



**Cátia Cristina Arranca Queimadelas**

Licenciatura em Ciências da Engenharia Biomédica

## **Automated segmentation, tracking and evaluation of bacteria in microscopy images**

Dissertação para obtenção do Grau de Mestre em  
Engenharia Biomédica

Orientador : José Fonseca, Prof. Auxiliar, FCT/UNL

Co-orientadores : André Mora, Prof. Auxiliar, FCT/UNL  
André Ribeiro, Prof. Assistente, TUT, Finlândia

Júri:

Presidente: Prof. Doutor Mário António de Basto Forjaz Secca

Arguente: Prof. Doutor Pedro Miguel Dinis de Almeida

Vogais: Prof. Doutor José Manuel Fonseca

Prof. Doutor André Damas Mora



FACULDADE DE  
CIÊNCIAS E TECNOLOGIA  
UNIVERSIDADE NOVA DE LISBOA

**Novembro, 2012**



## **Automated segmentation, tracking and evaluation of bacteria in microscopy images**

Copyright © Cátia Cristina Arranca Queimadelas, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa

A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objectivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.



*"You can think I'm wrong,  
but that's no reason to quit thinking."*

Gregory House



# Acknowledgements

First of all I want to thank my supervisors Prof. Dr. José Manuel Fonseca, Prof. Dr. André Damas Mora and Prof. Dr. André Ribeiro for all the support and sincere sympathy along these months of work.

I would like to thank too to Antti Häkkinen for the support and fast answers for all my questions.

To Teresa Neves and Filipa Ferreira is difficult to find words to thank for all you have done for me. We are "the same person".

I want to thank to Rodolfo Abreu, Ricardo Alves and Ana Cartaxo for making me laugh like no one. Thank you for all the fellowship specially in these last two years. I really don't know what I would have done without you!

To Mafalda Correia, Margarida Félix, Sofia Dias, Francisco Ferreira, Joana Vasconcelos and Diana Meixedo I thank for making these five years unforgettable.

A special thank to my parents, for always trusting me and for giving me the opportunity to be here today. To my brother and niece, a big thank for being who you are.

Last but not least I would like to thank to Mário Pinto for all the patience and good advices. You are just perfect.





# Abstract

---

Most of the investigation in microbiology relies on microscope imaging and needs to be complemented with reliable methods of computer assisted image processing, in order to avoid manual analysis.

In this work, a method to assist the study of the *in vivo* kinetics of protein expression from *Escherichia coli* cells was developed. Confocal fluorescence microscopy (CFM) and Differential Interference Contrast (DIC) microscopy images were acquired and processed using the developed method. This method comprises two steps: the first one is focused on the cells detection using DIC images. The latter aligns both DIC and CFM images and computes the fluorescence level emitted by each cell.

For the first step, the Gradient Path Labelling (GPL) algorithm was used which produces a moderate over-segmented DIC image. The proposed algorithm, based on decision trees generated by the Classification and Regression Trees (CART) algorithm, discards the background regions and merges the regions belonging to the same cell.

To align DIC/fluorescence images an exhaustive search of the relative position and scale parameters that maximizes the fluorescence inside the cells is made. After the cells have been located on the CFM images, the fluorescence emitted by each cell is evaluated.

The discard classifier performed with an error rate of  $1.81\% \pm 0.98\%$  and the merge classifier with  $3.25\% \pm 1.37\%$ . The segmentation algorithm detected  $93.71\% \pm 2.06\%$  of the cells in the tested images. The tracking algorithm correctly followed  $64.52\% \pm 16.02\%$  of cells and the alignment method successfully aligned all the tested images.

**Keywords:** Segmentation, Alignment, GPL, Classifier, CART.

---



# Resumo

---

A investigação na área da microbiologia assenta em imagens de microscopia e, de modo a evitar análises manuais, necessita de ser complementada com métodos fiáveis de processamento de imagem assistida por computador.

Neste trabalho foi desenvolvido um método para assistir o estudo da cinética da expressão proteica *in vivo* de células de *Escherichia coli*, a partir de imagens de microscopia confocal de fluorescência (MCF), complementadas com imagens de microscopia de contraste de interferência diferencial (CID). Este método é composto por dois passos principais, o primeiro dos quais consiste na segmentação das imagens de CID de forma a detetar as células e o segundo no alinhamento das imagens CID/MCF e posterior quantificação da fluorescência emitida por cada célula.

