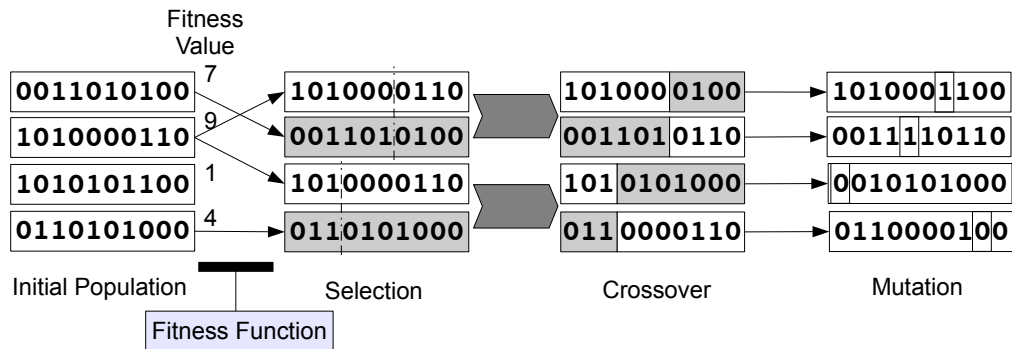


Figure 6.2: Genetic algorithm operations representation

genotypes used in GA are much smaller than those used in biological beings and, as a result, the phenotype (the solutions) which they produce are likewise simpler.

In implementation, a genotype is made up of one or more strings of binary digits (chromosomes) that encode the characteristic of the problem under study. Figure 6.2 illustrates four genotypes (that are made up of one chromosome each) and the different GA operations that are applied to them to produce a new solution for the problem they define. Each genotype represents one possible solution (although not necessarily an optimal one) for the problem under study. This first population of solutions is usually generated through a random process.

Each genotype has fitness value that determines how close it is to the best solution to the problem. One normally retains for 'breeding' the genotypes with the highest fitness value, and discards the genotypes with the lowest fitness; this is termed 'selection'. The selected genotypes are combined to produce sibling genotypes using a 'crossover' operation which typically takes parts from two selected parents to produce a new genotype.

One more location on a chromosome can be changed at random to mimic mutation, ensuring that the produced genotypes don't fall into a local minimum. After this the resultant genotypes are passed through the fitness function again and the whole process is repeated until either the desired fitness value is attained or the maximum number of iterations is reached. Further discussion of the use of GA in image processing and specifically in image segmentation will be given in Chapter 6.

### 6.4.2 The GA implementation details

A GA (Goldberg, 1989) search module was used in the evaluation environment first and foremost to decrease the processing time for the search as a whole. Furthermore, the GA is used as an additional module was used to optimise the selection of parameters. As is well-known, a GA emulates to some extent the supposed process of genetic evolutionary adaptation. For example, the GA algorithm employed allows mutation of chromosomes that represent the parameters as a mean of preventing them from becoming too close to each other and remaining in local minima. While in genetics a chromosome is a molecular package containing genes, in a GA a chromosome becomes a vector containing a set of variable elements.

The GA employed is a real-valued GA inspired by Polheim's GEATbx, a GA toolbox for Matlab (Polheim, 2005) (though the output may be integer-valued parameter settings). It employs extended intermediate recombination (Mühlenbein & Schlierkamp-Voosen, 1993), wherein an offspring gene  $g_O$  is conceived from its

two parent genes  $g_1$  and  $g_2$  by

$$g_O = \alpha g_1 + (1 - \alpha)g_2 \quad (6.4)$$

where  $\alpha$  is a scaling factor in the range  $[-d, 1 + d]$ . Assuming that the  $N$  genes in a chromosome form an  $N$ -dimensional hypercube then, when  $d = 0$ ,  $g_O$  lies along the straight line between  $g_1$  and  $g_2$ ; if  $d > 0$ , extrapolation is permitted. In the work in this Chapter,  $d = 0$ .

Given the range of parameter values, the starting point is to select random values of each parameter. It is possible to select a stopping point by checking when the difference in the cost or fitness function's value between successive generations passed below a threshold. However, because of the relatively lengthy time taken to evaluate each application of a segmentation algorithm to an image, stopping after a given number of generations is a practical alternative. The limit on the number of generations was chosen after a number of tests were carried out to determine the best value. The rate of convergence towards a global minimum value of the cost function for the mean-shift algorithm in the tests in Section 6.5 was found to be determined by the size of the population at each generation. However, the convergence over many generations was also examined.

## 6.5 GA optimisation results

---

### 6.5.1 Example of GA improvement over exhaustive search

The mean-shift algorithm (Comaniciu & Meer, 2002) makes a convenient example. The example here will illustrate the improvement in the time needed to find the

## 6.5 GA optimisation results

---

optimal set of parameters for the best segmentation quality. The approximate computational cost of an exhaustive search across parameter space for a single image on a single cluster node is as follows. The processing time on a single cluster node takes between a minimum of 907 ms (i.e. less than 1 s) and a maximum of 26799 ms (about 27 s), depending on the parameters selected for a particular image. The average time taken was 5937 ms. Assuming 6 s per test with each of three parameters varied between 1 and 20, (i.e. 8000 tests), then a complete run would take 13.3 hours. The parameter choice depended on the suggestion of the mean-shift authors, and the parameter range was chosen to give an example of the improvement in this case. However, in a later section in this chapter the range will be defined taking the mean-shift authors' suggestion (Comaniciu & Meer, 2002), and the expected range of each parameter. Evaluation of each result, the cost function after normalisation, consists of pixel-by-pixel comparison with a ground truth image from the Berkeley database. However, this calculation took less than 1% of the processing time. The cluster computer, introduced in section 5.4.1, was employed as a throughput engine, in the sense that no exploitation of internal parallelism took place. In the unlikely event that all nodes were available and disregarding input/output overhead the per image run time for a single image is still 22 min.

For the same image, a run with a GA took 130635 ms, i.e. 2.2 min, which on a cluster computer takes about 35 s. This is, of course a considerable reduction on an exhaustive search.

Table 6.1 is an illustrative rather than representative run showing the generation-by-generation best-fit selection of 'chromosomes'. Three parameters formed the chromosomes: `radiusR` (the range radius of the mean-shift sphere in colour

space), `radiusS` (the spatial radius in grid space), and `colorDistance` (defining the region merge threshold). There was a population of just five for each generation, which explains why the cost function's value does not reach a minimal value (as shown next).

The population size in the GA is how many chromosomes are in a population (in one generation). In the context of the GA's parameter search this means that population will contain five different parameter sets (i.e., chromosomes), each set will contain three random values representing the mean shift segmentation's parameters defined above. A very high value for the population size will produce too many chromosomes and as consequence will slow down the GA process. On the other hand, a very low value will generate very few chromosomes and will limit the GA process in finding a good solution.

The encoding of the problem (the parameter search, and the parameter set values in the chromosomes in this case) influence the choice of the population size (and other GA parameters), as the research shows. As a consequence there is a limit that depends on the problem-encoding in which GA solving will be significantly slower without improving the final solution search. This is an example, so a population size of 5 is just for illustration. However, as will be described in the coming sections, after extensive testing, a population size of 20 was found to be an acceptable value that gives the GA enough random parameter sets to evolve a solution from and combined with the choice of 100 generations and the other GA parameters defined gives an acceptable compromise that arrives at an optimal parameter set for the give problem in an efficient computation time. Further GA evaluation improvements will be explored in more details later.

Two GA parameters, the recombination rate, i.e.  $\alpha$  in the line recombination of

## 6.5 GA optimisation results

---

Section 6.4, and the mutation rate were set to 0.6 and 0.8 respectively. The latter governs the ability to escape from a local minimum. As with such evolutionary algorithms, it is not otherwise possible to directly govern their behaviour, which is probably their principal disadvantage.

However, the reduction in time needed at reaching a selection adequately compensates for that. For the image in this test, the average time for each call to the segmentation algorithm was 6 s, with the maximum time at 21 s and the minimum time at 0.9 s. The total time taken in running the calls was 298 s, while the total time for all processing was 299 s, i.e. about 5 min.

A disadvantage of truncating the search after a fixed number of generations is that there is no guarantee that the GA could be (say) performing hill climbing to leave a suspected local minimum. In Table 6.1, the seventh generation value is less than the eighth and, therefore, had the search been concluded at the eighth unhelpful parameter values would be selected. A possible solution to this problem is to allow the fittest individuals to ‘live’ for  $>1$  generation; another is to include a refinement or polishing stage in which the path taken by the GA over each generation is inspected to more closely approach a global minimum. This issue is returned to at the end of this Section in relation to graph-based segmentation.

Experimenting with the GA as in Table 6.2 with a population of ten genes in each generation gives rise to some interesting observations. The first point to notice is that the cost function’s value remains the same throughout the tests. The rapid descent to a global minimum arises because a population of ten was taken for each generation. As explained earlier, the higher value of the population size will provide the GA a wider set of candidate solutions to test at each generation and as a result it is expected that the GA will arrive at the optimal solution



## 6.5 GA optimisation results

---

		Chromosomes		
Gen.	Cost	radiusS	radiusR	colorDistance
1	18.63	3	8	16
2	14.17	3	14	16
3	15.27	3	16	14
4	14.61	1	16	17
5	16.74	11	4	17
6	13.76	11	18	19
7	11.87	3	17	19
8	14.26	12	17	19
9	11.19	2	15	19

Table 6.1: Example output of the evaluation of mean-shift parameters with the GA module.

faster, although the processing of each generation will take longer.

The other important point is that although `radiusR` and `radiusS` do not converge to produce an optimal set of parameters, `colorDistance` is never lower than 11. The author found that these two points are always true for the mean-shift algorithm over a range of different images, and although the lowest score reached will be different for each image, it can be concluded that `radiusR` and `radiusS` do not make a significant difference to the result as long as the `colorDistance` parameter is larger than 8.

The more common alternative to truncating the search after a fixed number of generations with a large initial population is to complete the search after a convergence criterion has been met. How the search is conducted is a pragmatic decision, as in essence a GA is simply a more disciplined way than random probing to explore a large problem space. To examine the behaviour over successive generations a very low population of just two members was deliberately chosen to give slow convergence.

The recombination rate was set to 0.6 and the mutation rate was set to 0.2

## 6.5 GA optimisation results

---

		Chromosomes		
Gen.	Cost	radiusS	radiusR	colorDistance
1	4.71	2	5	16
2	4.71	2	16	11
3	4.71	2	16	11
4	4.71	2	4	17
5	4.71	2	2	17
6	4.71	16	2	19
7	4.71	2	10	16
8	4.71	2	18	15
9	4.71	2	16	12
10	4.71	2	17	12
11	4.71	2	10	19
12	4.71	2	10	19
13	4.71	2	10	17
14	4.71	2	15	15
15	4.71	3	11	16
16	4.71	3	18	19
18	4.71	4	15	19
19	4.71	8	2	19
20	4.71	1	3	19

Table 6.2: Trial output of the evaluation of mean-shift parameters with the GA module.

for the mean-shift segmentation parameters. A best-fit linear regression line is found by a standard numerical method, without any claims for goodness-of-fit. See Fig. 6.3 for an illustration of the convergence.

The line shows that the cost function values have a slight negative trend showing continuous improvement, though with significant divergence and even lower values of the cost function at around 40 generations. In the unlikely event that a population of just two was used then linear convergence to the evident minimum at around 40 generations would be slow. However, the minimum in this example is not suitable, because of the low number of the population in each generation which don't give the GA enough choice in each generation to evolve an optimal solution (even with a relatively higher number of generations). Therefore, it's important to increase the number of generations as will be done in the coming sections. The aim, as stated earlier, is to strike a balance between the number of populations and generations while providing the GA with a good search space and allowing it to reach a solution in an acceptable time.

In Fig. 6.4, the values of the contributory parameters are superimposed. The values of the cost function are connected by a moving average of length three. In broad terms, the behaviour of the values chosen by the GA for the other parameter is oscillatory, displaying what approaches a systematic sampling of parameter space.

Figure 6.5 illustrates similar results for testing the watershed algorithm, when it's apparent that even before 40 iterations, the GA makes substantial progress in finding a good minimum for the cost function, i.e. the match to the ground truth image. However, thereafter there are oscillations in the value. Though there is a negative going trend to the results, the value after 100 generations is

## 6.5 GA optimisation results

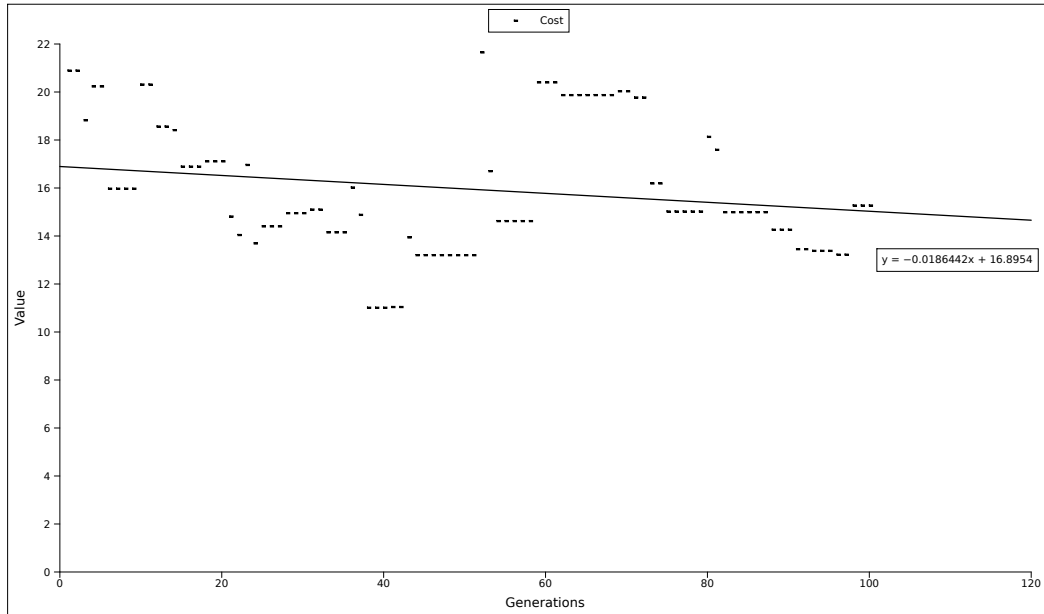


Figure 6.3: Mean-shift parameter convergence with linear trend for 100 generations of the cost function value.

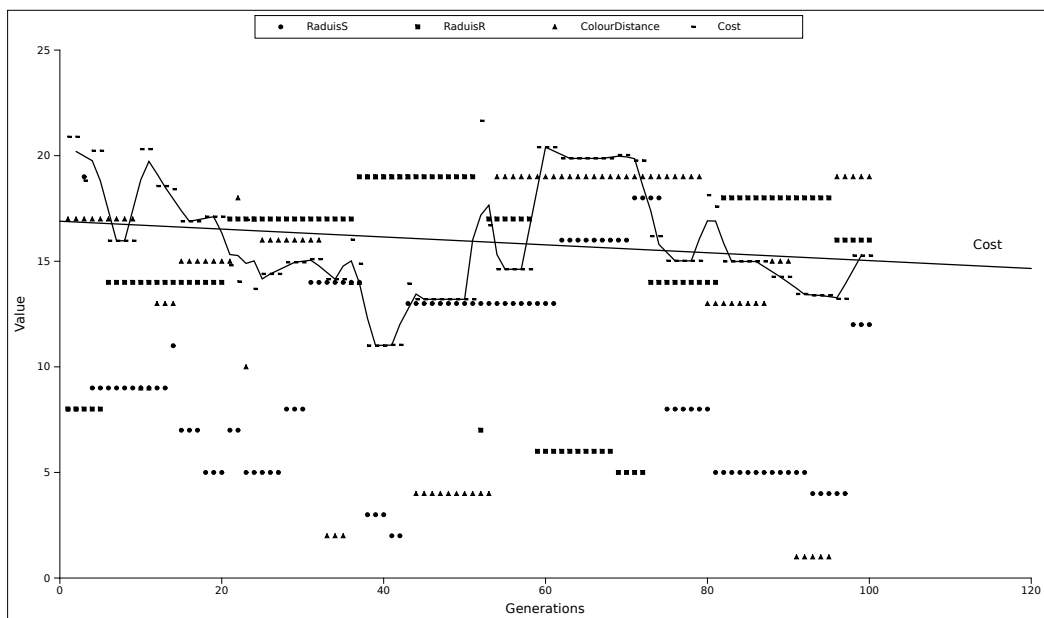


Figure 6.4: Mean-shift parameter convergence with linear trend of the cost function value, including the behaviour of contributory parameters.

## 6.5 GA optimisation results

not the minimum. Note here that there are many reasons why the cost function will not reach a zero value that are not related to the GA search directly. For example, the segmentation algorithm evaluated may not be the best choice for this test. However, the most important reason is that it is not expected that the segmentation algorithm will find the exact segmentation results as defined by the human hand-segmentation. Not only that as the segmentation algorithm can define segments that are not found by the human hand-segmenters. Even if the segments found by the human and the algorithms are exactly the same, the errors in obtaining the hand-segmentations as described earlier in Section 5.7 will provide enough differences to prevent the GA from achieving a zero cost function, even after 100 generations.

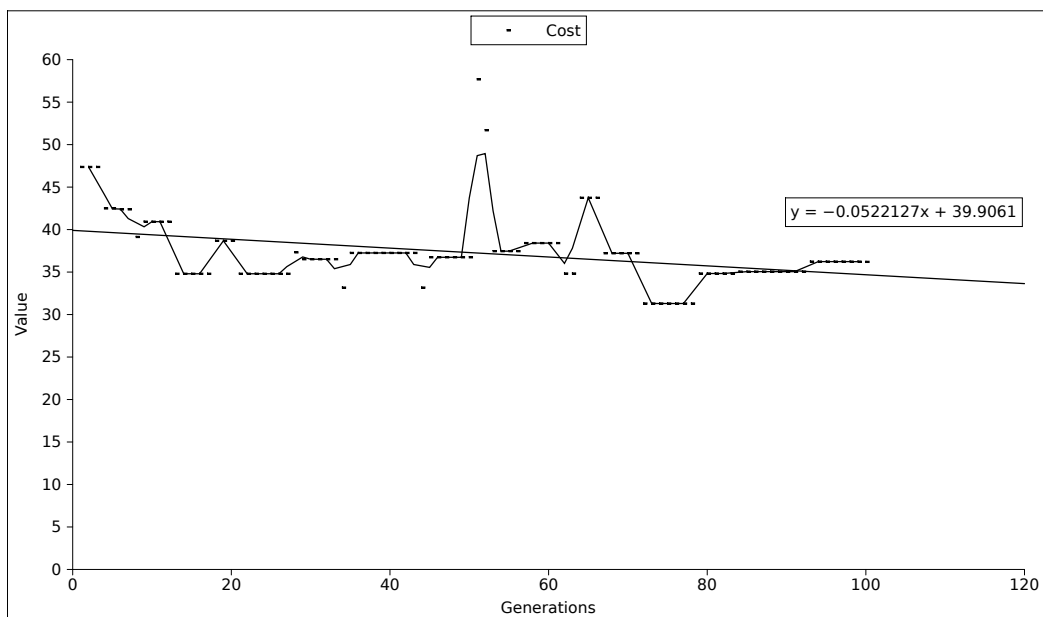


Figure 6.5: Watershed segmentation parameter convergence with linear trend of the cost function value.

Similarly to Fig. 6.4 and with the same GA search parameters, Fig. 6.6 shows

## 6.5 GA optimisation results

100 generations of a parameter search for the graph-based segmentation, again plotting a moving average through the data points. The slight negative of the linear trend is less apparent in this representation, as is the oscillatory nature of the testing of individual parameters.

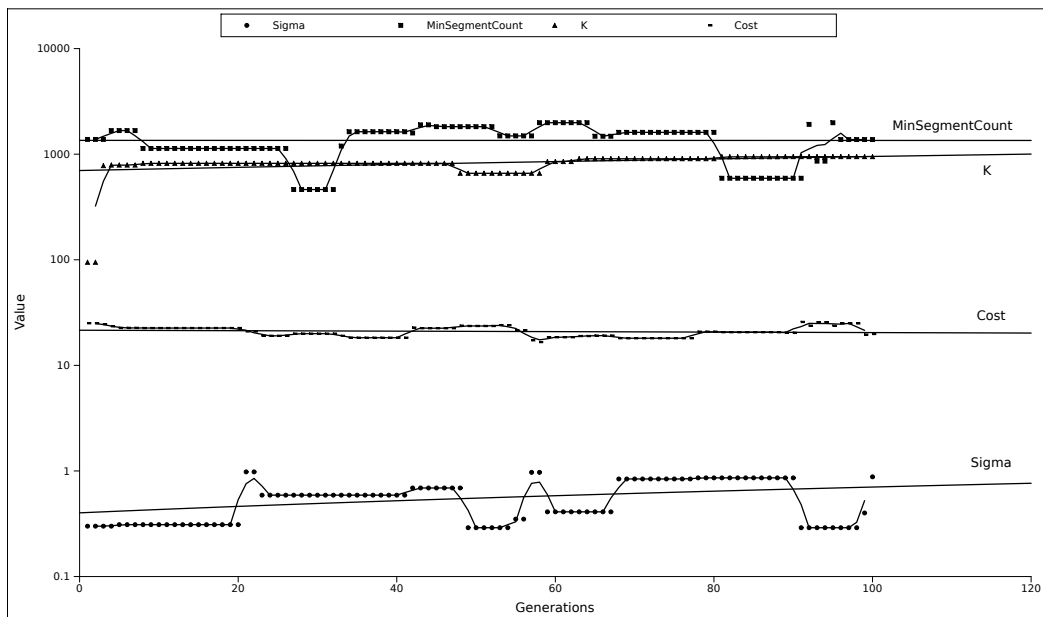


Figure 6.6: Graph-based segmentation parameter convergence with linear trend of the cost function value, including the behaviour of contributory parameters. (Note the logarithmic vertical axis.)

As mentioned earlier, it is possible to refine the output of the GA by application of a non-GA polishing algorithm to what is a non-constrained, non-linear optimisation problem. Newtonian methods are unsuitable if the cost function to be optimised is non-differentiable. Therefore, this work used the Nelder-Mead direct search, “simplex method” (Nelder & Mead, 1965).

The goal of the Nelder-Mead approach is to solve the following unconstrained optimisation problem:

$$\min f(\mathbf{x}) \tag{6.5}$$

where  $\mathbf{x} \in \mathbb{R}^n$ ,  $n$  is the number of optimisation parameters and  $f$  is the objective function where  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ . This algorithm is based on the iterative update of a *simplex*  $S$  made of  $n + 1$  points where  $S = \{\mathbf{v}_i\}_{i=1,n+1}$ . Each point in the simplex is called a *vertex* and is associated with a function value  $f_i = f(\mathbf{v}_i)$  for  $i = 1, n + 1$ . The vertices are sorted by increasing the function values so that the *best* vertex has index 1 and the *worst* vertex has index  $n + 1$ :

$$f_1 \leq f_2 \leq \dots \leq f_n \leq f_{n+1}. \tag{6.6}$$

The  $\mathbf{v}_1$  vertex (respectively the  $\mathbf{v}_{n+1}$  vertex) is called the *best* vertex (respectively *worst*), because it is associated with the lowest (respectively highest) function value. The centroid of the simplex  $\bar{\mathbf{x}}(j)$  is the centre of the vertices where the vertex  $\mathbf{x}_j$  has been excluded. This centroid is:

$$\bar{\mathbf{x}}(j) = \frac{1}{n} \sum_{i=1,n+1,i \neq j} \mathbf{v}_i. \tag{6.7}$$

The algorithm makes use of one coefficient  $\rho > 0$ , called the reflection factor. The standard value of this coefficient is  $\rho = 1$ . The algorithm attempts to replace some vertex  $\mathbf{v}_j$  by a new vertex  $\mathbf{x}(\rho, j)$  on the line from the vertex  $\mathbf{v}_j$  to the centroid  $\bar{\mathbf{x}}(j)$ . The new vertex  $\mathbf{x}(\rho, j)$  is defined by:

$$\mathbf{x}(\rho, j) = (1 + \rho)\bar{\mathbf{x}}(j) - \rho\mathbf{v}_j. \quad (6.8)$$

The main advantage of using the Nelder-Mead extension is that it can dynamically update the solution shape for the simplex. Thus, it can provide a reasonably fast convergence rate, especially when the cost function is made of a sharp valley. In general, the user should not expect a high accuracy from the algorithm. Nevertheless, in most cases, the Nelder-Mead algorithm provides a substantial *improvement* to the solution.