Na primeira etapa foi usado o algoritmo Gradient Path Labelling (GPL) que produz uma moderada sobre-segmentação da imagem CID. O algoritmo proposto, baseado em árvores de decisão geradas pelo algoritmo Classification and Regression Trees (CART), descarta as regiões do fundo da imagem e une as regiões pertencentes a cada célula.

Para alinhar as imagens CID/fluorescência é feita uma pesquisa exaustiva pelos parâmetros de posição e escala relativas que maximizam a fluorescência dentro das células. Uma vez localizadas as células na imagem de fluorescência é possível avaliar a fluorescência emitida por cada uma.

O classificador de descarte mostrou atuar com uma taxa de erro de  $1.81\% \pm 0.98\%$  e o classificador de união com  $3.25\% \pm 1.37\%$ . O algoritmo de segmentação detetou  $93.71\% \pm 2.06\%$  das células presentes nas imagens testadas. O algoritmo de *tracking* seguiu correctamente  $64.52\% \pm 16.02\%$  das células e o método usado para o alinhamento alinou correctamente todas as imagens testadas.

**Palavras-chave:** Segmentação, Alinhamento, GPL, Classificador, CART.

---



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Machine Learning</b>	<b>7</b>
2.1	Concepts, Instances and Attributes . . . . .	8
2.2	Types of Machine Learning . . . . .	9
2.2.1	Supervised Learning . . . . .	10
2.3	Output Representations . . . . .	10
2.3.1	Decision Trees . . . . .	12
2.3.2	Classification rules . . . . .	12
2.4	Training and Testing . . . . .	13
2.4.1	Error rate and performance . . . . .	13
2.4.2	Cost . . . . .	15
2.5	Decision Trees Induction . . . . .	16
2.5.1	C4.5 . . . . .	16
2.5.1.1	Construct Decision Trees . . . . .	16
2.5.1.2	Pruning . . . . .	18
2.5.2	CART . . . . .	20
2.5.2.1	Construct the maximum tree . . . . .	20
2.5.2.2	Pruning . . . . .	21
2.5.2.3	Select the best tree . . . . .	22
<b>3</b>	<b>Image Processing</b>	<b>23</b>
3.1	Digital Image Histogram . . . . .	23
3.2	Image interpolation . . . . .	24
3.2.1	Nearest-neighbor . . . . .	26
3.2.2	Bilinear interpolation . . . . .	26
3.2.3	Bicubic interpolation . . . . .	27
3.3	Image Segmentation . . . . .	28
3.3.1	Thresholding . . . . .	28

3.3.2	Region Growing . . . . .	29
3.3.3	Point, line and edge-Based Segmentation . . . . .	30
3.3.4	Gradient Path Labelling (GPL) . . . . .	30
<b>4</b>	<b>Methodology</b>	<b>35</b>
4.1	Cell Segmentation . . . . .	35
4.1.1	GPL . . . . .	35
4.1.2	Classifiers Building . . . . .	38
4.1.2.1	Discard Classifier . . . . .	38
4.1.2.2	Merge Classifier . . . . .	39
4.2	Errors Correction . . . . .	42
4.2.1	Discard Errors . . . . .	42
4.2.2	Merge Errors . . . . .	44
4.2.3	Inverted Images . . . . .	47
4.3	Cells Tracking . . . . .	47
4.4	Images Alignment . . . . .	48
4.4.1	Automatic Alignment . . . . .	51
4.4.2	Semi-Automatic Alignment . . . . .	52
4.5	Fluorescence Quantification . . . . .	53
4.5.1	Poles Determination . . . . .	53
4.6	Interface . . . . .	54
<b>5</b>	<b>Results and Discussion</b>	<b>59</b>
5.1	Results . . . . .	59
5.2	Results Discussion . . . . .	62
<b>6</b>	<b>Conclusion</b>	<b>63</b>
6.1	Future Work . . . . .	64
<b>7</b>	<b>Appendix 1</b>	<b>71</b>