In the implementation, the final values found by the GA and the cost function form the input to the algorithm. The values form the vertices of the simplex and at each iteration the worst one of these values is replaced. This is achieved by a number of trial evaluations of the cost function at the vertex after the simplex has been reflected, expanded or contracted. Fig. 6.7 shows how the Nelder-Mead procedure will find a lower value than that given at the end of the GA iterations. However, in this value the parameter settings found are no lower than those of the minimum value found in the course of the GA iterations. Therefore, applying the Nelder-Mead procedure should certainly improve upon the final GA result but the effect is to deepen that result rather than find a global minimum within parameter space, as described above.

## 6.6 Adding time as a factor

---

Additional improvement to the cost function was achieved by including the time taken to complete the segmentation for each parameter set as a factor. The ra-



## 6.6 Adding time as a factor

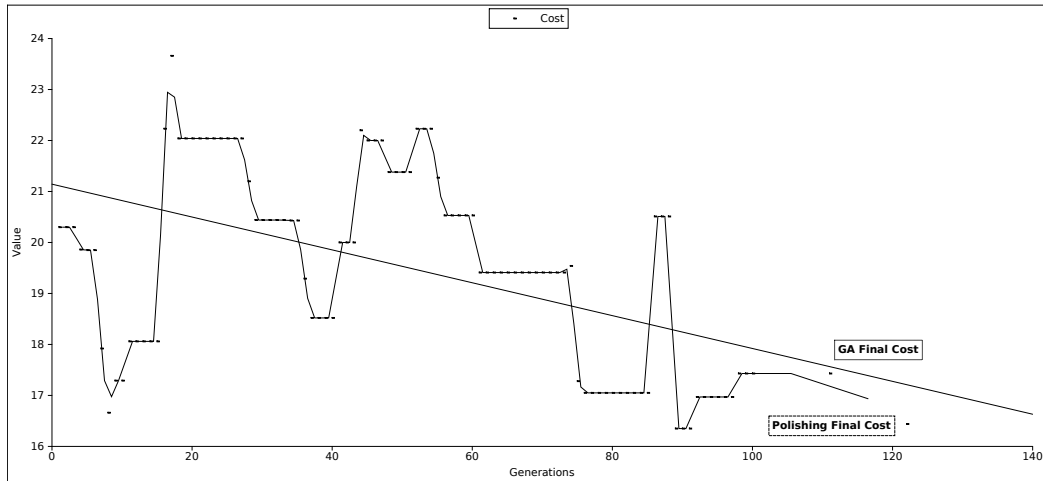


Figure 6.7: Application of Nelder-Mead polishing to GA optimisation applied on Graph-base segmentation, showing the final GA derived value and the value found after further iterations of the polishing algorithm.

rationale behind this inclusion is that the quality of the segmentation results is not the only value one would like to improve but there is also a need to balance this with the segmentation processing time performance. The ultimate aim is to improve both the segmentation quality and the computational performance of the segmentation process. Because even if a single segmentation execution can be improved to achieve a near real-time performance, the aim is to take in consideration that: firstly, the segmentation process is but a small part of the whole image understanding/computer vision framework, and secondly, with a multi-image database or multi-video feeds to process, the segmentation process will be executed multiple times per second, and as such the computational performance of the segmentation process plays a very pivotal role. And, as will be illustrated in the examples of the experiments detailed below, this factor gives some insight into the significance of some of the algorithms' parameters not only in respect to the processing time performance but also to the segmentation qual-

ity as defined by the segmentation evaluation method used. For instance, when experimenting without using the time factor the GA module will randomly vary certain parameters that actually don't affect the overall quality of the segmentation after optimising the significant parameters for the algorithm. However, this parameter variation can have a great effect on the processing time performance. So adding the time factor will still optimise the parameters to the same set as far as the significant parameters are concerned as before, while leading to a better performance, as will be illustrated in the results in the coming sections.

Also there is another advantage to the use of this time factor: It not only optimises the parameters for the segmentation algorithm itself according to the processing time performance, but also optimises the whole evaluation process, because the GA module is inclined to choose parameters that are time efficient. As a consequence the whole evaluation process take less processing time to complete.

The author is not aware of any extensive research on adding the processing time performance of the segmentation as a cost function factor to improve the segmentation algorithm parameters choice that balances between the segmentation quality and the computational performance. The research reported in [Everingham \*et al.\* \(Everingham \*et al.\*, 2001, 2002b\)](#) is the only notable work that discusses adding the segmentation processing time as an additional cost function for evaluation purposes. The authors of this prior work have emphasised the importance of considering time as a factor for the expected trade-off found for some algorithms between the segmentation accuracy and the processing time required to obtain a segmentation. However, the authors fail to discuss in detail what type of parameters were used and varied for each segmentation algorithms in the research they undertook.

### 6.6.1 The different models of adding the time factor

Adding processing time to the cost function can be taken using four different models. The first three are by combining the time factor with a segmentation quality evaluation factor: The first two are elementary arithmetic operations: addition and multiplication. The third, is as an argument of an exponential function. The fourth option is just to use the time factor alone. If the aim is to add extra emphasis to the processing time over the segmentation accuracy, then the timing as an input to an exponential function can be added to the cost function. There is one more model that uses only processing time value as the cost. However, this is not our aim here, as this is useful only if the timing performance is the only factor optimised.

The original cost function could be evaluated in two different ways. Either this can be using supervised methods which evaluate relative to a reference image. For example by a pixel-wise assessment of segmentation accuracy compared to a hand-segmented ground truth images with weighted penalties for mis-segmented pixels according to their distance to the nearest correct segment's pixel, as experimented with previously.

Or an unsupervised evaluation method can be used. As an example of the latter, something as simple as the number of regions found in the resultant segmentation can form the basis of evaluation.

Supervised evaluation allows a direct comparison between a segmented image and a reference image which can provide a finer resolution of the evaluation and can be implemented to produce empirical results for multiple images. However, generating a reference image is a difficult, time-consuming task, and for some

applications can be hard or even impossible. Reference images can be either obtained by human subjects which makes it subjective to a great degree or using computer-synthesised reference images. However, as Haralick argues (Haralick, 2000; Zhang *et al.*, 2008), the solutions obtained using computer-synthesised reference images evaluation can hardly be generalised. Furthermore, for most reference images, there is no guarantee that one generated reference image, either human-segmented or computer-synthesised, is better than another. In other words, reference images are essentially subjective. Thus, supervised evaluation methods using these reference images are particularly subjective.

The unsupervised evaluation independence of the reference images makes it possible to work with a wider choice of applications, conditions and a variety of image types. It also make it distinctively useful as an automatic evaluation method for real-time computer vision system with an online segmentation module. Furthermore, it is suitable for evaluating segmentation images that are not known before starting the segmentation process, that is to say, online real-time system. This idea will be explored in more detail later.

Hence, unsupervised evaluation is an objective method compared to the supervised methods. From this point on the author will use objective evaluation methods to mean unsupervised methods and subjective evaluation methods to identify the supervised methods.

Early experimental results with objective evaluation were illustrated in Section 6.3. However, many other characteristics of the segmented images can form the basis for objective unsupervised evaluation. The objective methods will be explored in more detail in Section 6.6.4.

The processing time can be added as a cost function factor to both of these

evaluation approaches. The arithmetic addition method is usually used when the cost function factors are of similar types. This is not the case for the time factor and the results of the subjective/objective evaluation. As such this method is not suitable and our experiments with it gave no useful information in respect to significance of processing times. That is almost all evaluation result values are higher than the timing values and as a result the time factor values are overshadowed by the evaluation results values in the cost function.

A more useful model is using multiplication. This provides a scaling trade-off between the evaluation factor and the timing factor and as such provides a more significant result and insight into the importance of processing timing. This can be contrasted with using timing in an exponential function. In this case, the optimisation process is forced to optimise heavily to find a solution with low processing time values.

For experiments in this Chapter it was found that multiplication model provided the best trade-off between all the other methods and it will be used for the rest of the results illustrated below.

### 6.6.2 The timing experiments image data-set and GA parameters

For the experimental results below a sample image set from the Berkeley Database was used. For illustration purposes 20 images were chosen for the experiments. These images were chosen to be representative of the whole dataset, while also making sure they have the characteristics of natural images. For example, they consist of multiple objects and not only have one object in the middle, which is a

## 6.6 Adding time as a factor

---

characteristic of some of the photographic images from the Corel dataset that the Berkeley group used. However, the conclusions are applicable to other images, as in all cases from the results below a general trends holds with no deviation.

GA parameters were adjusted for the following experiments. The original parameters discussed above were modified, and unlike the low population values that were used above for examination purposes, the population size was increased to 20 and at first was limited to 100 generations. These values were closer to make a better balance between providing a wide choice of solutions for the GA to evolve in each generation and the efficiency of processing each generation as fast as possible by the GA. Also this takes into consideration that creating 20 populations in each generation provides a good value for the limited number of parameters that will be tested (between 3-6 parameters) as discussed earlier. The recombination rate was fixed at 0.6 and the mutation rate at 0.2. The recombination rate will control the crossover performed between the populations at the end of each generation. If there is no crossover, an offspring is the exact copy of parents. If there is a crossover, an offspring is made from parts of the parents chromosome. If the recombination rate is 1.0 then all the offspring of the next generation are made by recombining the parents of the preceding generation. If it is zero then the whole coming generation is made from the exact copies of solutions from the populations of the preceding generation (but this does not mean that the new generation is the same). The aim is to make the solution to have parts of the successful population from the old generation while providing some randomness to improve the coming generations. The mutation rate controls mutation in the solution of the population carried over to the next generations. If the mutation rate is 1.0 then all the solution in the current population will

be changed randomly, whereas when its 0.0 then there will be no mutation and nothing changed. This mutation operation is performed after the recombination operation is completed. The value of 0.6 for the recombination rate provides a higher recombination rate to provide enough randomness for the GA to find new more optimal solutions while building upon previous older successful solutions. The mutation rate comparably was not very high at 0.2 but still useful to not using mutation at all, as it helps the GA to not fall into a local minima solution, while making sure that the whole GA solution is random and reaches a good optimal and stable solution in an acceptable time. Overall, the GA module does a good job at reaching an acceptably stable solution in much less than 100 generations with the higher population size. In other words, the GA minimises the cost function as much as possible using the evaluation method provided and the constraint of the given segmentation algorithms and the parameters' ranges given.

For consistency, mean-shift segmentation was used again. Figure 6.8 illustrates the results of varying the same three parameters used in Section 6.5.1 for 100 generations without the Nelder-Mead polishing stage (using image ID 101850 from the Berkeley data-set). (The Nelder-Mead polishing stage was explained in more detail in Section 6.5.1.) The Figure illustrates the search performed by the GA at each generation. The spikes in the trend lines represent the three parameters. However, there is clearly a stabilising trend after 20-30 generations. In more detail, it can be noticed that parameter *radiusS* is stabilising at a value of 3, parameter *radiusR* at a value of 2 and the *ColourDistance* parameter stabilising is at a 10. While still noticing that the spikes in the trend lines are the result of the GA testing other values to find a more optimal solution and then returning

again to the most stable solution found earlier.

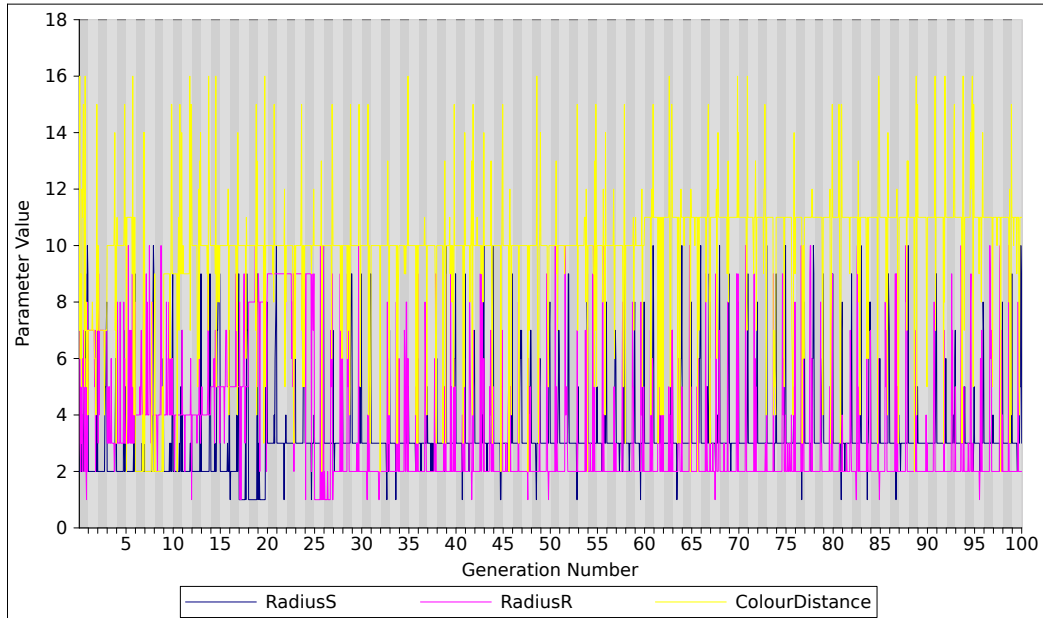


Figure 6.8: The Mean-shift GA evaluation with 100 generation, with three parameters  $radiusS$ ,  $radiusD$  and the Colour Distance values used for parameters space search

To have a better examination, Figure 6.9 provide a zoomed view of the first three generation. Each generation consist of 21 data points on the axis: the first 20 points correspond to the first 20 values of the population and the final 21st data point is the best result of those 20 values of the population results according to the cost function and as chosen by the GA. This last result is used as an input for the next generation in the GA computation process.

The spikes mentioned above in the distant view of the 100 generations are clear here in the varying values of the parameters. This implies that the GA searches for the best parameters according to the given cost function (in this case golden-truth subjective evaluation multiplied by the time factor). It's clear that the GA didn't reach a stable solution in the first generation even with a



## 6.6 Adding time as a factor

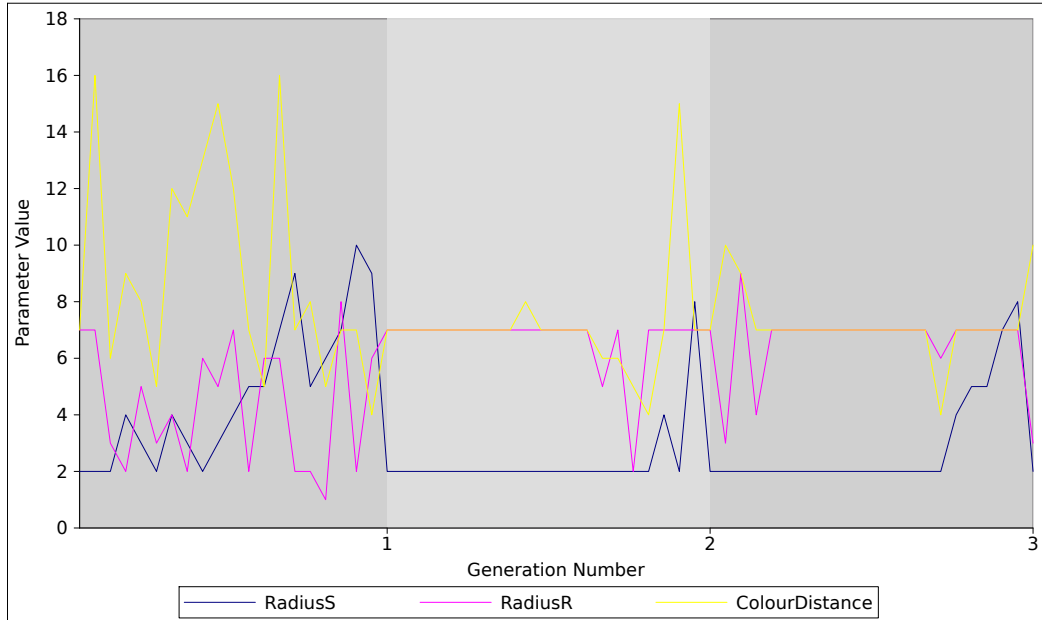


Figure 6.9: The first 3 generations of the Mean-shift GA evaluation each showing 20 populations and the final 21st result is chosen as an input to the next generation

population size of 20. However, it is also clear that in the second and the third generations the parameters' variation is less evident and a trend is observable. For example, *radiusS* tends to stabilise on value two, while the *radiusR* parameter by contrast stabilises at value 5.

This stable trend is clear in contrast with the rapid variation in the first generation in all the parameters. Its important to note here that there is still some possibility that this trend can change if a better parameters' combination is found in the coming generations.

Looking at a 20 generation version of the same Figure, as seen in Figure 6.10, it be can see that *radiusS* was actually lowered to value 1 and *radiusR*'s stable value increased to 8, while the *ColourDistance* parameter stabilised at a value of 10 in the last 10 generations of this Figure.

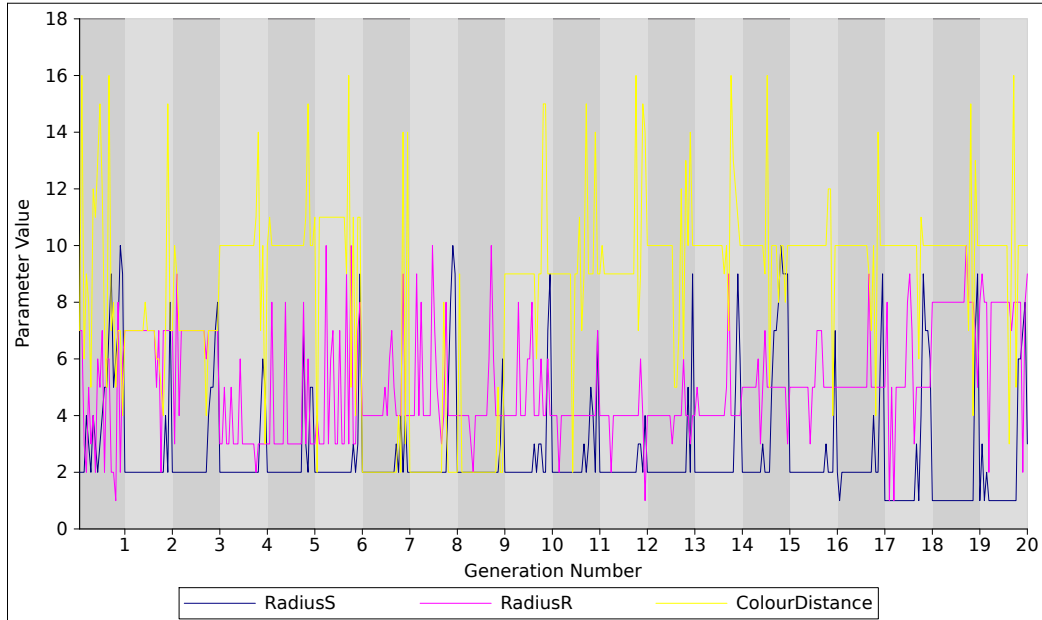


Figure 6.10: The first 20 Generation of the Mean-shift GA evaluation each showing 20 populations and the final 21st result is chosen as an input to the next generation

Now compared to the 100 generations version the 20 generations have reached a good combination of values for the parameters that minimise the cost function as much as possible. After experimenting with different settings and adding the polishing stage over the different segmentation algorithms, it was concluded that, for the rest of the experiments, 20 generations is more than enough for the GA to reach a stable solution and that actually after 30 generations without the polishing stage that the trend doesn't change at all.

So the combination of using a 20 generations' setting, in addition to the —computationally cheap— polishing stage provide a good trade-off of a computationally efficient evaluation, while still providing similar results to using 100 generations or more. Therefore, for the rest of the experiments and results shown in the following Section, 20 generations and a polishing stage were used as the

standard settings, in addition to setting the recombination rate at 0.6 and the mutation rate at 0.2.

### 6.6.3 Application of the time-weighted parameter search experiment

The following Sections will illustrate some of the results obtained using the time-weighted parameter optimisation experiments and the improvement achieved in both the time taken to complete the search test and the insight on the effect of some parameters on the time taken to perform the segmentation process. The results included below provide a representative example of other results obtained in testing experiments with other algorithms and under different settings. Additional results are provided in Appendix B.

To illustrate the effect of adding the time factor, two type of figures will be used for the experiment discussed below. The first showing the parameters' results before and after adding the time factor, on the x-axis the image ID is shown, while the y-axis represents the parameter values.

It's important to note here that the parameters' values have different meaning for different parameters and as such they don't have a relation across parameters most of the time. However, what is important is not the cross relation between parameters but the effect of adding the time factor on each parameter in isolation before and after the time factor.

The second type of figure illustrates the timing taken to complete the parameter search for each algorithm with and without the time factor. This type of figure illustrates any time-wise performance optimisation, which, as will be

illustrated below, is a significant improvement.

These results are also provided across multiple images, and don't illustrate the parameter variation effect on individual image segmentation quality and timing performance. This aspect will be explored in more detail in Chapter 4.

### 6.6.3.1 Colour edge detection evaluation

For colour images, edge detection can be applied by computing each colour channel separately to provide gradient results and then combine the results from different channels to classify the edges for the output. The Gevers & Stokman implementation used in this section defines three type of edges: material, high-light and shadow edges (Gevers & Stokman, 2003a).

The material edges correspond to edges usually found by a thresholding/non-maxima suppression technique, and the latter two edge types are additional edges that are considered as 'neighbour' edges to the material edges. Therefore, the first two parameters, the highlight edges' threshold parameter and the shadow edges' threshold parameter correspond to the computation of those two latter edges: highlight edges and shadow edges.

The values define how much the edges will be highlighted in the final segmentation output: higher values means higher highlight, while lower values means less highlight. These two parameters are varied between 1 and 255.

The third parameter correspond to an uncertainty factor that is used to consider if the gradient found in the pre-processing smoothing filter is an edge or not. Higher value means more "uncertainty" will be applied and less chance that a gradient will be considered as an edge. This parameter is varied between 1 and 5.

The parameters' values and ranges are used as suggested by the original algorithms authors recommendations and the algorithmic limits of the parameters. Therefore, for example, edge parameters can vary between 1 to 255 because the input image is encoded using these values. Thus, edges will be within this range. The parameters are used as input parameters to execute the segmentation algorithm. Each parameter set's choice computed by the GA is executed and then the segmentation output is used with the evaluation method to calculate the cost function, which finally the GA uses to evolve newer "fitter" solutions for the next generations.

Figure 6.11 shows the optimal results (as defined by the GA) for the three parameters found by the evaluation framework with and without the timing factor. On first observation, no clear trend can be distinguished. However, there are two important outcomes that are explained below.

Firstly, even with certain relatively high values of highlight and shadow parameters, on the overall range (1-255) of search they are low values (not more than 35) for certain images. This can be attributed to the composition of that certain image that needs a specific parameter set.

The second outcome is related to the uncertainty factor parameter which, with or without the timing factor, is always relatively stable between either high values of 4 or 5, which means that the evaluation is trying to minimise finding neighbourhood edges. There is less chance that a gradient will be labeled as a segment edge, as detailed above. This can be explained because the ground truth images that are used to calculate the cost function are done by humans. That is they focused on the main object(s) in the input image and discard segments of small details in the image. However these details are quite easy for the segmentation

## 6.6 Adding time as a factor

algorithms to pick up and identify as segments. This is a general trend that will be repeated with all the other segmentation algorithms when hand-segmented images are used as ground truth.

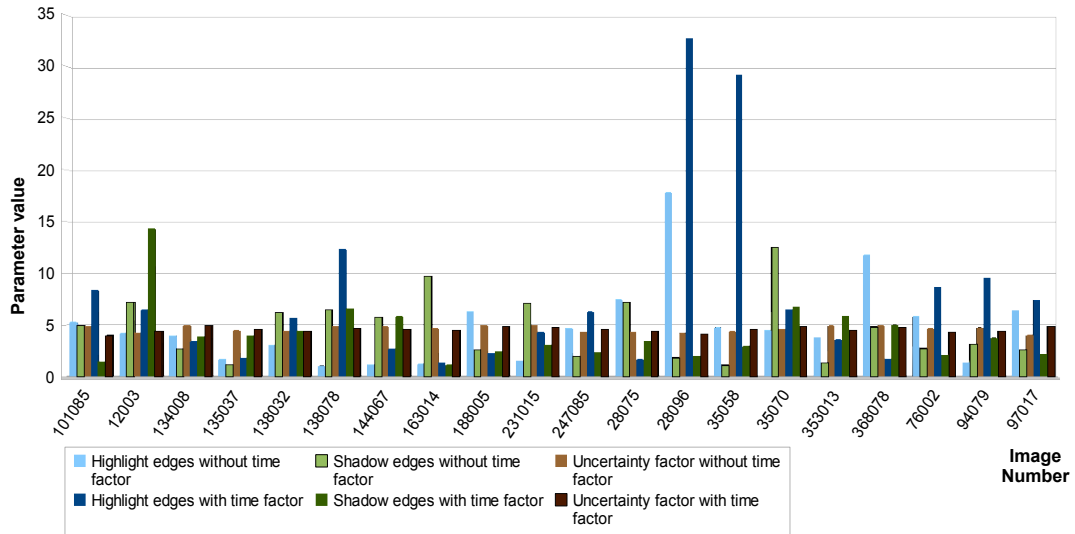


Figure 6.11: Colour Edge Detection evaluation 20 images with and without time factor parameters

Figure 6.12 shows the evaluation timing for the colour edge detection algorithm with and without the time factor, the results clearly show that for the images under test the overall timing is halved while using the time factor.

With all the previous results there is no single parameter that can be singled out by the evaluation with the time factor as a time intensive parameter.

### 6.6.3.2 Watershed segmentation evaluation

The watershed segmentation can be considered as a new approach to segmentation. It's related to the idea implemented in region growing segmentation paradigm. The idea is to still look for 'discontinuities' in the images to define

## 6.6 Adding time as a factor

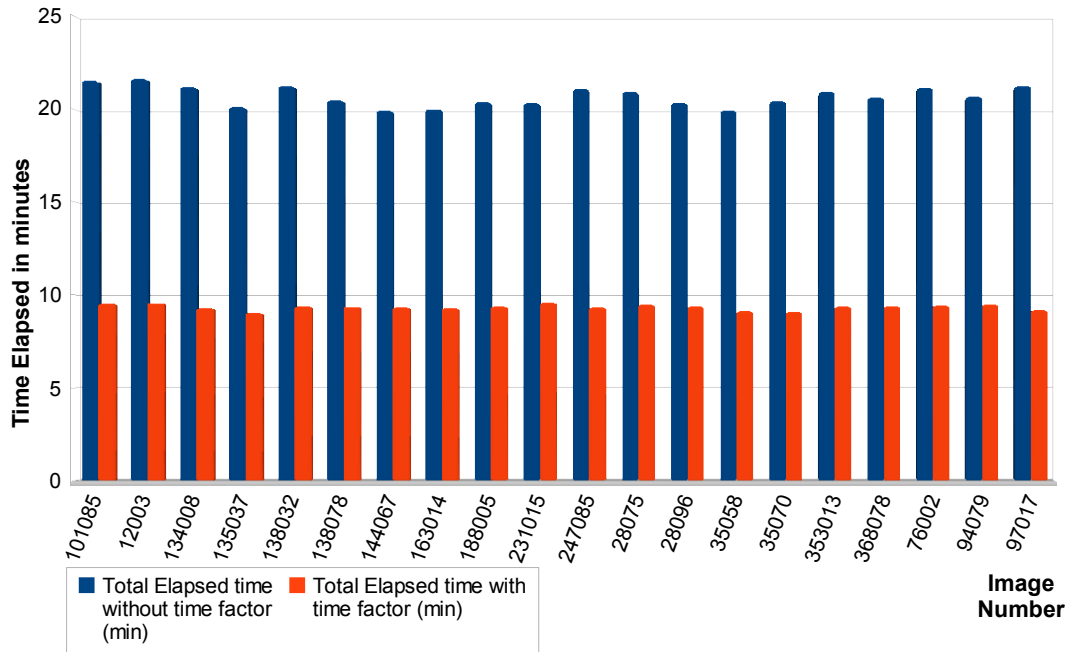


Figure 6.12: Colour Edge Detection evaluation 20 images with and without time factor timing

the segment edges. Generally, the discontinuities are defined on the colour levels. However, other image features can be used such as, for example, texture.

For testing purposes, and to contrast with other algorithms, a pre-processing colour quantisation stage was added to the normal watershed segmentation implementation. This stage provides an image ‘smoothing’ effect to the input image before processing by the core watershed algorithm.

Figure 6.13 illustrates the GA optimal results found by the evaluation with and without the time factor.

The main three parameters used were firstly a watershed threshold parameter for the core watershed algorithm. The parameter values’ ranges was chosen to reflect the original authors recommendations. This parameter is varied between 1 and 80. The second parameter is the number of colours parameter (*numberOf-*

## 6.6 Adding time as a factor

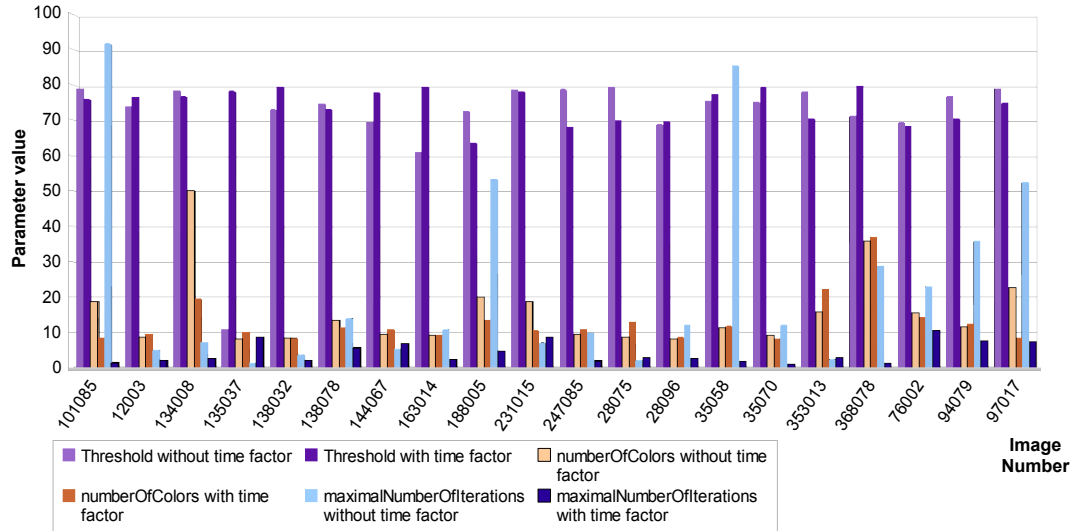


Figure 6.13: Watershed segmentation evaluation 20 images with and without time factor parameters

*Colors*) for the quantisation stage. In these tests, a k-means colour quantisation stage was used. The number k here means the number of colours that the image will be reduced to. Similar colours are clustered together according to the nearest pixel colour. This varied between 8 and 256. If the number of colours in the image (for example 256) is actually less than or equal to the number given by the parameter (for example 256) then no quantisation is performed on the image.

The last parameter is the maximum number of iterations (*maximalNumberOfIterations*) parameter for the k-means algorithm. This is a parameter that controls how many iterations are carried out to search the pixel's neighbours for colour similarity as part of the quantisation process.

The first point to observe is that the threshold is always very high, higher than 60, and there is no difference in this between using the time factor or not using it. The reason can be attributed to the point mentioned above about



using the hand-segmented images as ground truth. Higher thresholds eliminate more of the smaller details and segments that are not noticed by the human hand-segmenter and as such the evaluation tends to prefer higher values for the threshold parameter.

The other point to observe is that the evaluation is trying to optimise the parameter set always with lower values for the colour quantisation parameter, which means a higher smoothing effect on the input images. The quantisation parameter was not higher than 25-colour palette as target for all the images in the test but two, and actually the maximum is 50.

Noting that the maximum range is up to 256, then 50 is a relatively low number and results in high quantisation for natural images full of colours. However, there is no specific parameter value that is general for all the images.

This can be attributed to the fact each image will have specific original colour palettes. Also not all objects correspond comparably to the colour quantisation process and not all objects are identified equally by the quantisation process.

The final observation which has not been mentioned before is that the timing factor singles out the iteration parameter for ‘extra’ optimisation. The iteration parameter by definition means more computation time and as such the evaluation with the time factor keeps this parameter value as low as possible, not more than 10 iterations for all the images.

The evaluation without the time factor results gives no clear preference for high or low iteration values, which make one conclude that this parameter doesn’t have much importance for the segmentation accuracy. The evaluation without the time factor gave no preference for a parameter set with low computation time and as such did not consider low iteration values as an important target for

optimisation.

Figure 6.14 illustrates the time taken to complete the evaluation with and without the timing factor for the watershed segmentation. As before, the advantage of using the time factor is clear in reducing the overall time for completing the evaluation by at least half if not more.

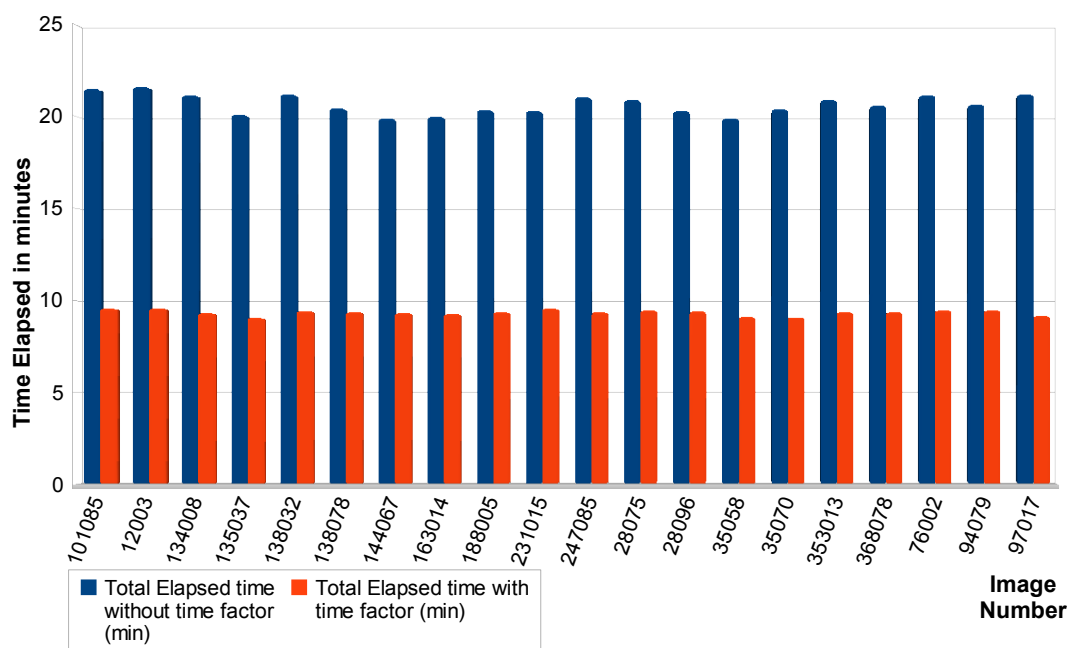


Figure 6.14: Watershed segmentation evaluation 20 images with and without time factor timing

### 6.6.3.3 Colour Watershed segmentation evaluation

Another variation of the watershed algorithm is the colour watershed implementation introduced by [Alvarado \(2004\)](#). This implementation uses additional pre/post-processing stages while still maintaining the same watershed algorithm as a core processing stage. The parameters used here are related to the different

processing stages. The kernel size parameter (*kernelSize*) is related to the pre-processing noise reduction stage using a median filter. This parameter is varied between 1 and 10, with higher values meaning a higher noise reduction effect on the input image. The watershed threshold factor parameter (*minPropForWatershedThreshold*) is related to the watershed segmentation core processing stage. The algorithm by design prefer over-segmented results to be processed by the post-processing stages. This parameter is varied between 0.0 and 0.5 values, with change resolution of 0.01. If the value is closer to zero then an over-segmented results will be produced, and higher values closer to 0.5 will produce an under-segmented results. The last two parameters are related to the post-processing stage of region merging. The merging threshold (*mergeThreshold*) parameter control what regions are merged. This parameter is varied between 0 and 1, with change resolution of 0.01. If only very similar regions need to be merged then this parameter needs to be set closer to zero. By contrast higher values closer to 1 make the merging process more tolerant and as a consequence more regions are merged.

The other parameter is the minimum number of regions (*minRegionNumber*) parameter. This used as an additional controlling parameter as a target for the number of region to be produced by the algorithm. This parameter is varied between 10 regions and 1000 regions. In this case the parameters' evaluation is divided into two different figures because the parameters have different scales, and the minimum number of regions parameter has a wider range of value. As such the other parameters will otherwise not be clear in the illustrative figure. Figure 6.15 shows the evaluation results for all the parameters except the minimum regions parameter. From the figure, one can observe that the parameters related

## 6.6 Adding time as a factor

to the first two processing stages: the noise reduction and the watershed stages, had fluctuating optimal values (as found by the GA) across all the image dataset, and a similar result was found when running the evaluation framework on the whole Berkeley dataset. However, the same can't be said for the region merging stage where the evaluation found a stable optimal result across all the images. Therefore, for the example in this figure it's clear that the value of the merging threshold is optimised closer to a value of 1. This means that more regions are merged and a lower number of regions are produced in the final segmentation output.

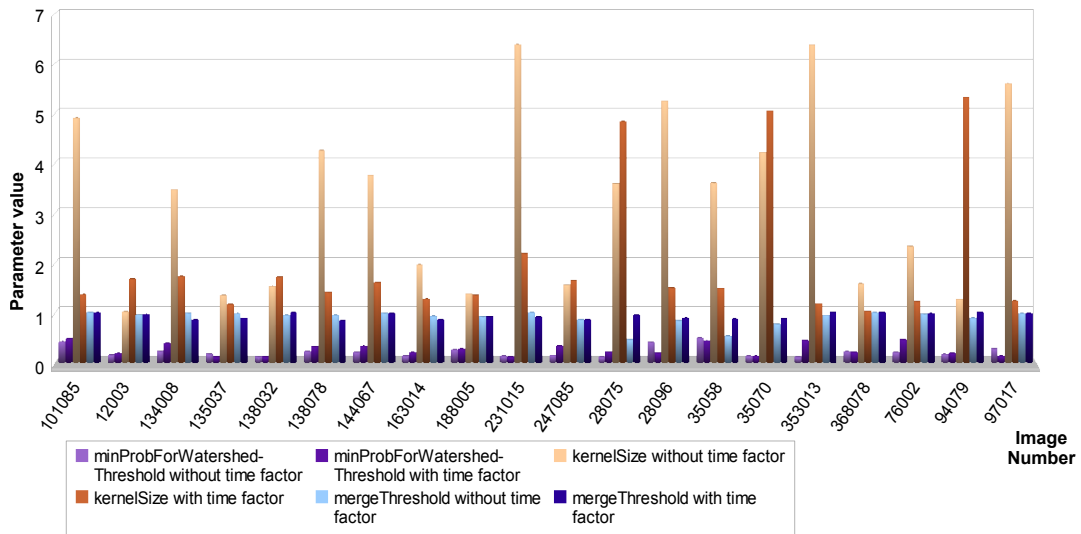


Figure 6.15: Colour Watershed segmentation evaluation 20 images with and without time factor parameters

Figure 6.16 illustrates the evaluation results for the minimum number of regions parameter. The values here don't have a stable value between all the image set. However, in general they tend to be low compared to the range given for the search between 10 and 1000. The value most likely is very image specific.

However, both parameters for the merging stage tend to optimise for output segmentation with less number of regions. These under-segmented outputs correspond to the hand-segmented ground truth.

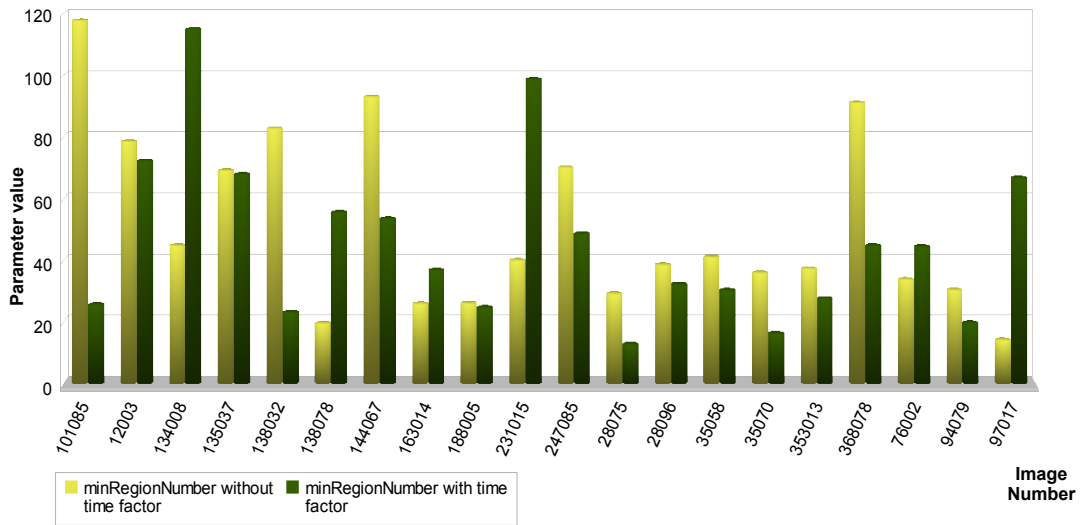


Figure 6.16: Colour Watershed segmentation evaluation 20 images with and without time factor parameters

Figure 6.17 illustrates the timing results for the GA evaluation and like the previous trend found above, there is a clear advantage when using the time factor to reduce the overall time taken to complete the evaluation.

### 6.6.3.4 K-means segmentation evaluation

The K-means segmentation algorithms is categorised as a clustering algorithm like the mean-shift algorithm which will be discussed below. The algorithm implementation uses colour information in the input image to create similar clusters and these cluster are considered at a later stage to find what constitutes segments in the image.

## 6.6 Adding time as a factor

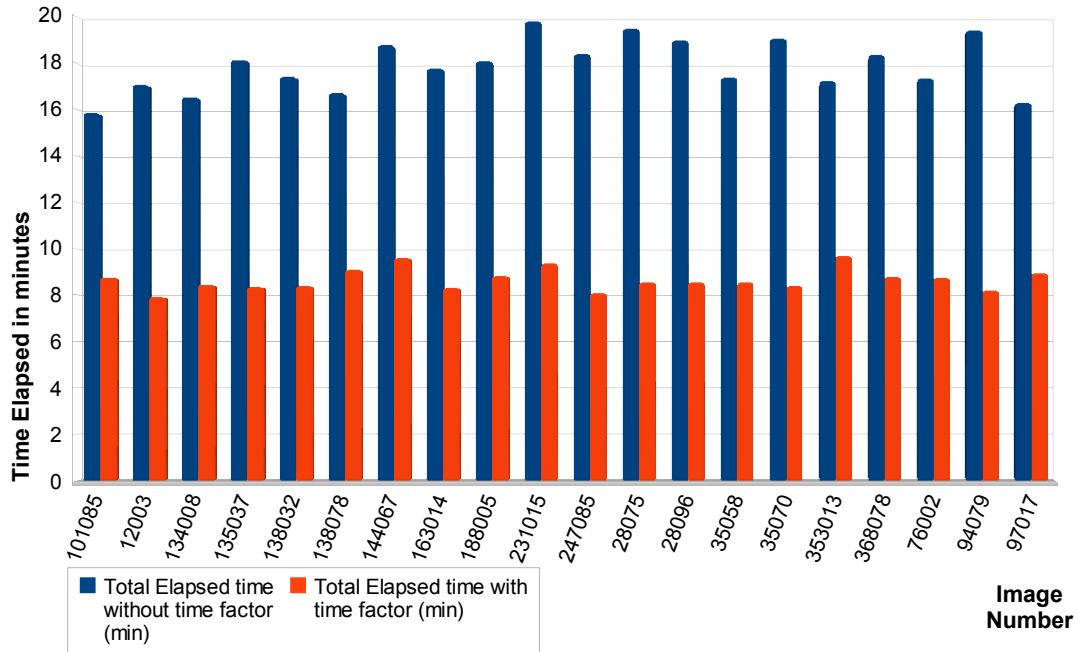


Figure 6.17: Colour Watershed segmentation evaluation 20 images with and without time factor timing

The algorithm uses firstly a k-means-based method to quantise the colours of the input image. This can be tweaked using three parameters: 1) a number of colours parameter (*numberOfColors*) used for the output image colour palette, this parameter is varied between 16 and 4 colours; 2) a threshold delta parameter (*thresholdDelta*) which is used as a factor to stop the algorithm processing iterations if the palette changes between the iteration is less than this value, this threshold value variation is given a range between 0.1 and 1.0; and finally 3) the maximum number of iterations parameter value, which is varied between 1 and 100 iteration. This stage is very similar to the smoothing/quantisation stage used for pre-processing in the watershed algorithm, as discussed above. This stage is followed by another smoothing stage that uses a k-nearest-neighbour smoothing algorithm to smooth out the image further. The kernel size (*KernelSize*) param-

## 6.6 Adding time as a factor

eter is used to adjust this stage processing. The value is varied between 3 and 5 (odd numbers). Interestingly, in this case both of these stages can be considered as a pre-processing stages rather than a ‘segments-finding’ stages. The smoothing stage in effect acts as an extra noise removing and segment defining stage. And the quantisation stage does most of the work at the start of the segmentation process. Figure 6.18 illustrates three parameters: kernel size, number of colours and the threshold delta parameter. The general observation is that there is no significant difference between the results with the time factor and without it.

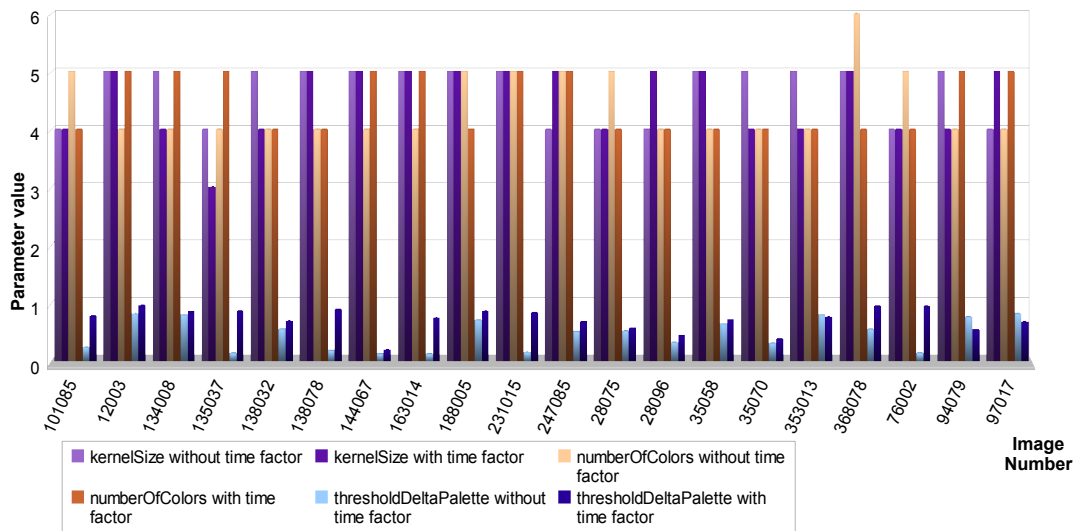


Figure 6.18: K-means segmentation evaluation 20 images with and without time factor parameters

The evaluation optimises for lower number of colours to quantise and higher kernel size for the smoothing filter, those two trends highlight the optimisation for segmentation outputs with a less number of segments in general and as such are closer to the hand-segmented ground truth images. There is no clear trend for the threshold delta in this results set. Further examination of the results of

## 6.6 Adding time as a factor

the values of the iteration parameter shows the improvement from adding the time factor to the evaluation. While using the time factor the evaluation always optimises for lower number of iterations (less than 20) compared to the results without using the time factor. This result is illustrated in Figure 6.19.