# List of Figures

1.1	DIC and CFM images and the respective relative position. . . . .	3
1.2	Developed algorithm schematics. . . . .	4
2.1	Decision tree for the mushroom data. . . . .	12
2.2	Cross-Validation method illustration. . . . .	14
2.3	Possibilities to the root node test, for the weather data tree. . . . .	17
2.4	Decision tree for the weather data. . . . .	19
2.5	Hypothetical decision trees. . . . .	19
3.1	Histogram interpretation. . . . .	24
3.2	Histogram equalization results example. . . . .	25
3.3	Interpolation illustration. . . . .	26
3.4	Bilinear interpolation example. . . . .	27
3.5	Interpolation methods example. . . . .	28
3.6	Otsu thresholding appliance example. . . . .	29
3.7	Watershed appliance example. . . . .	30
3.8	Edge-based segmentation (Sobel) appliance example. . . . .	31
3.9	An example showing the procedures of the objects detection algorithm. . . . .	32
3.10	GPL Merging algorithm. . . . .	32
3.11	The GPL algorithm step-by-step. . . . .	34
4.1	Different image channel and respective histograms. . . . .	36
4.2	Example of the GPL algorithm processing a DIC image. . . . .	36
4.3	An example of the GPL output. . . . .	37
4.4	GPL resulting in a over-segmented image. . . . .	37
4.5	Variables from discard classifier importance values, given by CART. . . . .	39
4.6	Discard trees set suggested by CART to the discard classifier. . . . .	39
4.7	Discard tree used to build the discard classifier. . . . .	40
4.8	Over-segmented image without background segments. . . . .	41

4.9	The given importance values by CART to the variables from the merge classifier. . . . .	42
4.10	Trees set suggested by CART to build the merge classifier. . . . .	42
4.11	Merge tree used to build the discard classifier. . . . .	43
4.12	A discard type 1 error. . . . .	43
4.13	A discard type 1 error solved. . . . .	44
4.14	A discard type 2 error. . . . .	44
4.15	A discard type 2 error solved. . . . .	45
4.16	A merge type 1 error. . . . .	45
4.17	A merge type 1 error solved. . . . .	45
4.18	A merge type 2 error. . . . .	46
4.19	A merge type 2 error solved. . . . .	46
4.20	Illustration of the used cell division criteria. . . . .	46
4.21	Inverted time series image. . . . .	47
4.22	GPL results of an inverted time series image. . . . .	47
4.23	Cell tracking example. . . . .	49
4.24	Otsu threshold (a) and median filter (b) appliance to the fluorescence image. . . . .	50
4.25	Example of the alignment between the DIC (the one bounded by the red square) and the correspondent CFM images. . . . .	50
4.26	Limits of exhaustive-search parameters. . . . .	52
4.27	An example of the DIC image aligned with the correspondent fluorescence image. . . . .	53
4.28	Cell positive and negative pole location. . . . .	53
4.29	Fluorescence quantification along the major axis of cells, in negative-positive pole direction. . . . .	54
4.30	Fluorescence quantification process. . . . .	55
4.31	Fluorescence quantification algorithm scheme. . . . .	56
4.32	Developed interface. . . . .	57
4.33	Developed interface showing a waiting bar while cells detection is running. . . . .	58



# List of Tables

2.1	Iris Data. . . . .	9
2.2	Weather Data. . . . .	10
2.3	Iris Problem as a Clustering Problem. . . . .	11
2.4	Mushroom quality problem data. . . . .	11
2.5	An example of the cost matrix for binary classification by cost-sensitive learning. . . . .	16
5.1	Time series n.1 error rates . . . . .	60
5.2	Timeseries n.2 error rates . . . . .	60
5.3	Time series n.3 error rates . . . . .	60
5.4	Time series n.4 error rates . . . . .	61
5.5	Time series n.5 error rates . . . . .	61



# Acronyms and Symbols

<i>E.coli</i>	Escherichia coli	<b>FP</b>	False Positive
<b>BR</b>	Bottom-Right coordinates	<b>MD</b>	Number of the decisions made by the merge classifier
<b>CART</b>	Classification and Regression Trees	<b>MER</b>	Merge classifier error rate
<b>DD</b>	Number of decisions made by the discard classifier	<b>MWD</b>	Number of the wrong decisions made by the merge classifier
<b>DER</b>	Discard classifier error rate	<b>PR</b>	Pattern Recognition
<b>DIC</b>	Differential Interference Contrast	<b>RNC</b>	real number of cells
<b>dic<sub>x</sub></b>	DIC image width	<b>SER</b>	Segmentation algorithm error rate
<b>dic<sub>y</sub></b>	DIC image height	<b>SNC</b>	Segmentation number of cells
<b>DWD</b>	Number of wrong decisions made by the discard classifier	<b>TER</b>	Tracking algorithm error rate
<b>fl<sub>x</sub></b>	Fluorescence image width	<b>TL</b>	Top-Left coordinates
<b>fl<sub>y</sub></b>	Fluorescence image height	<b>TN</b>	True Negative
<b>FN</b>	False Negative	<b>TP</b>	True Positive