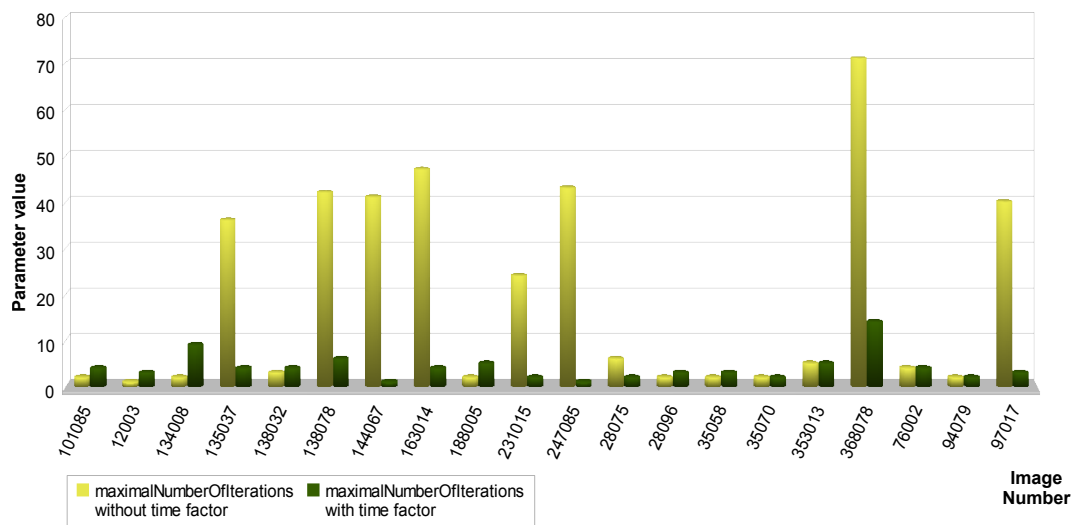


Figure 6.19: K-means segmentation evaluation 20 images with and without time factor parameters

In a similar way, using the time factor again provides much improvement of the algorithm processing time. The improvement is not the same between different images. However, there is always a clear improvement. This result is illustrated in Figure 6.20.

It is also important to point out again that in the case of a high throughput system, where the need is to process a high number of images not only 20 images but hundreds of thousands of images (or video frames) if not even more, any time improvement, however low, is always desirable.



## 6.6 Adding time as a factor

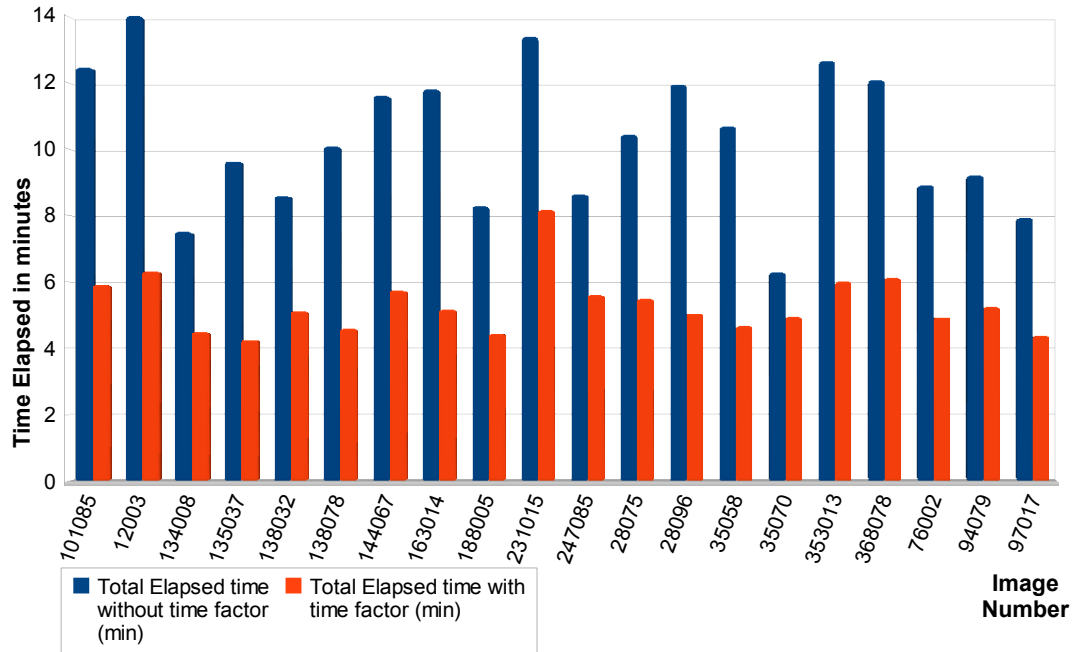


Figure 6.20: K-mean segmentation evaluation 20 images with and without time factor timing

### 6.6.3.5 Mean-shift segmentation evaluation

The mean-shift segmentation algorithm is another algorithm that can be categorised as a clustering algorithm, just as the k-means algorithm above. This test uses the algorithm implementation provided by [Comaniciu & Meer \(2002\)](#) in their Edge Detection and Image SegmentatiON (EDISON) System with some modification to expose the parameters used<sup>1</sup>. Like the k-means, the mean-shift algorithm uses the colour quantisation process as the main processing stage to achieve its goal of producing a segmented image. This means mainly that the segments relate to the colour of the objects in the input image, and while colour is not the only feature that defines an object in nature it's quite easy to identify and

<sup>1</sup>The authors' code can be found on the following link: <http://www.caip.rutgers.edu/riul/research/robust.html>

define in an algorithm compared to other features (such as texture for example).

Figure 6.21 illustrates the evaluation of the first three parameters for the mean-shift algorithm: The mean-shift search window radius in the colour space ( $radiusR$ ) and the window spatial radius in the image grid space ( $radiusS$ ) parameters. In addition to the maximum number of trials for choosing a colour of the chosen region/pixel defined as ( $maxNeighbourColorDistance$ ) in the mean-shift search process. All of these parameters are varied between values of 1 to 10 by the evaluation process. In the case of  $radiusS$  and  $radiusR$ , this represents the window size and the maximum number of trials for the last parameter.

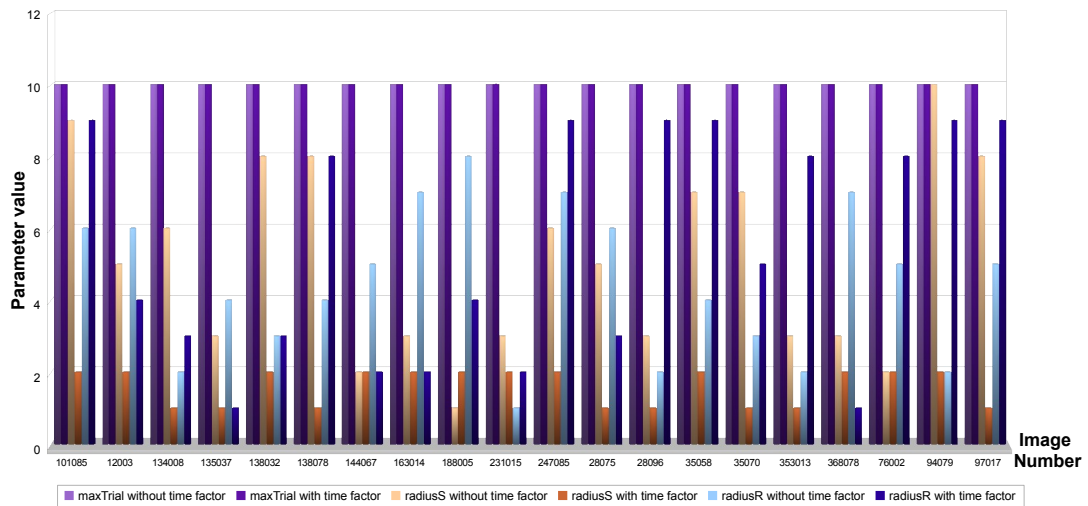


Figure 6.21: Mean-shift segmentation evaluation 20 images with and without time factor parameters

The main observation here is concerned with the effect of using the time factor on the value of  $radiusS$  parameter. With the time factor, the value is always equal or less than 2 in the 1-10 range. While without the time factor, the same parameter value does not have a specific trend, and changes between

## 6.6 Adding time as a factor

different images in the test. The best explanation for this is that this parameter doesn't have a great significance for the quality of the segmentation. However, higher values of this parameter are computationally expensive. There is no similar trend for the *radiusR* parameter, and the time factor doesn't have any effect on the number of trial parameters.

The other parameter used with the mean-shift algorithm is the maximum colour distance between neighbouring regions (*maxNeighbourColorDistance*). The results for this parameter are illustrated in Figure 6.22.

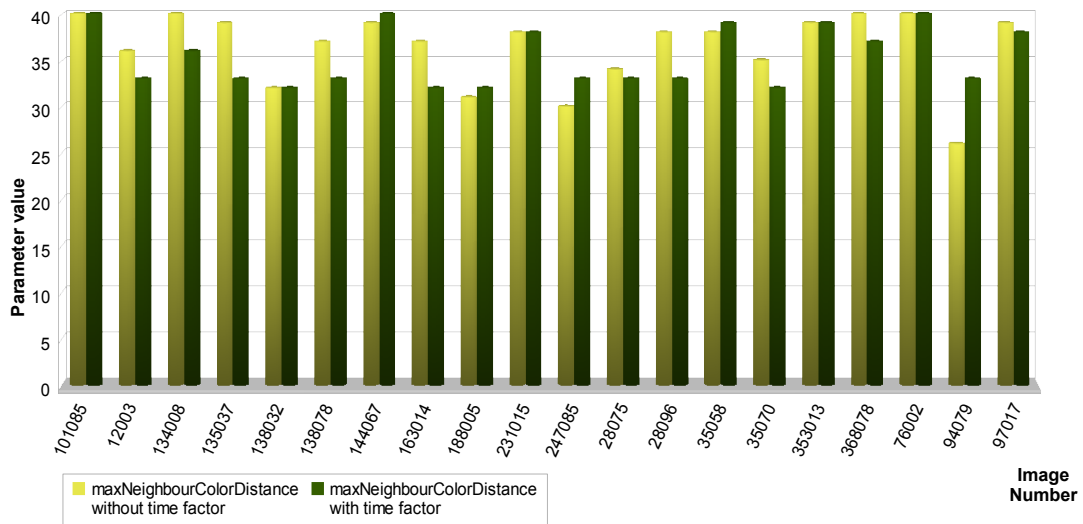


Figure 6.22: Mean-shift segmentation evaluation 20 images with and without time factor parameters for *maxNeighbourColorDistance* parameter

This parameter is varied between 2 and 40 value and although this parameter is generally always above 25, there is no clear influence of the time factor on changing the value selection for this parameter. Again in a similar trend the time factor addition to the evaluation process optimised the computational time performance for the whole process, where the time taken to complete the evaluation

## 6.6 Adding time as a factor

for each image is always less than 35 minutes. Figure 6.23 illustrate this result.

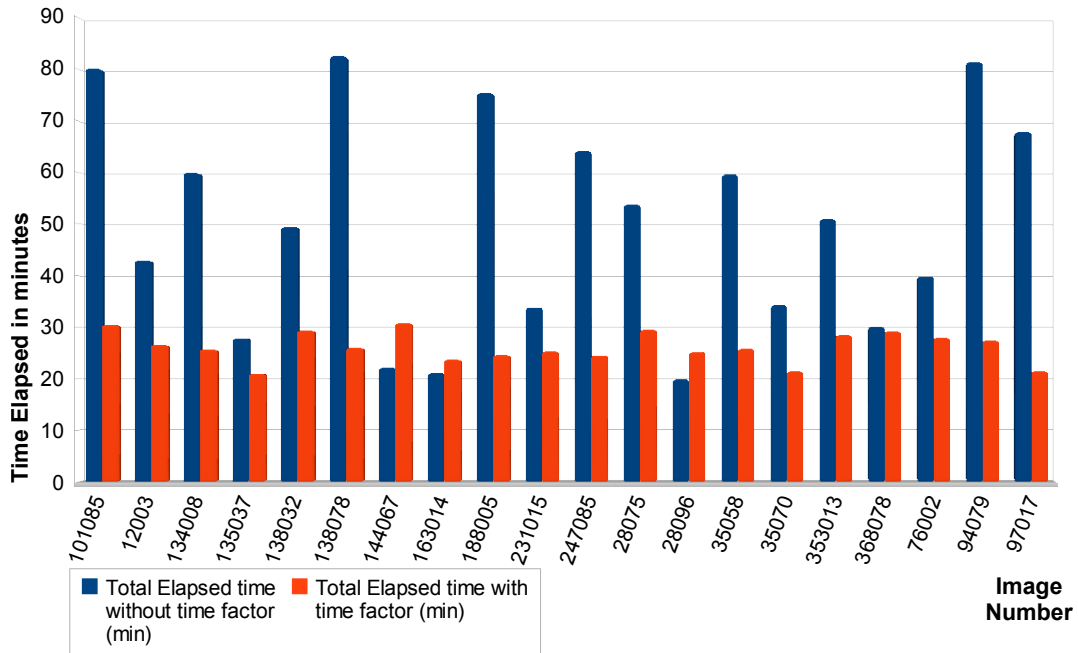


Figure 6.23: Mean-shift segmentation evaluation 20 images with and without time factor timing

Although in some cases, such as image 144067 for example, the time taken without the time factor is less than the time taken with the time factor for the evaluation by about 10 minutes, the overall improvement over all the images set under the test is still greater. Even though some images has a different execution time for the segmentation process, the parameter choice has a bigger effect on the segmentation execution time. Overall, the parameters can be optimised to perform a computationally efficient segmentation process while also not losing any of the required segmentation quality in the process.

### 6.6.3.6 Graph-based segmentation evaluation

The graph-based segmentation algorithm introduces a new implementation idea to finding segments in images and for this experiment Felzenszwalb & Huttenlocher's implementation was used. This implementation has three main parameters: a smoothing value (*smoothingSigma*) for a pre-processing Gaussian filter, a threshold value (*thresholdK*) for the core thresholding stage, and finally a minimum segment size parameter (*postSgmntSize*) imposed by a post-processing stage.

An interesting observation to note is that the graph-based algorithm is composed of three distinct processing stages, some of these stages were used in earlier algorithms' implementation (like pre-processing smoothing filters, or post-processing size-enforcing stages) to be used here again. This type of segmentation algorithm design keeps repeating itself even across different segmentation algorithm categories in the taxonomy. Figure 6.24 illustrates the evaluation results of the smoothing filter parameter. This parameter is varied between 0.01 and 1.0 range with change resolution of 0.01, following the algorithm's authors suggestion. In general this value is larger than 0.4. However, there is no specific observation on the effect of the time factor on the value of this parameter across all the images.

Compared to previous results we should expect a clear trend of a higher smoothing value for this parameter. However, this not the case. This can be explained by the use of the post-processing stage that can have a greater defining effect on the output segmentation that over-shadows the effect from the smoothing filter. As such the GA evaluation prefers to optimise on that parameter value.

Figure 6.25 illustrates the evaluation results of both the threshold and the

## 6.6 Adding time as a factor

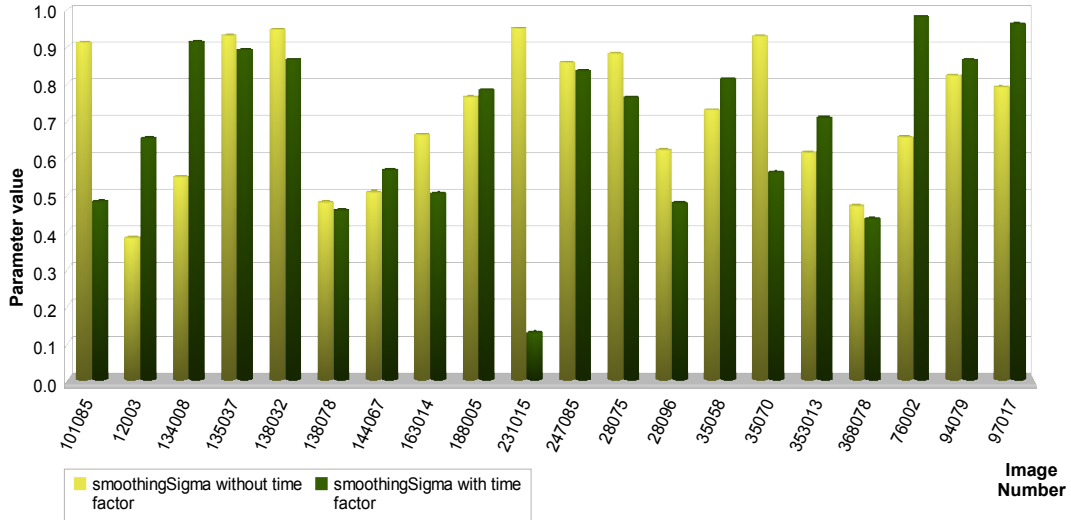


Figure 6.24: Graph-based segmentation evaluation 20 images with and without time factor for *smoothingSigma* parameter

segment size parameter. The threshold parameter is varied between 10 and 1000, and the segment size parameter is varied between 1 and 2000 ranges both with change resolution of 1, following the algorithm's authors' suggestion.

The threshold value tends to be of a lower values, as the value is usually around 500. However, in a few cases it is even lower, as low as 100. The most significant observation was the value of the segment size parameter that affects the post-processing stage. It has a very clear trend in which the segment value is always 1500 or higher but in case of one image, image 28096, is lower than 1500. However, it is still higher than 800. This trend for higher segment sizes is the same with and without using the time factor. In other words, the evaluation framework is again trying to optimise the results to match the hand-segmented images, which have few segmented regions that focus on the main and centred objects in the image. Furthermore, the pruning stage is computationally cheap.

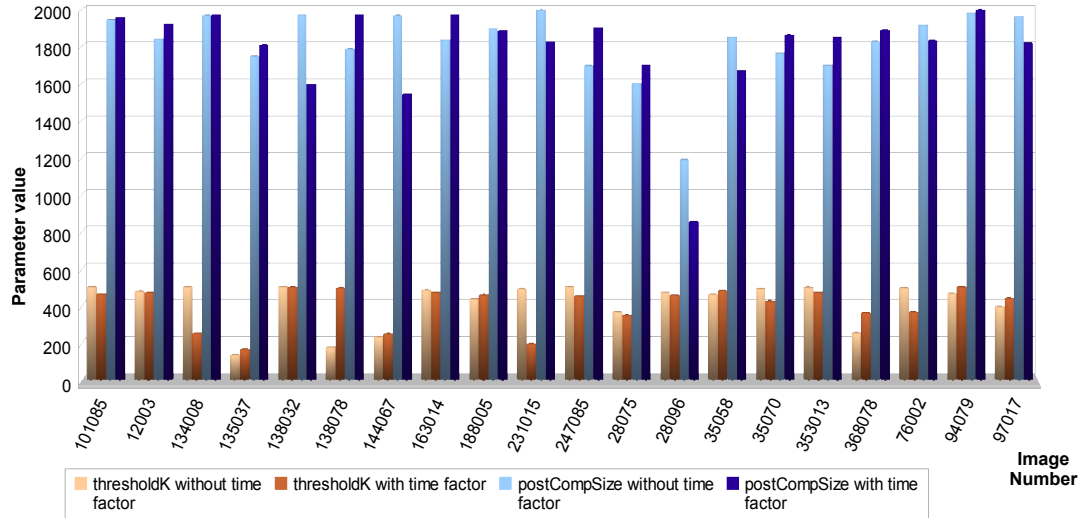


Figure 6.25: Graph-based segmentation evaluation 20 images with and without time factor parameters

Thus, the evaluation framework didn't find a reason to optimise it.

No single parameter has a higher computation time and, therefore, no such parameter will be singled out by the time factor test. However, the use of the time factor still provides the same time improvement in the overall set of experiments across all images. This result is illustrated in Figure 6.26.

#### 6.6.4 Applying time-weighted parameter search on objective segmentation evaluation

The other significant evaluation method for image segmentation is using objective techniques as discussed earlier in Section 2.4. As mentioned earlier in (Zhang *et al.*, 2008), others have already surveyed the use of objective evaluation methods for image segmentation. The aim here is to explore the effect of the time-weighted GA parameter search while using objective evaluation and comparing the seg-

## 6.6 Adding time as a factor

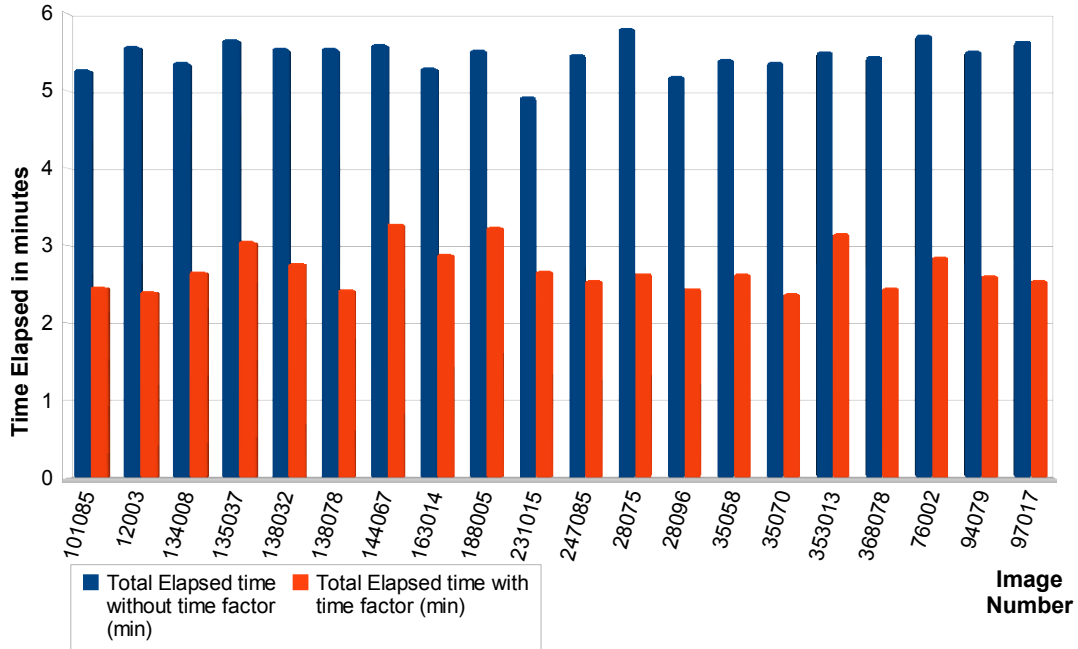


Figure 6.26: Graph-based segmentation evaluation 20 images with and without time factor timing

mentation evaluation results with the results achieved earlier with the subjective evaluation based on hand-segmented ground truth images.

### 6.6.4.1 Objective evaluation metrics definitions and equations

Objective evaluation methods (also known as unsupervised evaluation or standalone methods) by definition don't use any reference image to perform the segmentation evaluation. For objective evaluation, the  $F$  measure evaluation method was chosen as a representative example.  $F$  measure evaluation method was introduced by Liu & Yang (1994). This evaluation measures the average squared colour error of the segments. Additional improved measures were proposed by Borsotti *et al.* (1998),  $F'$  and  $Q$  measures, which build on the  $F$  measure definition to minimise any bias for over-segmentation. When a segmentation algorithm pro-



duces more segments than desired for a single object in the image, this outcome is called over-segmentation. The opposite outcome is called under-segmentation, when less segments are produced than needed to define the objects in the input image and as such two or more objects will be defined by one segment.

To define the  $F$ ,  $F'$  and  $Q$  measures, the following notations will be used. Let  $S$  is the input image with height  $S_h$  and width  $S_w$ , both in pixels. Let  $A_S$  be the area of the input image as follow:  $A_S = S_h \times S_w$ . A segmentation is defined as a set of  $R$  regions.  $P_i$  is the set of pixels in region  $i$ , and  $A_i = \|P_i\|$  denote the area of the region  $i$ .  $x$  denote a single colour component in the image, and  $V_x(p)$  is the value of component  $x$  for the single pixel  $p$ . Thus, the two important equations to calculate evaluation metrics are: a) the average value of a single component for each region, and from that we can get b) the squared colour error of each region. For the colour component  $z$  average value for region  $i$  equation can be defined as follow:

$$\hat{V}_z(P_i) = \left( \sum_{p \in P_i} V_z(p) \right) / A_i \quad (6.9)$$

The squared colour error  $e$  of region  $i$  and component  $z$  can be defined as follows:

$$e_z^2(P_i) = \sum_{p \in P_i} \left( V_z(p) - \hat{V}_z(P_i) \right)^2 \quad (6.10)$$

The  $F$  measure for image  $I$  with  $R$  segmented regions can be defined as follows:

$$F(I) = \sqrt{R} \sum_{j=1}^R \left( \frac{e_j^2}{\sqrt{A_j}} \right) \quad (6.11)$$

The  $F'$  measure for image  $I$  with  $R$  segmented regions can be defined as follows:

$$F'(I) = \frac{1}{1000 \cdot A_I} \sqrt{\sum_{a=1}^{MaxArea} [R(a)]^{1+1/a} \sum_{j=1}^R \frac{e_j^2}{\sqrt{A_j}}} \quad (6.12)$$

Where  $R(a)$  is the number of segmented regions in the segmented image with area size that equal  $a$  exactly, and  $MaxArea$  is the area of the largest region in the segmented image.

The  $Q$  measure for image  $I$  with  $R$  segmented regions can be defined as follow:

$$Q(I) = \frac{\sqrt{R}}{1000 \cdot A_I} \sum_{j=1}^R \left[ \frac{e_j^2}{1 + \log A_j} + \left( \frac{R(A_j)}{A_j} \right)^2 \right] \quad (6.13)$$

The  $F$  and  $Q$  measures already penalises over-segmentation by a weighting proportional to the square root of the number of segments. The  $F$  measure is also unaffected by the type or the contents of the input image and doesn't need any user defined parameters. As such it is a completely unsupervised evaluation method. So the  $F$  measure is a good representative objective evaluation method. For objective evaluation, the  $F$  measure evaluation method was chosen as a representative example. Further discussion on the author's findings will be provided

below.

### 6.6.4.2 Results of the $F$ measure evaluation

This test performs the same evaluation test illustrated earlier in Section 6.6.3.5 with subjective evaluation methods on the Mean-shift segmentation. However, instead of using supervised evaluation with hand-segmented reference images, the  $F$  measure was used as the evaluation method. This provided the cost function for the GA parameter search.

Figure 6.27 illustrates the objective evaluation results on the mean-shift segmentation similar to Figure 6.21. The only difference is the use of the objective evaluation  $F$  measure.

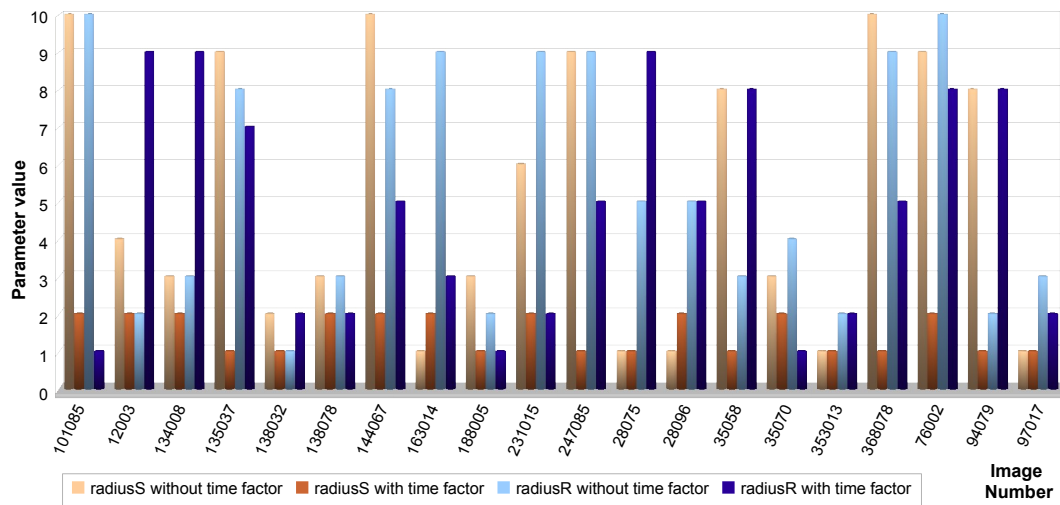


Figure 6.27: Objective evaluation of Mean-shift segmentation on 20 images with and without time factor for  $radiusS$  and  $radiusR$  parameters

The main observation is the effect of using the time factor with the objective evaluation on the value of  $radiusS$  parameter is similar to the effect found when

using the subjective evaluation of the hand segmented images. That is with the time factor the value of  $radiusS$  is always equal or less than 2 in the 1-10 range. While without the time factor the same parameter value does not have a specific trend, and changes between different images in the test. One conclusion here is that the  $radiusS$  is a computationally expensive parameter with high values and as such the time-weighted search method prefers lower values. Additionally, we can conclude that the  $F$  measure is very similar to the subjective evaluation methods. The reason is that subjective evaluation is dependent on the human hand-segmented images, which usually contain a very low number segments that only correspond to the human high-level understanding of the objects in the images. Likewise the  $F$  measure by design prefers segmentation results with less segments. Hence, the parameters for the segmentation methods tested above in Section 6.6.2 with the subjective supervised evaluation methods produced similar results and trends under the objective  $F$  measure evaluation to what was achieved earlier with the subjective evaluation.

In summary, the results found for  $F$ ,  $F'$ , and  $Q$  (in addition to other intra-region uniformity measures like  $ECW$ ) were similar to the results obtained by [Zhang et al. \(2008\)](#). These measures can be improved by decreasing the number of the regions found in the final segmentation results. Furthermore, segmentation methods that produce segments with few and uniform regions, choose the main objects (with uniform colours), and discards both the textures and noises in the input image scores higher with these measures. In other words, segmentation results very similar to the hand-segmented reference images were used in the supervised evaluation. In addition, there are other objective methods that can overcome some of this bias against noisy/textured images segmentation. However,

## 6.6 Adding time as a factor

they have other disadvantages too. The aim is not to use one measure, but a few of them while balancing between them. The objective evaluation and the subjective supervised evaluation will be discussed further below.

With regard to the effect of the time factor on computation performance of the evaluation, the trend is still evident and using the time-weighted search optimises the whole evaluation process. With the time factor the overall evaluation process doesn't take more than 38 minutes for any of the images while there it is not possible to predict how long it can take without the time factor as the results fluctuate. Figure 6.28 illustrates the result.

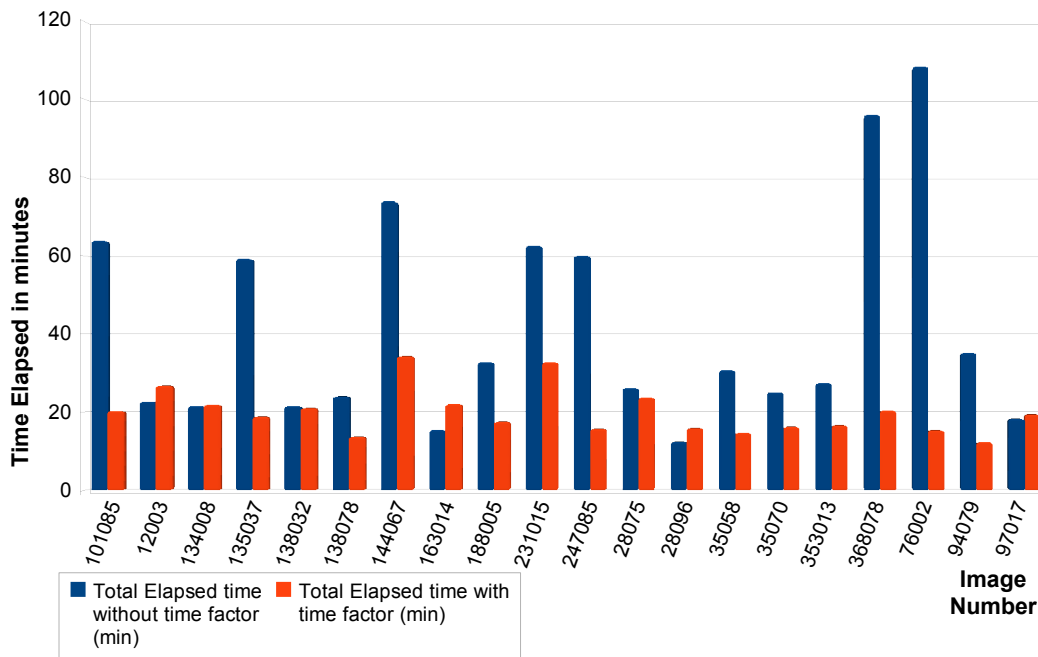


Figure 6.28: The timing performance of the objective evaluation of Mean-shift segmentation on 20 images with and without time factor

So for the case of using the evaluation framework for experiments with many images (from hundreds to thousands and more) or with video frames from a one to multiple streaming videos, the optimisation achieved with the time factor will

be too significant not to be utilised to improve the evaluation processing speed.

## 6.7 Concluding Remarks

---

### 6.7.1 Summary

This chapter introduced some new concepts that improve and speed up the segmentation evaluation process: using a GA to reduce the search time significantly; and introducing a polishing stage to improve the GA results and reduce the number of generations needed to reach the GA optimal solution and as a consequence improve the timing performance. Furthermore using a time-weighted cost function for the GA allows the GA to optimise the segmentation results on the basis of time in addition to the output quality.

All of these ideas provide improved computation performance with better quality (avoiding local minima solutions). This improvement is found in both the parameter search for individual images and also overall evaluation for a set of images. Also all of these ideas were explored in detail and tested in different research directions: choice of segmentation algorithms, choice of image set, and parameter type. There are some important observation that can be concluded from the results found, which are general observation that cover other segmentation algorithms not considered in this section for space reasons, even though they were explored during experimentation.

Firstly, the segmentation algorithms are ‘evolving’ with time while still using the old techniques as sub-stages in some way. For example some implementations of the edge detection algorithm uses thresholding as the second stage while also

tries to improve over the thresholding results by adding a pre-processing smoothing filter. Or a better example is the graph-based segmentation algorithm that uses a pre-processing smoothing stage, a core thresholding stage and finally a post-processing stage that imposes a certain size for the output segments.

While in some cases this ‘evolution’ is quite significant and introduces innovative methods to solve the same problem, like using a non-maxima suppression technique instead of thresholding as introduced by Canny and as such the Canny edge-detection method can be seen as a wholly new method. The author still thinks the links between these algorithms are significant and could span different branches of the taxonomy as introduced at the start of this thesis. Most similarities are found in the pre/post-processing stages that are used in different segmentation methods. However, these are not usually exposed to the algorithm user.

For example one of the processing components that was encountered regularly during this research is the smoothing filter, which is regularly used as a pre-processing stage. The colour segmentation algorithms use these smoothing filters to usually quantise the colours reduce the noise while preserving the edges in the images and the noise-reduction also provide regions with uniform colours which as a consequence reduce the number of the colours in the image palette before moving it to the core processing stage that will define the segments’ boundaries. It was clear from the results above that this pre-processing stage is a useful stage in providing a good segmentation results, while being a relatively cheap computation stage.

With subjective evaluation using human hand-segmented images the higher smoothing always provides better results that are very close to the hand-segmented

results provided by the Berkeley database. The higher smoothing levels correspond to a lower number of segments that define the objects in the images and in most cases closely resemble the hand-segmented results. Because the hand-segmented images are dependent on human perception of the image. And the human perceptions and high level semantic human understanding of the image are both also affected very highly by the colours in the images compared to the other features (i.e. textures, brightness, among others).

This work concentrated on colour segmentation. However, to some extent, some of the algorithms tested used stages that take into consideration the brightness and texture cues from the image. An example is the anisotropic-based segmentation algorithm which provides slightly better quality results in some cases, though with slightly higher computation cost. Additionally, the pre-processing stage was not the only significant stage for the segmentation quality. A post-processing stage that eliminates segments according to a pre-defined segments size or number of segments required in the output image also provides good result quality in the case of both the subjective and objective evaluation. It is also has a relatively cheap computation cost compared to the core boundary-defining stage of the segmentation algorithm.

### 6.7.2 Discussion

The segmentation algorithm design can be abstracted into a design that includes three stages: 1) a pre-processing stage; 2) a core boundary defining stage; and 3) a post-processing stage(s). Pre-processing stages are usually used to simplify the details in the image before applying the core segmentation processing. So in



## 6.7 Concluding Remarks

case of colour images, smoothing or colour quantisation is a good pre-processing stage. Post-processing stages deal with the outcome of the core segmentation stages as segments rather than image features. For example, they can be as simple as defining a certain size for the segments' sizes to a relatively more complex processing stages like for example creating a binary tree of the segments and combining a closely related segments according to a pre-defined parameter (Salembier & Garrido, 1998, 2000).

Figure 6.29 illustrates the idea explained above of the segmentation design including the different stages using the graph-based and the mean-shift segmentation as an example from different branches of the taxonomy that shares similar design stages.

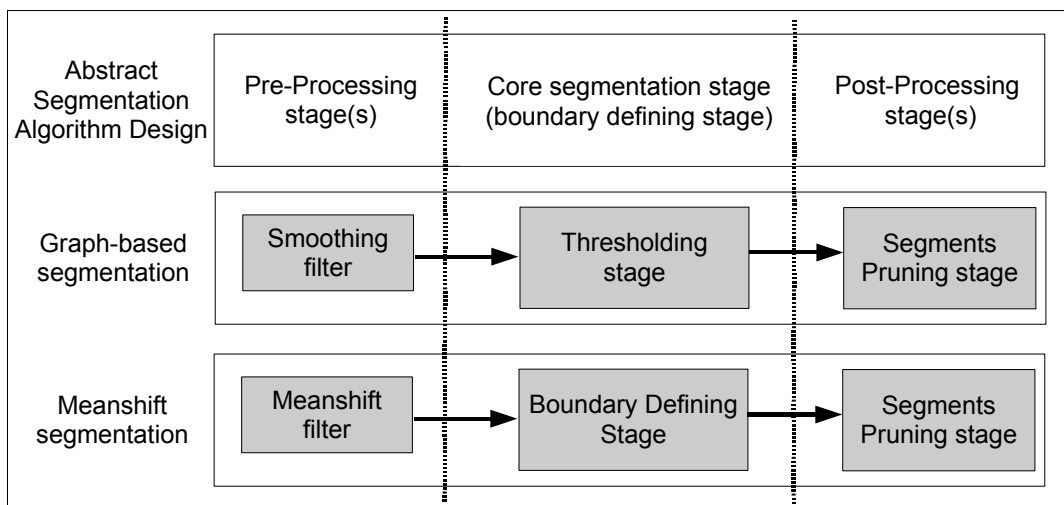


Figure 6.29: The three stages of the segmentation algorithm design with example of the graph-based and the mean-shift segmentation algorithms

Now not all the segmentation algorithm designs correspond clearly with the abstract design illustrated above, and in some cases some designs don't use one of the stages (or don't even use both stages, usually in simpler and older seg-

mentation algorithms). However this design gives a good general impression of the processing components of the segmentation algorithm. As seen in the results of this chapter, the pre-/post-processing stages have a greater effect on the final segmentation quality than the processing power spent on iteration of the boundary defining stage for example. In our research, these stages usually are focused on colour features of the images. However, combining other features to filter the images for the boundary defining stage can prove to be even more useful as found by [Martin \*et al.\* \(2004\)](#).

The second point is influenced by the fact that subjective evaluation is dependent on the hand-segmented images as ground truth. The evaluation tends to optimise the parameters to find solutions that provide segmentation results closer to the hand-segmented images. Hence, the results are focused on main objects in the hand-segmented images. Human subjects, while identifying and segmenting objects in the images (or in the world around them), are affected by both physiological reasons (the deficiencies in the human eye colour perception and other tendencies to not distinguish the slight changes in the colour levels) and psychological reasons (human tendencies to have some preconception and use previously acquired-knowledge to identify objects in their sight frame).

The segmentation algorithms are not affected by a these deficiencies. However, the smoothing filter that provides highly-smoothed results actually is providing these ‘human deficiencies’ and hence the evaluation is trying to optimise on the parameter related to this fact and similarly with other parameters. And even with the objective evaluation, the  $F$  measure in this case is by design and was made to prefer segmentation outputs with lower number of segments (and large segments in the same time), which resembles closely the hand-segmented ground

truth used by the subjective evaluation. As a result, the objective evaluation chooses parameters that correspond closely with the parameter set found with the subjective evaluation. The other measures based on the  $F$  measure, the  $F'$  and the  $Q$ , which categorise as intra-region uniformity evaluation metrics also exhibit bias for under-segmented and less noisy images (hence the significant effect of the smoothing filter on the final results) and this conclusion corresponds closely with the conclusion found by *Zhang et al.* in his extensive survey paper (*Zhang et al., 2008*).

Even for other objective evaluation methods that don't have these biases, *Zhang et al. (2008)* concludes after his extensive survey that all the existing objective evaluations assume that the pixels in the input images follow one homogenous distribution model. They usually use a Gaussian-based distribution model, and are only focused on low-level features of the images and don't include any high-level semantic understanding of the objects in the images. Higher-level semantics can only usually be employed for segmentation algorithms that are aimed at certain applications. Medical cancer cell identification is one example, where a prior medical knowledge of the shape and size of the cancer cell can be incorporated into the evaluation, and also in the segmentation algorithm design.

Another method to incorporate this prior knowledge is by using machine learning techniques. For example, using a continuous evaluation process built into the segmentation process (like the machine learning frameworks mentioned in the introduction) (*Zhang et al., 2005, 2006*). The author thinks that this approach is an interesting approach for future work in this area and will need further study in the coming years. And our findings of the improvement achieved with the GA, polishing stage, and the time-weighted cost functions can be easily implemented

in such frameworks.

In conclusion, the author thinks that evaluating different segmentation algorithms against each other will not give the best way of achieving the goal of the ultimate segmentation algorithm design, but looking at the algorithms' processing stages and how they affect the segmentation results. Furthermore the future approaches to designing better segmentation algorithms should not be based on producing new designs of the segmentation algorithms that are usually slightly better under certain condition, but looking at what processing stages that make the segmentation algorithms and how effective they are in both the segmentation quality and the computation performance. So instead of building the machine learning framework that optimises the segmentation algorithms, we can provide the framework with interchangeable and parameter-varied filtering components that can be combined to produce a suitable algorithms. We can't emphasise more the need to include the computation performance as an evaluation metric which is usually neglected in previous research while only focusing on the segmentation quality.

# 7

## Conclusion

The research for this thesis explored the field of colour image segmentation and the balance required to achieve a good quality segmentation results while still attaining an efficient processing performance. As segmentation is but only one of the stages in any computer vision or image understanding application, a highly efficient and fast computation process is a goal in itself. However, speed itself is of little value if the results are wrong, and as such, the segmentation evaluation field has also progressed. This field provides a wide variety of methods to test the quality of segmentation results and the performance of the segmentation algorithms and their implementations. The author's research focussed on finding the best ways to optimise the current set of segmentation algorithms, especially because there is still no general mathematical model that can be used as a basis for a general-solution to segmentation. The optimisation aims to achieve the best segmentation quality possible with the methods available while still performing this in a computationally efficient manner. The research led us to the impor-

tance of the segmentation parameters used and how they relate to the general image filtering processes that are used across the different categories of image segmentation algorithms.

Along the way, the research led to an efficient implementation of an evaluation set-up for the segmentation algorithms to help with optimising the results. Combining cluster computing and genetic algorithms improves the overall search speed for the best parameters for a number of segmentation algorithms. This framework can be modified and improved to help related research into evaluation in the image-processing field.

## 7.1 Findings

---

The thesis had two focuses: firstly, understanding the image-segmentation process and what affects this process' output segmentation quality and, at the same time, affects its overall performance. To achieve this, the second focus was using and optimising the evaluation framework to help in the same goal. So, for example, adding the GA module optimised the parameter search speed to arrive at an optimal parameter set. In addition, the evaluation framework was enhanced using distributed computing and genetic algorithms. Both areas brought important insights and improved the author's understating of the general requirements and trade-offs faced by a computer vision system designer.

The performance gained by the evaluation framework improvements can help in implementing the evaluation as an online system integrated with the segmentation system itself for continuous testing and adjustment of the segmentation parameters depending on the input images. Those inputs can have high com-

putational requirements, especially if consideration is taken of the fact that the input can be a continuous video stream and not one but multiple streams. The advent of high definition images/videos also makes demands on any processing system.

The first contribution of this thesis is the proposal for a computationally efficient environment for conducting quantitative algorithm testing for image segmentation. Utilising a harness in a scripting language provides a structured route to testing, in such a way that various evaluation metrics can be applied. To improve the speed of evaluation a cluster computer can act as a throughput engine. However, this assumes a full or exhaustive search is made, whereas the search can be optimised by using a genetic algorithm as a convenient tool. The thesis provides guidance on how to apply a genetic algorithm to the problem of finding input parameters to a segmentation algorithm. The given parameters should improve the segmentation according to an objective or cost function.

The second contribution arose from the experimental output of the first contribution: the importance of the parameter settings on the final segmentation quality and the execution performance. Furthermore, this resulted in finding the link between those parameters and the underlying components of the image-processing filters that are used to build the different segmentation algorithms. The process basically is divided into three stages: 1) pre-processing stage; 2) core segmentation stage; and 3) a post-processing stages. Using common image-filtering stages between the different image segmentation algorithms is helpful. Those filters usually are basic filters (like a smoothing filter, colour quantisation, and region pruning). However, they have a significant affect on the segmentation performance and output quality.

---

## **7.2 Reflections**

---

The second contribution is the more important finding in the author's opinion. Research into segmentation algorithms tends to introduce new ideas and implementations in their algorithms' design. Usually these new algorithms only provide slightly better results for certain applications. Furthermore, the research only provides test results on a few images or a very limited set without image type variety, that is the results are application-specific. One of the reasons for the author's belief is that there is no standard framework to test and evaluate the segmentation algorithms, or even a standard set of images for evaluation that different segmentation research can use and compare results. Therefore, providing this facility will help contrast the segmentation results across the field, and advances can be quantified in a better way.

Although this thesis discussed several solutions for image segmentation evaluation frameworks, and image data-sets, and used one of them for testing: the Berkeley Segmentation data-set, and researched several approaches to evaluation (analytical, subjective and objective), each has some shortcomings that have been highlighted previously. To summarise:

The research found that, by design, most algorithms share quite similar processing stages, even across algorithms that belong to different categories in the taxonomy presented. Furthermore, these processing stages are actually related to the segmentation algorithms' input parameters and certain values are quite advantageous for both the segmentation quality and computation performance. For example, using the Berkeley hand-segmented images, then smoothing stages with the high value for smoothing actually provides better results and reduces the



over-segmentation. Hand-segmentation seems very dependent on the recognition of colour in the images and, although the boundary-defining processing without any pre-processing can usually define boundaries in coloured objects, these are usually subtle and ‘missed’ or unrecognised by the humans. Even segmentation evaluation using computer-synthesised images as ground truth can hardly be generalised. (Haralick, 2000; Zhang *et al.*, 2008). Furthermore, for most reference images, there is no guarantee that one generated reference image, either human-segmented or computer-synthesised, is better than another. In other words, reference images are essentially subjective. Thus, evaluation methods using these reference images are considerably subjective. Similarly, both subjective and objective evaluation methods have their own shortcomings. For example, subjective evaluation has a dependence on the external definition of the segments before starting the evaluation, and hand-segmentation can take a long time to complete. For objective evaluation, there is still no one method that can be considered as the best evaluation method.

On the other hand, the recent suggestions and advances in research using machine-learning techniques for both improving and speeding-up the segmentation process are, the author thinks, a step in the right direction. This research shows clear advantages to using GAs to speed-up the evaluation process and to arrive at an optimal answer faster. The framework itself was improved further by using mathematical optimisation in a polishing stage and testing the segmentation quality and computation performance according to their parameter variations. The author can’t emphasise enough the advantage gained using the time-weighted cost functions in the evaluation process, and future research will need to take advantage of this improvement.