# Introduction

Most of the recent studies using microbes as model organisms rely on microscope imaging. To improve and accelerate those studies, they need to be complemented with reliable and fast methods of computer assisted image processing. These methods aim at facilitating the extraction of information from images of bacterial populations with single cell resolution, by avoiding manual analysis, which is fastidious, time consuming and subject to observer variances[1, 2].

In this work it was developed a method to assist the study of the *in vivo* kinetics of protein expression, one protein at a time, from confocal fluorescence microscopy (CFM) images of Escherichia coli (E. coli) cells (for growth and induction conditions see [3]).

In confocal fluorescence microscopy, the specimen is generally illuminated by a laser. The term *excitation* rather than *illumination* is more appropriated in what follows, since it more explicitly refers to the contrast-generating process: the excitation of fluorophores (small molecules with fluorescence properties[4]), through absorption, causing detectable fluorescence. After the absorption, the fluorophores start to fluoresce, emitting a light with a longer wavelength than the one emitted from the excitation process. A fraction of the emitted fluorescence is collected by the microscope objective and imaged onto the detector[5].

This work was strongly motivated by the fact that CFM images only show the emitted fluorescence, while cells boundaries are not visible, which hinders the evaluation of the fluorescence emitted by each cell. Hence, cell biologists usually combine fluorescence microscopy with functional images and other types of microscopy to acquire anatomical information allowing to track the fluorescence signals position[6].

This way, a Differential Interference Contrast (DIC) image is simultaneously acquired along with the CFM image, in which cells are well visible. The main of this approach is to

superimpose and adjust both images to allow the evaluation of the emitted fluorescence by each cell.

DIC microscopy renders good contrast for optically transparent biological samples without the need of introducing any exogenous contrast agents. Due to this non-invasive nature, DIC microscopes are widely used in biology laboratories [7, 8]. DIC microscopy makes use of differences in the refractive index and uses two beams of light recombined into one. Because of slight differences in refractive index of the substances each beam passed through, the combined beams are not totally in phase but instead create an interference effect. This effect intensifies even little differences in cell structure[9]. To identify the cells on DIC image, image analysis techniques are essential.

Biomedical cell image analysis is one of the main application fields of computerized image analysis. Visual interpretation is, however, tedious and in many cases, error-prone. Therefore, ever since the first appearance of computers, significant development efforts have been aiming at supplementing or replacing human visual inspection with computer analysis[10].

In order to extract information from microscopy images, those images are segmented. Image segmentation is the process of dividing an image into its constituents objects, or parts, and a background. It is one of the most important and most difficult steps in an image analysis task [11, 10].

Due to the difficulty of image segmentation, many segmentation techniques have been developed by researchers worldwide. Many of those techniques require an interaction with the user and don't produce a satisfactory result on the more difficult types of cell and tissue images. The problems arise when the cells are clustered, the image background varies or when there are intensity variations within the cells[10].

In this work, a two steps segmentation method is proposed. The first step consists on applying the Gradient Path Labelling (GPL) algorithm which produces a slightly over-segmented image. In the second step, machine learning techniques are used in order to overcome that over-segmentation. Two different classifiers were constructed, one that decides which regions of the over-segmented image are to be discarded (as background or air bubbles regions) and the other classifier that decides which regions are to be merged (regions belonging to a same cell).