### 7.3 Improvements and Further work suggestions

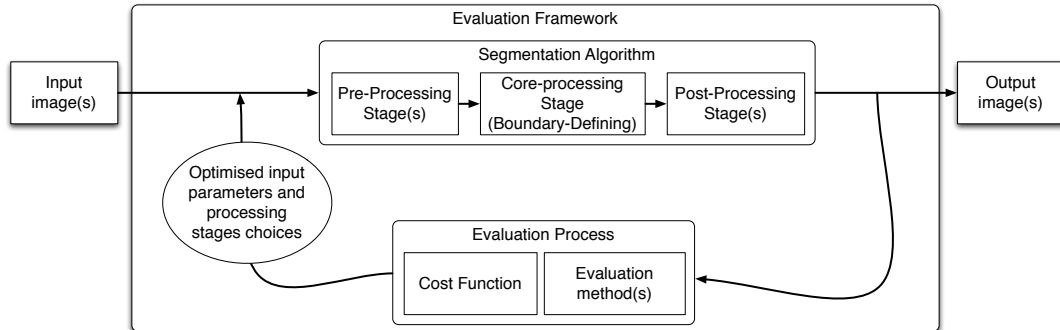


Figure 7.1: An abstract graph of an improved and combined segmentation and evaluation framework

### 7.3 Improvements and Further work suggestions

For future work, the aim is to access these processing stages discussed above, and add an additional factor of variation into a continuous evaluation framework that can optimise the segmentation process as more input images are processed, as suggested recently by [Chabrier \*et al.\* \(2008\)](#); [Zhang \*et al.\* \(2006\)](#). The difference will be, instead of only optimising parameters for the segmentation algorithms as a whole, as the current solution proposes, the pre-/post-processing filters need to be varied too as needed (turning them on/off) while also varying their parameters to optimise the final segmentation results. This we propose will both provide better segmentation results and allow for higher sensitivity for different image segmentation applications.

Figure 7.1 illustrates an abstract graph for a future system of combined segmentation and evaluation framework. The framework should be scripted to run the same test on a set of images automatically. However, the framework in this case will not only vary the parameter automatically for a pre-set of parameters range and step value, as was carried out in this research, but will also adjust and

### 7.3 Improvements and Further work suggestions

---

choose the type and the number of pre-/post-processing stages. The framework can also be improved to automatically run not only one configured evaluation algorithm, but also run multiple of evaluation algorithms, the results of which can be combined in the end to choose the best parameter sets and processing stages to segment the next image in the set. The cost function can also combine a time-weighted factor to take into consideration the computational performance of the segmentation algorithm.

Although this framework will add additional steps to the segmentation process, these steps can be distributed to different processing nodes, either multiple processors or processor cores. So for example at the start of the framework run, a couple of segmentation processes can be started on different nodes and then their results can be sent for evaluation. In this case, each evaluation operation can also take place on a completely different processing node. The evaluation results can be then fed back to optimise the next batch of the segmentation runs, and this cycle can continue completely automated and independent from external input after the first start.



## Publications

### Published:

1. H. Al-Muhairi, M. Fleury, and A. Clark. A computationally efficient evaluation environment for image segmentation. In *International Conference on Machine Vision ICMV07*, pages 129–134, 2007. (Al-Muhairi *et al.*, 2007a)
2. H. Al-Muhairi, M. Fleury, and A. Clark. Computationally efficient quantitative testing of image segmentation with a genetic algorithm. In *IEEE 3rd Int. Conf. on Signal-Image Technology and Internet-based Systems*, 2007. (Al-Muhairi *et al.*, 2007b)
3. H. Almuhairi, M. Fleury, and A. Clark. Time-weighted quantitative testing of image segmentation with a genetic algorithm. In *Signal Processing and Information Technology (ISSPIT), 2009 IEEE International Symposium on*, pages 271 –276, 14-17 2009. (Almuhairi *et al.*, 2009)
4. H. Almuhairi, M. Fleury, and A. F. Clark. Time-weighted evaluation of im-

---

age segmentation with a genetic algorithm. In *5th International Conference on Computer Vision Theory and Applications (VISAPP 2010)*, Angers, France, May 2010. (Almuhairi *et al.*, 2010c)

**Submitted:**

1. H. Almuhairi, M. Fleury, and A. F. Clark. Genetic algorithm-based testing of image segmentation algorithms. In S. Ramakrishnan and I. M. M. El Emary, editors, *Computational Intelligence Techniques in Handling Image Processing and Pattern Recognition*. LAP Lambert Academic Publishing, 2010. (Almuhairi *et al.*, 2010a)
2. H. Almuhairi, M. Fleury, and A. F. Clark. Parameter-orientated segmentation algorithm evaluation. In *The 3rd International Conference on Image and Signal Processing (CISP 2010)*, Yantai, China, 2010. (Almuhairi *et al.*, 2010b)

# B

## Appendix B

### **B.1 Thresholding evaluation**

---

For this experiment, two parameters were exposed from the thresholding algorithm to be varied by the GA evaluation framework. Thresholding algorithm produce a ‘binary’ segmentation of either ‘objects’ and ‘background’ pixels as detailed in Chapter 2. The parameters define the thresholds values for the pixels, where within the range between these two values the pixels are considered as ‘object’ and the rest as ‘background’ segments.

Figure B.1 illustrate the results of the parameters for the Thresholding algorithm. In this case there was no distinguishable trend in the parameters found as solutions between the different images under test or even between the evaluation without and with the time factor. However this not the case with all the rest of the algorithms as will be described below for other algorithms cases.

The other important illustration shows the same images set -represented on

## B.1 Thresholding evaluation

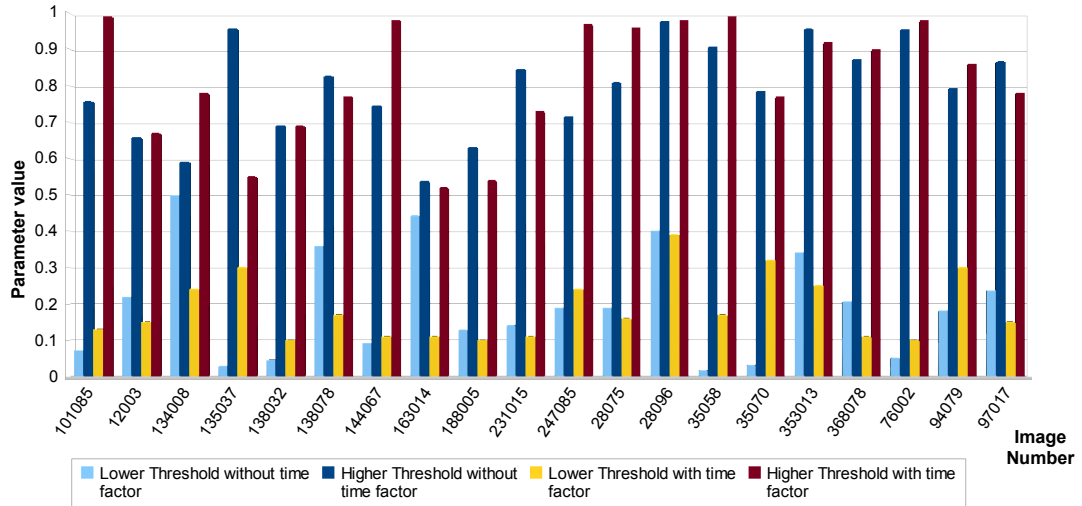


Figure B.1: Thresholding evaluation 20 images with and without time factor parameters

the x-axis like above- however representing the time taken to complete the full evaluation for 20 generations with and without the time factor. The y-axis represent the time taken in minutes.

Figure B.2 shows the timing results for the thresholding algorithm, in the case its clear that the effect of adding the time factor on the overall time taken by the evaluation is to lower the overall time taken compared to the evaluation without the time factor.

In general, for the set of 20 images tested on, the evaluation with the time factor didn't take more than 30 minutes and not less than 10 minutes, as such, there is very small difference between the time taken among all the images.

This not the case without the time factor, the reason is that without the time factor, the GA evaluation have no consideration for the time taken by the parameter combination solution in each population created and in some cases

## B.1 Thresholding evaluation

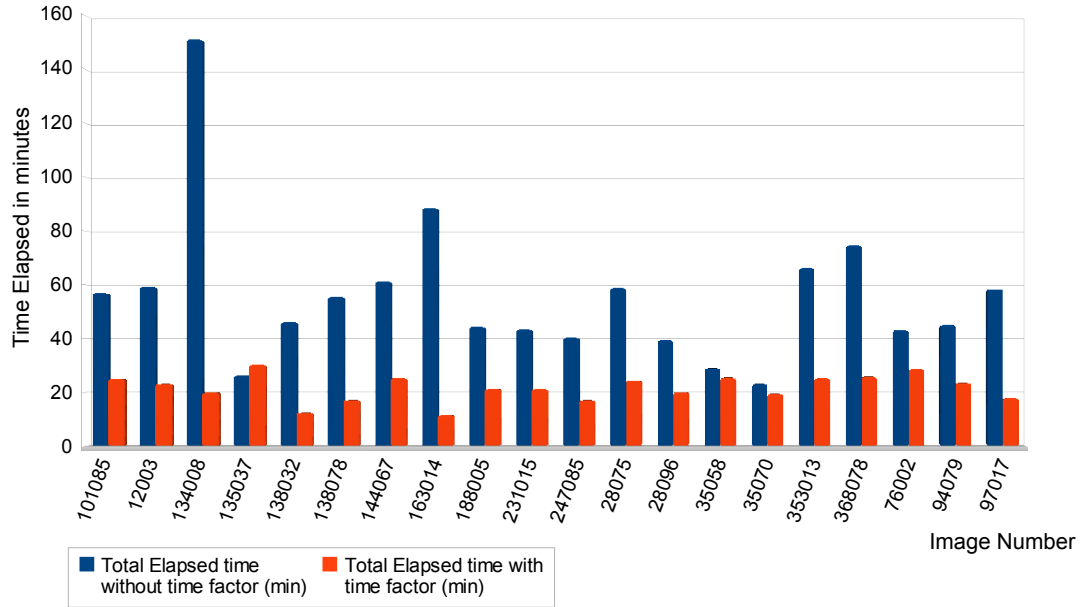


Figure B.2: Thresholding evaluation 20 images with and without time factor timing

can actually spend more time with the parameters sets that are computationally expansive however don't actually provide any comparable improvement on the segmentation accuracy.

And for the same reason there is a big difference between the time taken between different images and even different runs for the same image, in this case for the results illustrated in Figure B.2, between a maximum of 150 minutes or take as minimum as 20 minutes. And as expected the GA evaluation without the time factor don't optimise the results on the time taken basis and as such don't have any preferences for population that are time efficient.

So in this case even when the time factor didn't provide any new insights on the parameters values, where further tweaking of the evaluation setting can provide some useful results, its still the great saving in the time taken to complete



the evaluation alone is a great improvement that we would like to highlight here with these results.

## B.2 Edge Detection evaluation

---

Edge Detection similarly have two parameters for use in the evaluation. In this case, the parameters are lower threshold (also called maximal threshold), which represent the fraction of the automatically computed maximum threshold found in the input image.

And higher threshold (also called minimum threshold) is used to consider if the neighbour pixels to the edges found by the first parameters are also edges if they are higher than the product of the two parameters.

A more detailed illustration with visual segmentation results will be provided in the coming Chapter 4. For this section the focus is on the timing factor effect on the results.

Figure B.3 show the parameters found to be best with the GA evaluation with and without the time factor. In this case, unlike the thresholding, there is a clear ‘best’ parameter set values and there is no difference between the results without the time factor and with the time factor, as such there is no parameter that is time intensive.

Its important to note here that these two parameters are actually related to sub-stage in the edge detection, the non-maxima suppression stage, which is similar to a thresholding stage, and actually can be interchanged with thresholding, however for our testing here we used the non-maxima suppression to test more segmentation sub-processing stages.

## B.2 Edge Detection evaluation

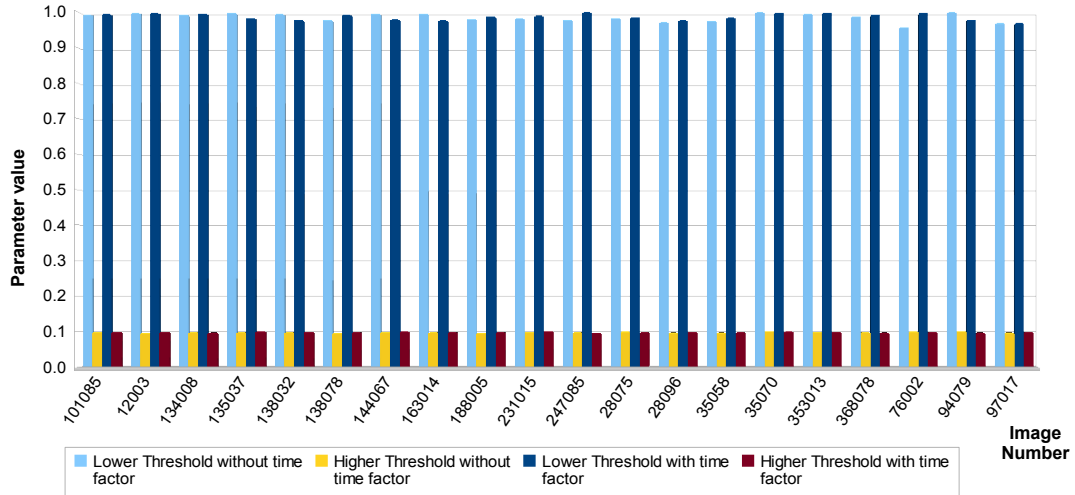


Figure B.3: Edge Detection evaluation 20 images with and without time factor parameters

This stage is actually preceded with an image gradient computation -a smoothing filter- stage. This stage uses Sobel operator in this case however other can also be used (Prewitt operator for example).

The focus here is not to explore all the variation of the edge detection algorithm, other researchers provide more detailed descriptions, however the focus is to highlight how algorithms ‘evolve’, like edge detection depend on ‘thresholding’ stage, however include pre-processing smoothing stage to complete the segmentation process.

Figure B.4 illustrate the timing results and again its clear that the overall evaluation time is reduced by the using the timing factor in the evaluation.

In this case looking at the previous figure too, the evaluation process with the timing factor reduces the timing while still arriving at the same parameters set found by the evaluation without the timing factor.

## B.2 Edge Detection evaluation

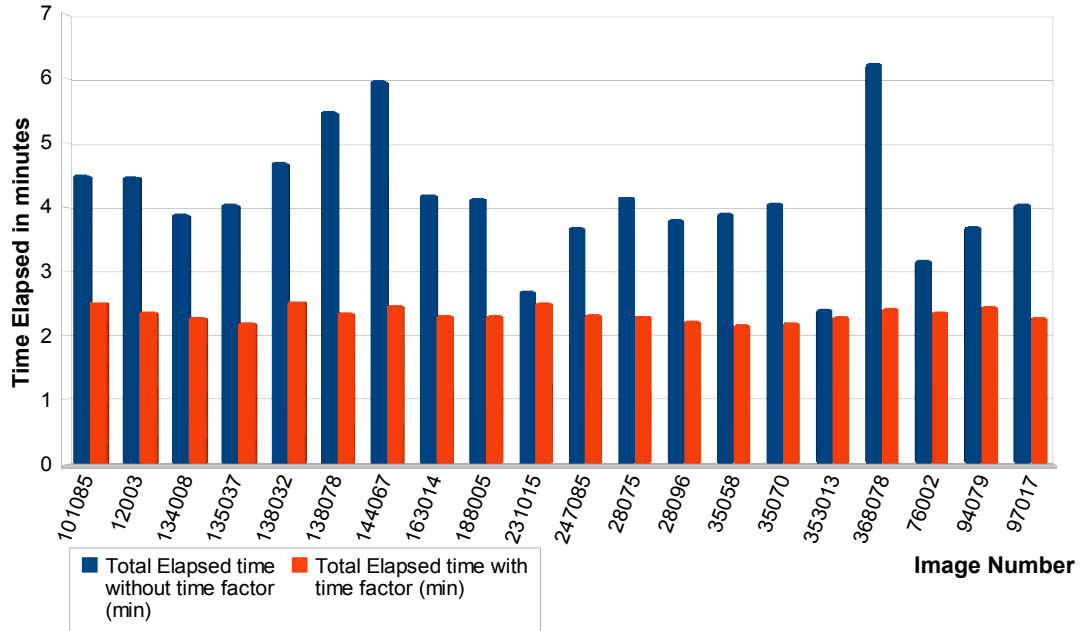


Figure B.4: Edge Detection evaluation 20 images with and without time factor timing

### B.2.1 Canny Edge Detection evaluation

Canny edge detection algorithm adds improvement to the ‘smoothing’ stage, the original authors suggest a more optimal gradient filter, the second stage is a similar non-maxima suppression stage to the section above for consistency which was actually first introduced by Canny in his 1986 paper. Similarly the same two parameters are used.

Figure B.5 shows the parameters, and similar to results of the edge detection algorithm found above, there is a clear trend to the ‘optimal’ parameters set. That is lower threshold parameter equal 1.0 and higher threshold parameter equal 0.1, and there is no difference between the results found with the time factor or without it, and is consistent among all the images under test.

### B.3 Rain-falling Watershed segmentation evaluation

---

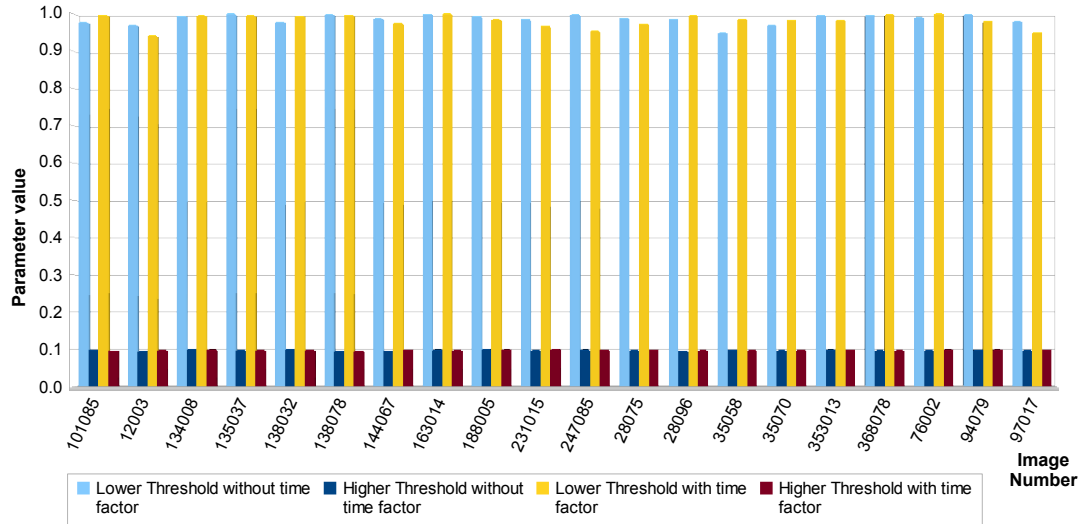


Figure B.5: Canny Edge Detection evaluation 20 images with and without time factor parameters

Similarly there is a similar result on using the time factor on the overall time taken to complete the evaluation, the time factor almost halves the time taken in almost all the 20 image evaluations performed, Figure B.6 illustrate the evaluation timing for the Canny Edge detection with and without the timing factor.

### B.3 Rain-falling Watershed segmentation evaluation

---

### B.4 Anisotropic-based segmentation evaluation

---

## B.4 Anisotropic-based segmentation evaluation

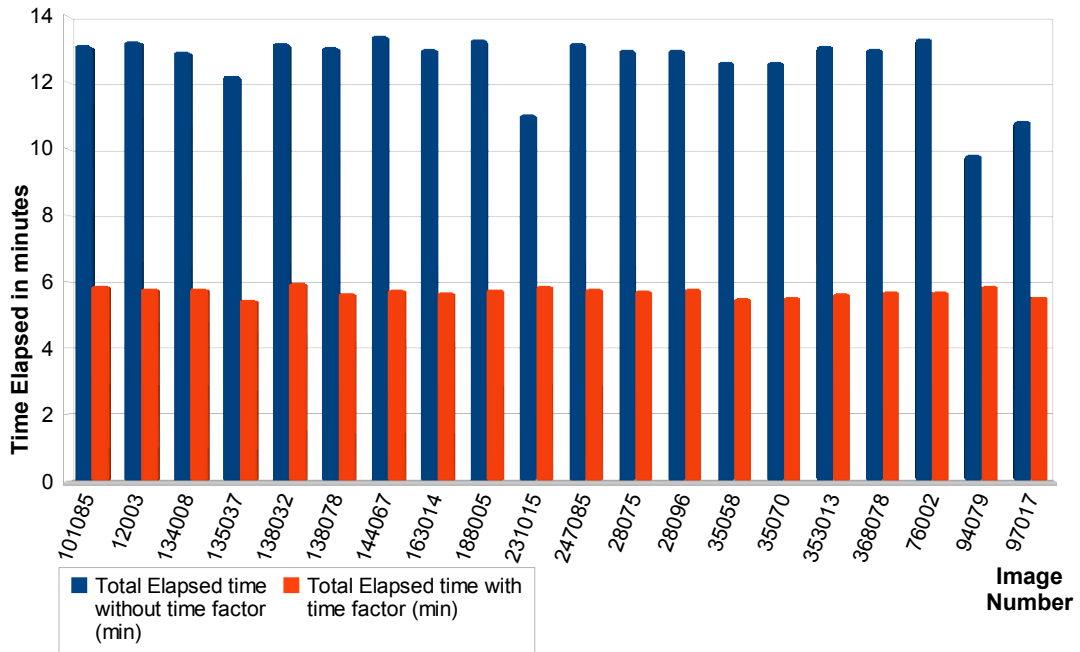


Figure B.6: Canny Edge Detection evaluation 20 images with and without time factor timing

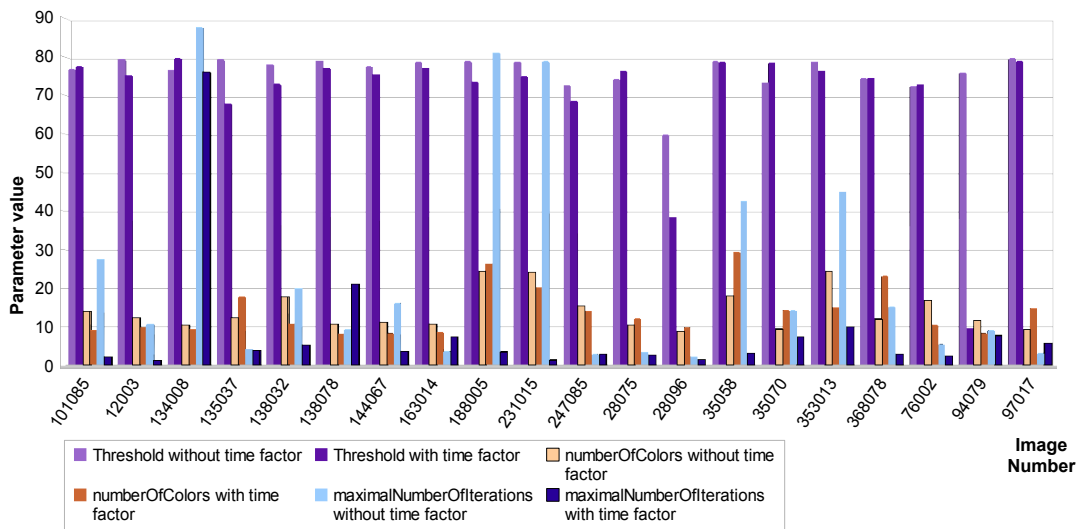


Figure B.7: Rain-falling Watershed segmentation evaluation 20 images with and without time factor parameters

## B.4 Anisotropic-based segmentation evaluation

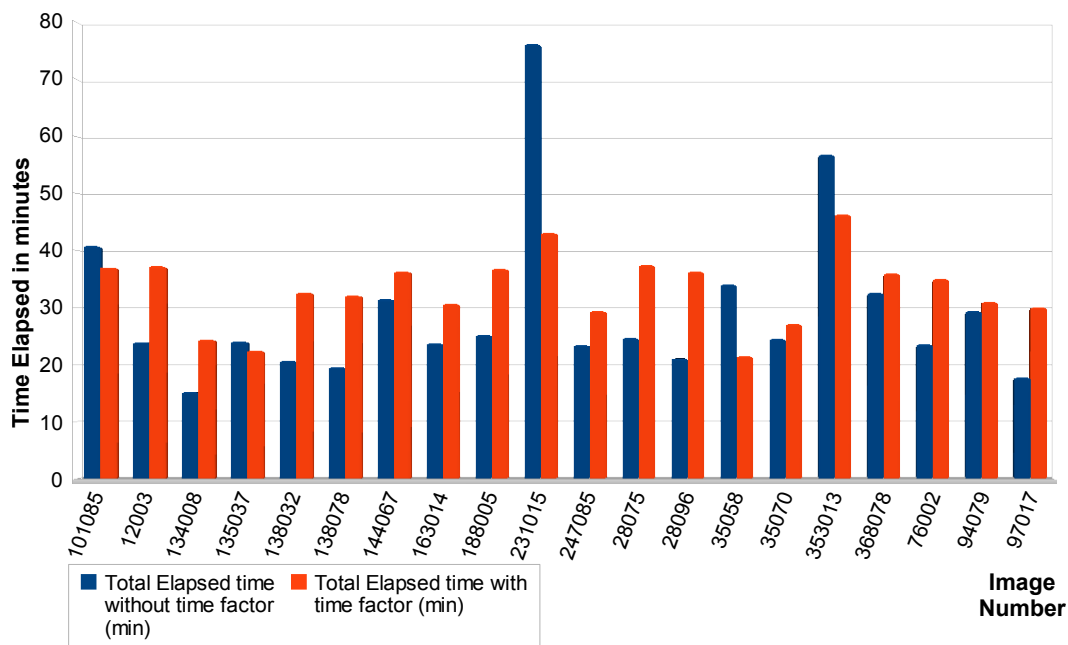


Figure B.8: Rain-falling Watershed segmentation evaluation 20 images with and without time factor timing

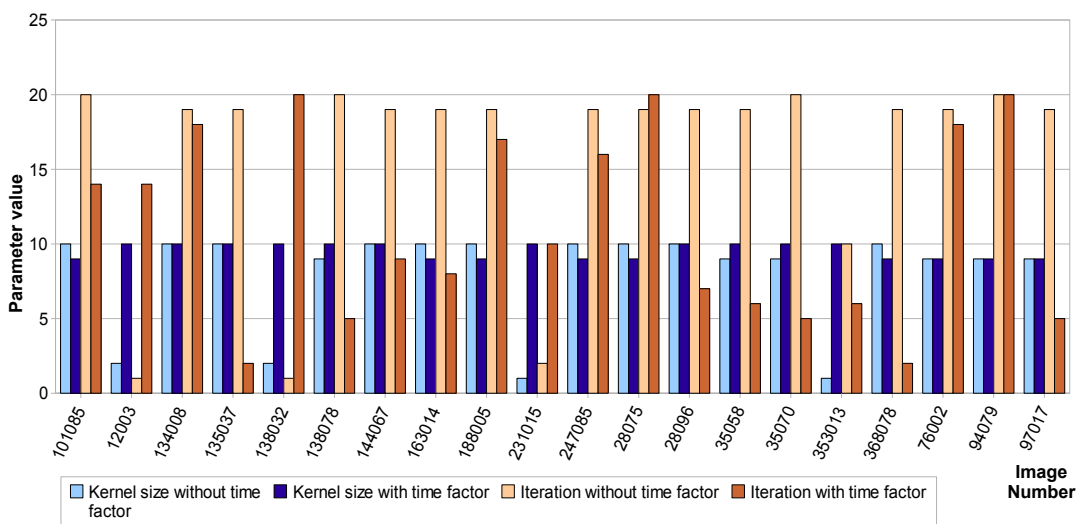


Figure B.9: Anisotropic-based segmentation evaluation 20 images with and without time factor parameters

## B.4 Anisotropic-based segmentation evaluation

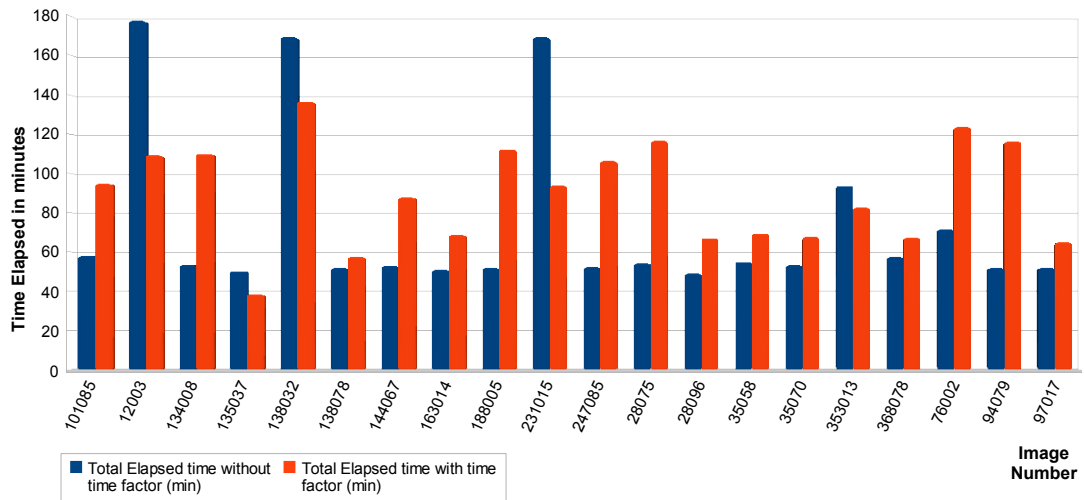


Figure B.10: Anisotropic-based segmentation evaluation 20 images with and without time factor timing

# C

## Appendix C



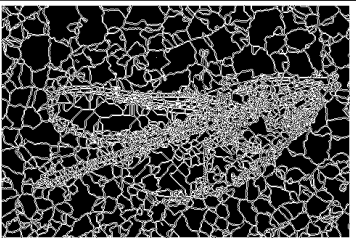
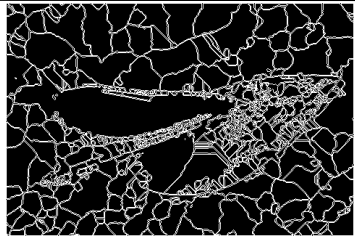
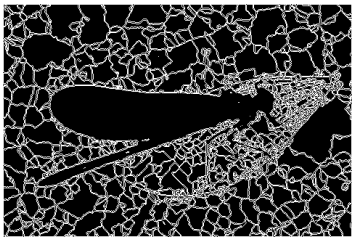
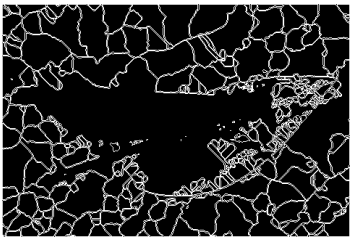
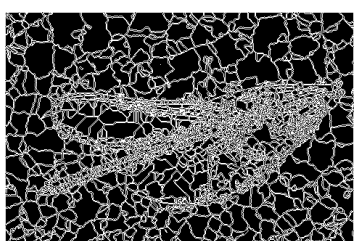

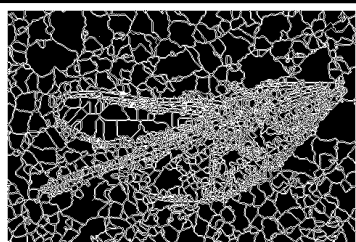
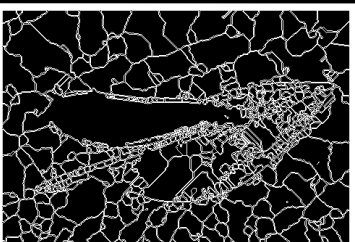
		No. of C		256	16
Th	D	I			
1	0.1	1			
60	0.1	1			
1	1	1			
1	0.1	100			

Figure C.1: Variation of segmentation for the same results in Fig. 4.7 with parameter settings for Watershed segmentation

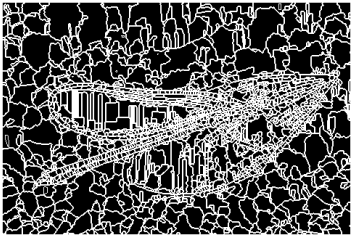

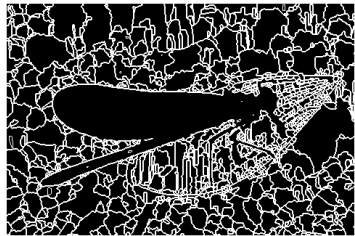

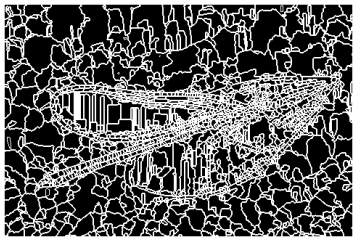

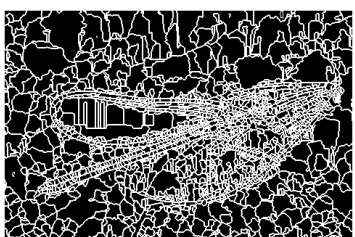
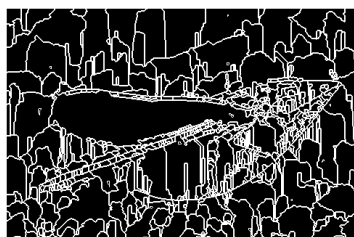
Th	No. of C		256	16
	D	I		
1	0.1	1		
60	0.1	1		
1	1	1		
1	0.1	100		

Figure C.2: Variation of segmentation for the same results in Fig. 4.11 with parameter settings for Rain-falling segmentation

		Merge Thresh.	
		0.0	0.5
Th	Ks		
0.1	1		
0.5	1		
0.1	10		
0.5	10		

Figure C.3: Variation of segmentation for the same results in Fig. 4.15 with parameter settings for Colour Watershed segmentation


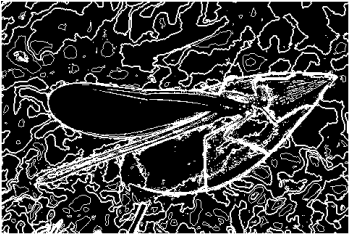
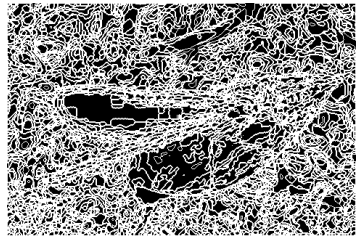
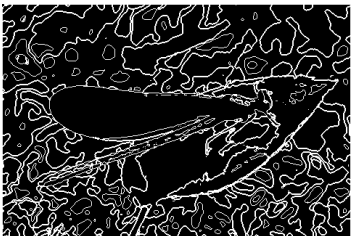
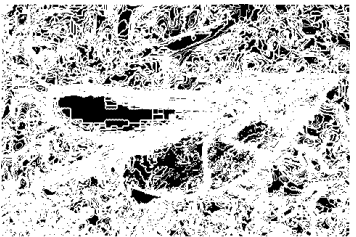
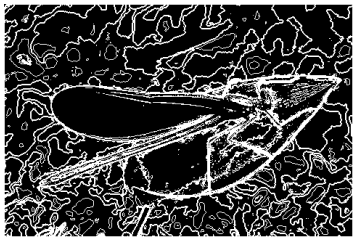


K	No. of C		256	8
	D	I		
1	0.1	1		
5	0.1	1		
1	1	1		
1	0.1	10		

Figure C.4: Variation of segmentations for the same results in Fig. 4.17 with parameter settings for K-mean segmentation

		Colour Distance		1	10
		rdS	rdR		
1	1				
10	1				
1	10				
10	10				

Figure C.5: Variation of segmentations for the same results in Fig. 4.19 with parameter settings for Mean-shift segmentation

# References

- AL-MUHAIRI, H., FLEURY, M. & CLARK, A. (2007a). A computationally efficient evaluation environment for image segmentation. In *International Conference on Machine Vision ICMV07*, 129–134. [7](#), [145](#), [216](#)
- AL-MUHAIRI, H., FLEURY, M. & CLARK, A. (2007b). Computationally efficient quantitative testing of image segmentation with a genetic algorithm. In *IEEE 3rd Int. Conf. on Signal-Image Technology and Internet-based Systems*. [7](#), [145](#), [216](#)
- ALMUHAIRI, H., FLEURY, M. & CLARK, A. (2009). Time-weighted quantitative testing of image segmentation with a genetic algorithm. In *Signal Processing and Information Technology (ISSPIT), 2009 IEEE International Symposium on*, 271–276. [216](#)
- ALMUHAIRI, H., FLEURY, M. & CLARK, A.F. (2010a). Genetic algorithm-based testing of image segmentation algorithms. In S. Ramakrishnan & I.M.M. El Emary, eds., *Computational Intelligence Techniques in Handling Image Processing and Pattern Recognition*, LAP Lambert Academic Publishing. [217](#)
- ALMUHAIRI, H., FLEURY, M. & CLARK, A.F. (2010b). Parameter-orientated segmentation algorithm evaluation. In *The 3rd International Conference on Image and Signal Processing (CISP 2010)*, Yantai, China. [217](#)
- ALMUHAIRI, H., FLEURY, M. & CLARK, A.F. (2010c). Time-weighted evaluation of image segmentation with a genetic algorithm. In *5th International Con-*

- 
- ference on Computer Vision Theory and Applications (VISAPP 2010)*, Angers, France. 217
- ALVARADO, P. (2004). *Segmentation of color images for interactive 3D object retrieval*. Ph.D. thesis, RWTH-Aachen. 29, 64, 94, 182
- BADER, D.A., JÁJÁ, J., HARWOOD, D. & DAVIS, L.S. (1996). Parallel algorithms for image enhancement and segmentation by region growing, with an experimental study. *The Journal of Supercomputing*, **10**, 141–168. 35
- BASU, M. (2002). Gaussian-based edge-detection methods - a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, **32**, 252–260. 55
- BENNETT, K.P. & PARRADO-HERNÁNDEZ, E. (2006). The interplay of optimization and machine learning research. *J. Mach. Learn. Res.*, **7**, 1265–1281. 144
- BEUCHER, S. & LANTUEJOUL, C. (1979). Use of watersheds in contour detection. In *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, Rennes, France..* 28
- BHANU, B., LEE, S. & MING, J. (1989). Adaptive image segmentation using a genetic algorithm. In *Proceedings of a workshop on Image understanding workshop*, 1043–1055, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 143
- BORRÀS, A. & LLADÓS, J. (2009). Modelling human segmentation trough color and space analysis. In *CAIP '09: Proceedings of the 13th International Confer-*

- ence on Computer Analysis of Images and Patterns*, 898–905, Springer-Verlag, Berlin, Heidelberg. [13](#)
- BORSOTTI, M., CAMPADELLI, P. & SCHETTINI, R. (1998). Quantitative evaluation of color image segmentation results. *Pattern Recogn. Lett.*, **19**, 741–747. [196](#)
- BOWYER, K. & PHILLIPS, P.J. (1998a). *Empirical Evaluation Techniques in Computer Vision*. IEEE Computer Society Press, Los Alamitos, CA, USA. [119](#)
- BOWYER, K.W. & PHILLIPS, P.J. (1998b). Overview of work in empirical evaluation of computer vision algorithms. In K.W. Bowyer & P.J. Phillips, eds., *Empirical Evaluation Techniques in Computer Vision*, IEEE Comp Press, CA, USA. [119](#)
- BURJORJEE, K. (2007). Towards a sound theory of adaptation for the simple genetic algorithm. [144](#)
- BUSIN, L., VANDENBROUCKE, N. & MACAIRE, L. (2008). *Color spaces and image segmentation*, vol. 151 of *Advances in Imaging and Electron Physics*, chap. 2, 65–168. Elsevier Inc., Orlando, FL, USA. [20](#)
- CAMPBELL, N.W., THOMAS, B.T. & TROSCIANKO, T. (1996). Neural networks for the segmentation of outdoor images. In *Solving Engineering Problems with Neural Networks. Proceedings of the International Conference on Engineering Applications of Neural Networks (EANN'96)*. Syst. Eng. Assoc, Turku, Finland, vol. 1, 343–6. [13](#)



- 
- CANNY, F.J. (1986). A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, **8**, 679–698. [22](#), [23](#), [55](#)
- CASELLES, V., KIMMEL, R. & SAPIRO, G. (1997). Geodesic active contours. *Int. J. Comput. Vision*, **22**, 61–79. [13](#), [16](#)
- CHABRIER, S., LAURENT, H. & EMILE, B. (2005a). Performance evaluation of image segmentation: application to parameters fitting. In *13th European Signal Processing Conference EURASIP'05*. [112](#), [144](#)
- CHABRIER, S., ROSENBERGER, C. & EMILE, B. (2005b). Segmentation evaluation by fusion with genetic algorithm. In *13th European Signal Processing Conference EURASIP'05*. [112](#), [144](#)
- CHABRIER, S., ROSENBERGER, C., EMILE, B. & LAURENT, H. (2008). Optimization-based image segmentation by genetic algorithms. *EURASIP Journal on Image and Video Processing*, **2008**, 1–10. [112](#), [144](#), [214](#)
- CHAMORRO-MARTÍNEZ, J., SÁNCHEZ, D. & PRADOS-SUAREZ, B. (2003). A fuzzy color image segmentation applied to robot vision. In J. Benitez, O. Cordón, F. Hoffman & R. Roy, eds., *Advances in Soft Computing, Engineering, Design and Manufacturing*, 129–138, Springer. [13](#)
- CHEN, C.W., LUO, J. & PARKER, K.J. (1998). Image segmentation via adaptive k-mean clustering and knowledge-based morphological operations with biomedical applications. *IEEE Transactions on Image Processing*, **7**, 1673–1683. [32](#)

- CHEN, T.W., CHEN, Y.L. & CHIEN, S.Y. (2008). Fast image segmentation based on k-means clustering with histograms in hsv color space. In *Multimedia Signal Processing, 2008 IEEE 10th Workshop on*, 322–325. [13](#)
- CHENG, H.D., JIANG, X., SUN, Y. & WANG, J. (2001). Color image segmentation: advances and prospects. *Pattern Recognition*, **34**, 2259–2281. [12](#), [14](#)
- CHOJNACKI, W., BROOKS, M., VAN DEN HENGEL, A. & GAWLEY, D. (2004). A new constrained parameter estimator for computer vision applications. *Image and Vision Computing*, **22**, 85–91. [144](#)
- CIESIELSKI, K.C. & UDUPA, J.K. (2007). A general theory of image segmentation: level set segmentation in the fuzzy connectedness framework. In J.P.W. Pluim & J.M. Reinhardt, eds., *Medical Imaging 2007: Image Processing*, vol. 6512, 65120W, SPIE. [12](#)
- COHEN, P.R. (1995). *Empirical methods for artificial intelligence*. MIT Press, Cambridge, MA, USA. [123](#)
- COMANICIU, D. & MEER, P. (1997). Robust analysis of feature spaces: color image segmentation. In *Proceedings of 1997 IEEE Conference on Computer Vision and Pattern Recognition*, 750–755. [13](#)
- COMANICIU, D. & MEER, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, **24**, 603–619. [67](#), [69](#), [153](#), [154](#), [189](#)

- COURTNEY, P., THACKER, N. & CLARK, A.F. (1997). Algorithmic modelling for performance evaluation. *Mach. Vision Appl.*, **9**, 219–228. [8](#), [119](#)
- CRAMARIUC, B., GABBOUJ, M. & ASTOLA, J. (1997). Clustering based region growing algorithm for color image segmentation. In *Digital Signal Processing Proceedings, 1997. DSP 97., 1997 13th International Conference on*, vol. 2, 857–860. [12](#)
- CUMANI, A. (1991). Edge detection in multispectral images. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, **53**, 40–51. [57](#)
- DE SMET, P. & PIRES, R.L.V.P.M. (2000). Implementation and analysis of an optimized rainfalling watershed algorithm. vol. 3974, 759–766, SPIE. [29](#)
- DESOUZA, G.N. & KAK, A.C. (2002). Vision for mobile robot navigation: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, **24**, 237–267. [3](#)
- DROOGENBROECK, M.V. & BARNICH, O. (2005). Design of statistical measures for the assessment of image segmentation schemes. In A. Gagalowicz & W. Philips, eds., *CAIP*, vol. 3691 of *Lecture Notes in Computer Science*, 280–287, Springer. [43](#)
- EVERINGHAM, M., MULLER, H. & THOMAS, B. (2001). Evaluating image segmentation algorithms using monotonic hulls in fitness/cost space. In T. Cootes & C. Taylor, eds., *Proceedings of the 12th British Machine Vision Conference (BMVC2001)*, 363–372, BMVA. [117](#), [143](#), [144](#), [166](#)

- EVERINGHAM, M., VAN GOOL, L., WILLIAMS, C.K.I., WINN, J. & ZISSERMAN, A. (2008). The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. <http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html>. 118
- EVERINGHAM, M.R., MULLER, H. & THOMAS, B.T. (2002a). Algorithm evaluation by probabilistic fitness/cost analysis and application to image segmentation. In D. Suter & A. Bab-Hadiashar, eds., *Proceedings of the 5th Asian Conference on Computer Vision (ACCV2002)*, 580–586, Asian Federation of Computer Vision Societies (AFCV). 111, 112, 143, 144
- EVERINGHAM, M.R., MULLER, H. & THOMAS, B.T. (2002b). Evaluating image segmentation algorithms using the pareto front. In A. Heyden, G. Sparr, M. Nielsen & P. Johansen, eds., *Proceedings of the 7th European Conference on Computer Vision (ECCV2002), Part IV (LNCS 2353)*, 34–48, Springer. 112, 143, 144, 166
- FAN, J., HAN, M. & WANG, J. (2009). Single point iterative weighted fuzzy c-means clustering algorithm for remote sensing image segmentation. *Pattern Recognition*, **42**, 2527 – 2540. 2
- FELZENSZWALB, P.F. & HUTTENLOCHER, D. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, **29**, 167–181. 104, 193
- FU, K. & MUI, J. (1981). A survey on image segmentation. *Pattern Recognition*, **13**, 3–16. 6

- GEIST, A., BEGUELIN, A., DONGARRA, J., JIANG, W., MANCHEK, R. & SUNDERAM, V. (1994). *PVM: Parallel virtual machine: a users' guide and tutorial for networked parallel computing*. MIT Press, Cambridge, MA, USA. [36](#)
- GEVERS, T. (2002). Adaptive image segmentation by combining photometric invariant region and edge information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **24**, 848–852. [34](#)
- GEVERS, T. & STOKMAN, H.M.G. (2003a). Classifying color edges in video into shadow-geometry, highlight, or material transitions. *IEEE Transactions on Multimedia*, **5**, 237–243. [24](#), [176](#)
- GEVERS, T. & STOKMAN, H.M.G. (2003b). Robust photometric invariant region detection in multispectral images. *International Journal of Computer Vision*, **53**, 135–151. [34](#), [57](#), [63](#)
- GOLDBERG, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. [150](#), [152](#)
- GOLDMAN, D., YANG, M. & BOURBAKIS, N. (2002). A neural network-based segmentation tool for color images. In *ICTAI '02: Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence*, 500, IEEE Computer Society, Washington, DC, USA. [17](#)
- GONZALEZ, R.C. & WOODS, R.E. (2002). *Digital Image Processing*. Prentice-Hall, Upper Saddle River, N.J., USA, 2nd edn. [3](#), [11](#), [19](#), [23](#), [58](#)

- GUYON, I., MARKHOUL, J., SCHWARTZ, R. & VAPNIK, V. (1998). What size test set gives good error rate estimates? *IEEE Trans. Pattern Anal. Mach. Intell.*, **20**, 52–64. [117](#)
- HALL, D. (2006). Automatic parameter regulation of perceptual systems. *Image and Vision Computing*, **24**, 870–881. [144](#)
- HARALICK, R. (2000). Validating image analysis algorithms. *Keynote Address at SPIE Medical Imaging*, 2–16. [168](#), [213](#)
- HARALICK, R.M. & SHAPIRO, L.G. (1985). Survey: Image segmentation techniques. *Comput. Vision, Graphics, Image Processing*, **29**, 100–132. [6](#), [38](#), [40](#)
- HARIS, K., ESTRADIADIS, S.N., MAGLAVERAS, N. & KATSAGGELOS, A.K. (1998). Hybrid image segmentation using watersheds and fast region merging. *IEEE Transactions on Image Processing*, **7**, 1684–1699. [29](#)
- HOLLAND, J.H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press. [150](#)
- HOOVER, A., JEAN-BAPTISTE, G., GOLDFOF, D.B. & BOWYER, K.W. (1994). A methodology for evaluating range image segmentation techniques. In *Second IEEE Workshop on Applications for Computer Vision*, 264–271. [39](#)
- HOOVER, A., JEAN-BAPTISTE, G., JIANG, X., FLYNN, P.J., BUNKE, H., GOLDFOF, D.B., BOWYER, K.W., EGGERT, D.W., FITZGIBBON, A.W. & FISHER, R.B. (1996). An experimental comparison of range image segmentation algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, **18**, 673–689. [39](#)

- HUANG, H., CHEN, Y. & HSU, W. (2002). Color image segmentation using a self-organizing map algorithm. *Journal of Electronic Imaging*, **11**, 136. [17](#)
- HUANG, Q. & DOM, B. (1995). Quantitative methods of evaluating image segmentation. In *ICIP '95: Proceedings of the 1995 International Conference on Image Processing (Vol. 3)-Volume 3*, 3053, IEEE Computer Society, Washington, DC, USA. [42](#), [127](#)
- IBANEZ, L., SCHROEDER, W., NG, L. & CATES, J. (2005). *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-15-7, <http://www.itk.org/ItkSoftwareGuide.pdf>, 2nd edn. [116](#)
- ITO, N., SHIMAZU, Y., YOKOYAMA, T. & MATUSHITA, Y. (1995). Fuzzy logic based non-parametric color image segmentation with optional block processing. In *CSC '95: Proceedings of the 1995 ACM 23rd annual conference on Computer science*, 119–126, ACM, New York, NY, USA. [17](#)
- JAIN, A.K., MURTY, M.N. & FLYNN, P.J. (1999). Data clustering: a review. *ACM Computing Surveys*, **31**, 264–323. [31](#)
- KASS, M., WITKIN, A. & TERZOPOULOS, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, **1**, 321–331. [13](#), [16](#)
- KLINKER, G.J., SHAFER, S.A. & KANADE, T. (1987). Using a color reflection model to separate highlights from object color. In *Proc. First International Conference on Computer Vision (ICCV)*, 145–150. [34](#)

- KLINKER, G.J., SHAFER, S.A. & KANADE, T. (1988). Color image analysis with an intrinsic reflection model. In *Proceedings of the Second International Conference on Computer Vision (ICCV)*, 292–296. [34](#)
- KLINKER, G.J., SHAFER, S.A. & KANADE, T. (1990). A physical approach to color image understanding. *International Journal of Computer Vision (IJCV)*, **4**, 7–38. [13](#), [34](#)
- KOSCHAN, A. (1995). A comparative study on color edge detection. In *2nd Asian Conf. on Computer Vision ACCV'95*, 574–578. [22](#), [24](#)
- KOSCHAN, A. & ABIDI, M. (2005). Detection and classification of edges in color images. *Signal Processing Magazine, IEEE*, **22**, 64 – 73. [23](#)
- KÖTHER, U. (1995). Primary image segmentation. In G. Sagerer, S. Posch & F. Kummert, eds., *DAGM-Symposium*, Informatik Aktuell, 554–561, Springer. [14](#), [25](#)
- KRAVTSCHENKO, V. & LITTLE, J. (1999). Efficient color object segmentation using the dichromatic reflection model. *Communications, Computers and Signal Processing, 1999 IEEE Pacific Rim Conference on*, 90–94. [34](#)
- KUBASSOVA, O., BOYLE, R.D. & RADJENOVIC, A. (2006). Evaluation of colour image segmentation results. In *Proceedings of the Joint Disease Workshop, 9th International Conference on Medical Image Computing and Computer Assisted Intervention*, vol. 1, 72–79. [43](#)



- KURGÖLLÜS, F. & SANKUR, B. (1999). Image segmentation based on multi-scan constraint satisfaction neural network. *Pattern Recognition Letters*, **20**, 1553–1563. [11](#), [12](#)
- KUROKAWA, H., KANEKO, S. & YONEKAWA, M. (2009). A color image segmentation using inhibitory connected pulse coupled neural network. *Advances in Neuro-Information Processing: 15th International Conference, ICONIP 2008, Auckland, New Zealand, November 25-28, 2008, Revised Selected Papers, Part II*, 776–783. [13](#)
- KURUGOLLU, F., SANKUR, B. & HARMANCL, A.E. (2001). Color image segmentation using histogram multithresholding and fusion. *Image Vision Comput.*, **19**, 915–928. [13](#)
- LAVALLE, S.M. & HUTCHINSON, S.A. (1995). A framework for constructing probability distributions on the space of image segmentations. *Comput. Vis. Image Underst.*, **61**, 203–230. [12](#)
- LEVINE, M.D. & NAZIF, A.M. (1985). Dynamic measurement of computer generated image segmentations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **PAMI-7**, 155–164. [41](#)
- LIM, Y.W. & LEE, S.U. (1990). On the color image segmentation algorithm based on the thresholding and the fuzzy c-means techniques. *Pattern Recogn.*, **23**, 935–952. [17](#)
- LIU, J. & YANG, Y.H. (1994). Multiresolution color image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, **16**, 689–700. [196](#)

- LIU, S., QIAO, Y.Y. & WEN, Q.K. (2009). Segmentation of multispectral remote sensing images based on ant colony optimization algorithm. In *GEC '09: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, 891–894, ACM, New York, NY, USA. [2](#)
- LUCCHESI, L. & MITRA, S.K. (1999). Unsupervised segmentation of color images based on k-means clustering in the chromaticity plane. In *CBAIVL '99: Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, 74, IEEE Computer Society, Washington, DC, USA. [31](#), [34](#)
- LUCCHESI, L. & MITRA, S.K. (2001a). Color image segmentation: A state-of-the-art survey. In *Proc. of the Indian National Science Academy (INSA-A)*, vol. 67, 207–221, Indian National Science Academy. [2](#), [6](#), [23](#)
- LUCCHESI, L. & MITRA, S.K. (2001b). Color segmentation through independent anisotropic diffusion of complex chromaticity and lightness. In *Proc. of 2001 Int'l Conference on Image Processing (ICIP 2001)*, vol. 1, 746–749. [34](#)
- LUGER, G.F. (2004). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison Wesley, 5th edn. [32](#)
- MA, W.Y. & MANJUNATH, B.S. (1997). Edge flow: A framework of boundary detection and image segmentation. In *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, 744–749, IEEE Computer Society, Washington, DC, USA. [12](#)
- MACQUEEN, J. (1967). Some methods for classification and analysis of multivariate observations. In L.M. Le Cam & J. Neyman, eds., *Proceedings of the*

- 
- fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, 281–297, University of California Press, Berkeley, CA, USA. [66](#)
- MAEDA, J., KAWANO, A., SAGA, S. & SUZUKI, Y. (2007). Unsupervised perceptual segmentation of natural color images using fuzzy-based hierarchical algorithm. In B.K. Ersbøll & K.S. Pedersen, eds., *SCIA*, vol. 4522 of *Lecture Notes in Computer Science*, 462–471, Springer. [13](#), [17](#)
- MALAMAS, E.N., PETRAKIS, E.G.M., ZERVAKIS, M., PETIT, L. & LEGAT, J.D. (2003). A survey on industrial vision systems, applications and tools. *Image and Vision Computing*, **21**, 171–188. [2](#)
- MARROQUIN, J.L. & GIROSI, F. (1993a). Some extensions of the k-means algorithm for image segmentation and pattern classification. Tech. rep., Massachusetts Institute of Technology, Cambridge, MA, USA. [31](#)
- MARROQUIN, J.L. & GIROSI, F. (1993b). Some extensions of the k-means algorithm for image segmentation and pattern classification. Tech. rep., Cambridge, MA, USA. [66](#)
- MARTIN, D.R., FOWLKES, C., TAL, D. & MALIK, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision*, 416–425. [43](#), [117](#), [118](#), [127](#), [146](#), [147](#)
- MARTIN, D.R., FOWLKES, C.C. & MALIK, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, **26**, 530–549. [117](#), [123](#), [206](#)

- MAXWELL, B. & SHAFER, S. (1997). Physics-based segmentation of complex objects using multiple hypotheses of image formation. *Computer vision and image understanding(Print)*, **65**, 269–295. [17](#)
- MCINERNEY, T. & TERZOPOULOS, D. (1996). Deformable models in medical image analysis: A survey. *Medical Image Analysis*, **1**, 91–108. [27](#)
- MEIJSTER, A. & ROERDINK, J.B.T.M. (1995). A proposal for the implementation of a parallel watershed algorithm. In *CAIP '95: Proceedings of the 6th International Conference on Computer Analysis of Images and Patterns*, 790–795, Springer-Verlag, London, UK. [28](#), [35](#)
- MELKEMI, K.E., BATOCHE, M. & FOUFOU, S. (2006). A multiagent system approach for image segmentation using genetic algorithms and extremal optimization heuristics. *Pattern Recogn. Lett.*, **27**, 1230–1238. [144](#)
- MOGA, A.N. & GABBOUJ, M. (1997). Parallel image component labeling with watershed transformation. *IEEE Trans. Pattern Anal. Mach. Intell.*, **19**, 441–450. [29](#), [36](#)
- MOGA, A.N., CRAMARIUC, B. & GABBOUJ, M. (1995). A parallel watershed algorithm based on rainfaling simulation. In *Proc. 12th European Conf. Circuit Theory and Design*, 339–342. [29](#), [36](#)
- MÜHLENBEIN, H. & SCHLIERKAMP-VOOSEN, D. (1993). Predictive models for the breeder genetic algorithm i. continuous parameter optimization. *Evolutionary Computation*, **1**, 25–49. [152](#)

- NELDER, J.A. & MEAD, R. (1965). A simplex method for function minimization. *Computer Journal*, **7**, 308–313. [162](#)
- NETHERCOTE, N. & SEWARD, J. (2007). Valgrind: a framework for heavyweight dynamic binary instrumentation. *SIGPLAN Not.*, **42**, 89–100. [49](#)
- NICOLESCU, C., ALBERS, B. & JONKER, P. (1999). Parallel watershed algorithm on images from cranial ct-scans using pvm and mpi on distributed memory system. In J. Dongarra, E. Luque & T. Margalef, eds., *PVM/MPI*, vol. 1697 of *Lecture Notes in Computer Science*, 418–425, Springer. [35](#), [36](#)
- NIXON, M. & AGUADO, A.S. (2008). *Feature Extraction & Image Processing, Second Edition*. Academic Press, 2nd edn. [55](#)
- OLAGUE, G. (2007). Evolutionary computer vision. In *GECCO '07: Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, 3458–3507, ACM, New York, NY, USA. [144](#)
- OTSU, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, **9**, 62–66. [51](#)
- OUSTERHOUT, J.K. (1998). Scripting: higher level programming for the 21st century. *Computer*, **31**, 23–30. [120](#)
- OUYANG, C.S., CHOU, C.T., JHAN, C.F. & HUANG, J.Y. (2009). An improved approach for image segmentation based on color and local homogeneity features. In *ICASSP '09: Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 1225–1228, IEEE Computer Society, Washington, DC, USA. [12](#)

- PAPPAS, T. (1992). An adaptive clustering algorithm for image segmentation. *IEEE Transactions on Signal Processing*, **40**, 901–914. [32](#)
- PARAGIOS, N. & DERICHE, R. (1999). Geodesic active regions for supervised texture segmentation. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, 926, IEEE Computer Society, Washington, DC, USA. [3](#)
- PARK, M.H., PARK, R.H. & LEE, S.W. (2005). Efficient shot boundary detection for action movies using blockwise motion-based features. In *Advances in Visual Computing*, vol. 3804, 478–485. [53](#)
- PHAM, D.L., XU, C. & PRINCE, J.L. (2000). A survey of current methods in medical image segmentation. In *Annual Review of Biomedical Engineering*, vol. 2, 315–338. [2](#)
- PHAM, N.A., MORRISON, A., SCHWOCK, J., AVIEL-RONEN, S., IAKOVLEV, V., TSAO, M.S., HO, J. & HEDLEY, D. (2007). Quantitative image analysis of immunohistochemical stains using a cmyk color model. *Diagnostic Pathology*, **2**, 8. [20](#)
- PIGNALBERI, G., CUCCHIARA, R., CINQUE, L. & LEVIALDI, S. (2003). Tuning range image segmentation by genetic algorithm. *EURASIP J. Appl. Signal Process.*, **2003**, 780–790. [144](#)
- POLAK, M., ZHANG, H. & PI, M. (2008). An evaluation metric for image segmentation of multiple objects. *Image and Vision Computing*, **In Press**, **Corrected Proof**, -. [43](#)

- POLHEIM, H. (2005). *GEATbxSurvey: Evolutionary Algorithm Toolbox for MATLAB, version 3.7*. Online at <http://www.geatbx.com/docu/index.html>. 152
- PRATT, W.K. (2001). *Digital Image Processing: PIKS Inside*. John Wiley & Sons, Inc., New York, NY, USA. 116
- PUN, T., GERIG, G. & RATIB, O. (1994). Image analysis and computer vision in medicine. *Computerized Medical Imaging and Graphics*, **18**, 85–96, (Special Issue: Multimedia Techniques in the Medical Environment). 2
- REDDI, S., RUDIN, S. & KESHAVAN, H. (1984). Optimal multiple threshold scheme for image segmentation. *IEEE TRANS. SYST. MAN CYBER.*, **14**, 661–665. 22
- REHRMANN, V. & PRIESE, L. (1998). Fast and robust segmentation of natural color scenes. In *ACCV '98: Proceedings of the Third Asian Conference on Computer Vision-Volume I*, 598–606, Springer-Verlag, London, UK. 12
- REKIK, A., ZRIBI, M., HAMIDA, A.B. & BENJELLOUN, M. (2007). Review of satellite image segmentation for an optimal fusion system based on the edge and region approaches. *International Journal of Computer Science and Network Security (IJCSNS)*, **7**, 242–250. 3
- RIDLER, T. & CALVARD, S. (1978). Picture thresholding using an iterative selection method. *Systems, Man and Cybernetics, IEEE Transactions on*, **8**, 630–632. 21
- RIPLEY, B.D. & HJORT, N.L. (1995). *Pattern Recognition and Neural Networks*. Cambridge University Press, New York, NY, USA. 32

- ROBERTS, L.G. (1963). *Machine perception of three-dimensional solids*. PhD in Electrical engineering, Dept. of Electrical Engineering, Massachusetts Institute of Technology, US. [22](#)
- ROERDINK, J.B.T.M. & MEIJSTER, A. (2000). The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informaticae*, **41**, 187–228. [60](#)
- ROTEM, O., GREENSPAN, H. & GOLDBERGER, J. (2007). Combining region and edge cues for image segmentation in a probabilistic gaussian mixture framework. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1–8. [12](#)
- RUSSELL, S.J. & NORVIG, P. (2002). *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall. [32](#)
- S. LEVACHKINE, J.S. (2000). Image segmentation as an optimization problem. *Computation and Systems*, **3**, 245–263. [144](#)
- SABER, E., TEKALP, A.M. & BOZDAGI, G. (1996). Fusion of color and edge information for improved segmentation and edge linking. In *ICASSP '96: Proceedings of the Acoustics, Speech, and Signal Processing, 1996. on Conference Proceedings., 1996 IEEE International Conference*, 2176–2179, IEEE Computer Society, Washington, DC, USA. [12](#)
- SAHOO, P.K., SOLTANI, S., WONG, A.K. & CHEN, Y.C. (1988). A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, **41**, 233–260. [6](#), [18](#), [22](#), [38](#), [41](#)



- SALEMBIER, P. & GARRIDO, L. (1998). Binary partition tree as an efficient representation for filtering, segmentation and information retrieval. *Image Processing, International Conference on*, **2**, 252. [205](#)
- SALEMBIER, P. & GARRIDO, L. (2000). Binary partition tree as an efficient representation for filtering, segmentation and information retrieval. *IEEE Transactions on Image Processing*, **9**, 561–576. [113](#), [205](#)
- SAPIRO, G. & RINGACH, D.L. (1996). Anisotropic diffusion of multivalued images with applications to color filtering. *Image Processing, IEEE Transactions on*, **5**, 1582–1586. [73](#)
- SCHOENEMANN, T. & CREMERS, D. (2007a). Globally optimal image segmentation with an elastic shape prior. In *IEEE International Conference on Computer Vision (ICCV)*, Rio de Janeiro, Brazil. [144](#)
- SCHOENEMANN, T. & CREMERS, D. (2007b). Introducing curvature into globally optimal image segmentation: Minimum ratio cycles on product graphs. In *IEEE International Conference on Computer Vision (ICCV)*, Rio de Janeiro, Brazil. [144](#)
- SETHIAN, J.A. (1999). *Level Set Methods and Fast Marching Methods*. Cambridge Monograph on Applied and Computational Mathematics, Cambridge University Press, Cambridge, U.K. [13](#), [14](#), [16](#), [25](#), [27](#)
- SEZGIN, M. & SANKUR, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, **13**, 146–165. [19](#), [20](#), [53](#)

- SHAFFER, S.A. (1985). Using color to separate reflection components. **10**, 210–218. [34](#)
- SHEN, H.L., ZHANG, H.G., SHAO, S.J. & XIN, J.H. (2008). Chromaticity-based separation of reflection components in a single image. *Pattern Recognition*, **41**, 2461 – 2469. [34](#)
- SHOTTON, J., WINN, J., ROTHER, C. & CRIMINISI, A. (2006). Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision*, 1–15. [118](#)
- SINGH, A., TERZOPOULOS, D. & GOLDBOF, D.B. (1998). *Deformable Models in Medical Image Analysis*. IEEE Computer Society Press, Los Alamitos, CA, USA. [13](#)
- SNIR, M., OTTO, S.W., WALKER, D.W., DONGARRA, J. & HUSSELEDERMAN, S. (1995). *MPI: The Complete Reference*. MIT Press, Cambridge, MA, USA. [36](#)
- SONKA, M., HLAVAC, V. & BOYLE, R. (1993). *Image Processing, Analysis, and Machine Vision*. Thomson-Engineering. [21](#), [51](#)
- SUMENGEN, B. & MANJUNATH, B.S. (2005). Edgeflow-driven variational image segmentation: Theory and performance evaluation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. [73](#)
- TSAI, Y.H.R. & OSHER, S. (2003). Level set methods in image science. In *International Conference on Image Processing (ICIP)*, vol. 2, 631–634. [16](#), [27](#)

- VEREVKA, O.A. & BUCHANAN, J.W. (1995). Local k-means algorithm for color image quantization. In *Graphics/Vision Interface '95*, Quebec Canada. [66](#)
- VINCENT, L. & SOILLE, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell.*, **13**, 583–598. [16](#), [28](#), [87](#), [88](#)
- WANG, H. & SUTER, D. (2003). Color image segmentation using global information and local homogeneity. In C. Sun, H. Talbot, S. Ourselin & T. Adriaansen, eds., *Proc. VIIth Digital Image Computing: Techniques and Applications*. [11](#), [12](#), [14](#)
- WANG, J., THIESSON, B., XU, Y. & COHEN, M. (2004). Image and video segmentation by anisotropic kernel mean shift. In *Computer Vision - ECCV 2004*, 238–249, Springer. [69](#)
- WELCH, B.B. & HOBBS, J. (2003). *Practical Programming in Tcl & Tk*. Prentice Hall Professional Technical Reference. [120](#)
- WESOLKOWSKI, S., TOMINAGA, S. & DONY, R.D. (2000). Shading- and highlight-invariant color image segmentation using the mpc algorithm. vol. 4300, 229–240, SPIE. [34](#)
- WESZKA, J. & ROSENFELD, A. (1978). Threshold evaluation techniques. *IEEE Trans. Systems, Man and Cybernetics*, **8**, 622–629. [38](#)
- WESZKA, J.S. (1978). A survey of threshold selection techniques. *Computer Vision, Graphics, and Image Processing*, **7**, 259–265. [6](#), [18](#), [38](#), [41](#)

- WITHEY, D. & KOLES, Z. (2007). Medical image segmentation: Methods and software. In *Noninvasive Functional Source Imaging of the Brain and Heart and the International Conference on Functional Biomedical Imaging, 2007. NFSI-ICFBI 2007. Joint Meeting of the 6th International Symposium on*, 140–143. [2](#)
- XU, C., PHAM, D.L. & PRINCE, J.L. (2000). Image segmentation using deformable models. In M. Sonka & J.M. Fitzpatrick, eds., *Handbook of Medical Imaging*, vol. 2, 129–174, SPIE Press. [13](#), [27](#)
- YANG, C.K. & TSAI, W.H. (1996). Reduction of color space dimensionality by moment-preserving thresholding and its application for edge detection in color images. *Pattern Recogn. Lett.*, **17**, 481–490. [20](#)
- YANG, L., ALBREGTSEN, F., LØNNESTAD, T. & GRØTTUM, P. (1995). A supervised approach to the evaluation of image segmentation methods. In *CAIP '95: Proceedings of the 6th International Conference on Computer Analysis of Images and Patterns*, 759–765, Springer-Verlag, London, UK. [38](#), [39](#), [40](#)
- YANG, Q. & KANG, W. (2009). General research on image segmentation algorithms. *I.J. Image, Graphics and Signal Processing*, **1**, 1–8. [12](#), [14](#)
- YASNOFF, W.A., MUI, J.K. & BACUS, J.W. (1977). Error measures for scene segmentation. *Pattern Recognition*, **9**, 217–231. [117](#)
- YEO, N., LEE, K., VENKATESH, Y. & ONG, S. (2005). Colour image segmentation using the self-organizing map and adaptive resonance theory. *Image and Vision Computing*, **23**, 1060–1079. [17](#)

- ZHANG, C. & WANG, P. (2000). A new method of color image segmentation based on intensity and hue clustering. In *ICPR '00: Proceedings of the International Conference on Pattern Recognition*, vol. 3, 613 – 616, IEEE Computer Society, Los Alamitos, CA, USA. [13](#)
- ZHANG, H., FRITTS, J.E. & GOLDMAN, S.A. (2003). An entropy-based objective evaluation method for image segmentation. vol. 5307, 38–49, SPIE. [42](#), [126](#)
- ZHANG, H., FRITTS, J.E. & GOLDMAN, S.A. (2005). A co-evaluation framework for improving segmentation evaluation. In I. Kadar, ed., *Signal Processing, Sensor Fusion, and Target Recognition XIV. Edited by Kadar, Ivan. Proceedings of the SPIE, Volume 5809, pp. 420-430 (2005).*, vol. 5809 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, 420–430. [112](#), [144](#), [207](#)
- ZHANG, H., CHOLLETI, S., GOLDMAN, S.A. & FRITTS, J.E. (2006). Meta-evaluation of image segmentation using machine learning. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1138–1145, IEEE Computer Society, Washington, DC, USA. [112](#), [144](#), [207](#), [214](#)
- ZHANG, H., FRITTS, J. & GOLDMAN, S. (2008). Image segmentation evaluation: A survey of unsupervised methods. *Computer Vision and Image Understanding*, **110**, 260–280. [6](#), [43](#), [48](#), [168](#), [195](#), [200](#), [207](#), [213](#)
- ZHANG, Y.J. (1996). A survey on evaluation methods for image segmentation. *Pattern Recognition*, **29**, 1335–1346. [6](#), [38](#), [41](#), [48](#), [126](#)

## REFERENCES

---

ZHANG, Y.J. (1997). Evaluation and comparison of different segmentation algorithms. *Pattern Recognition Letters*, **18**, 963–974. [38](#)

ZHANG, Y.J. (2001). A review of recent evaluation methods for image segmentation. In *Signal Processing and its Applications, Sixth International Symposium on*, vol. 1, 148–151. [42](#)