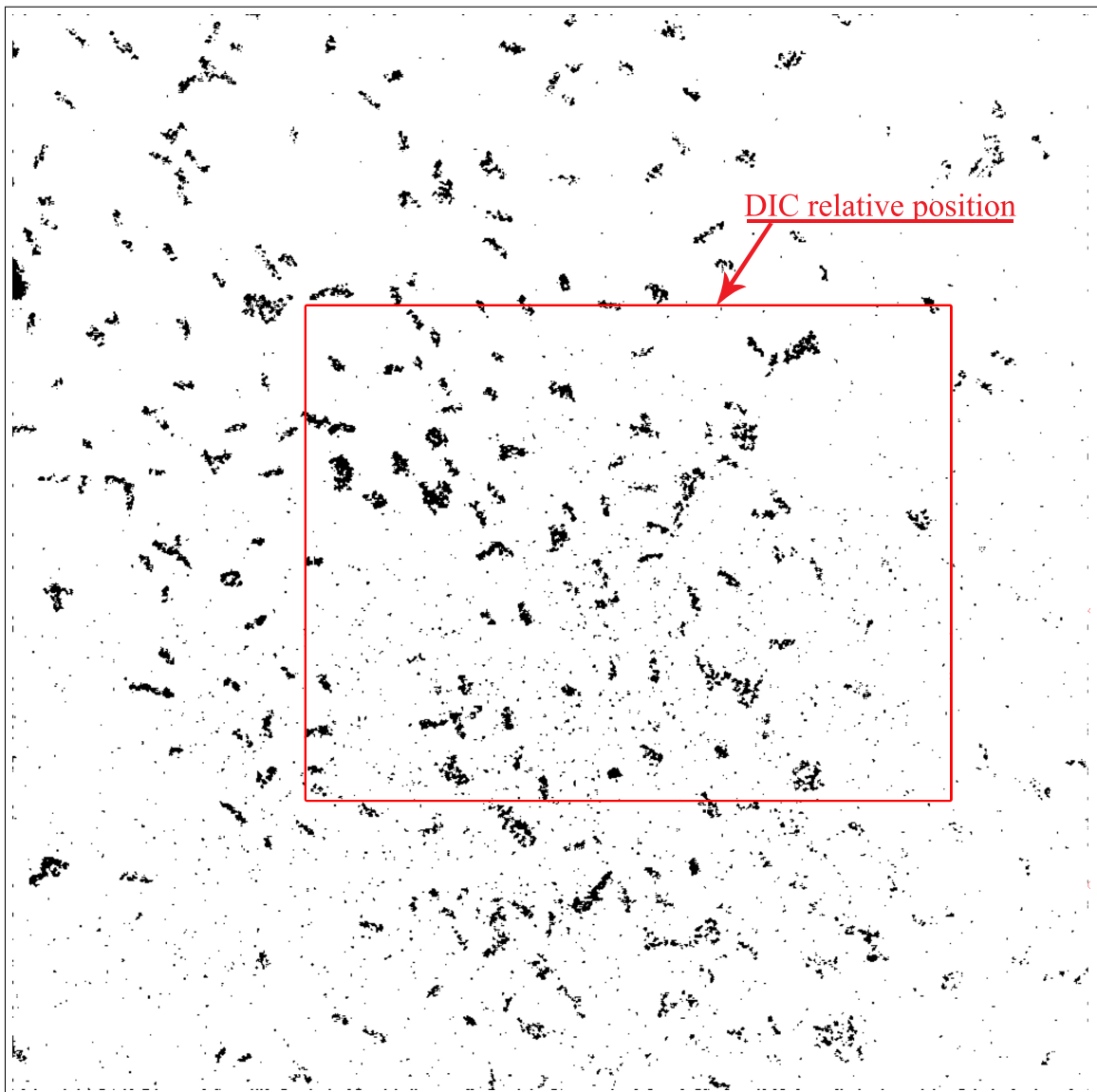
Once the cells present on the DIC images are correctly identified, it is necessary to align those images with the correspondents CFM images. This alignment is not trivial because images differ not only in position but also in scale, as shown in Figure 1.1. In this work a simple alignment technique based on an *exhaustive-search* of position and scale that maximize the fluorescence within cells contours is proposed. For a better visualization of the fluorescence on CFM images an Otsu threshold was applied. Thus, all fluorescence images shown in this dissertation are not literal reproductions of the original ones.

A block diagram of the proposed method is shown on Figure 1.2.

Repeating this process for a set of images acquired over time, it is possible to follow



(a) DIC image.



(b) CFM image with correspondent DIC position marked in red.

**Figure 1.1:** DIC and CFM images and the respective relative position.

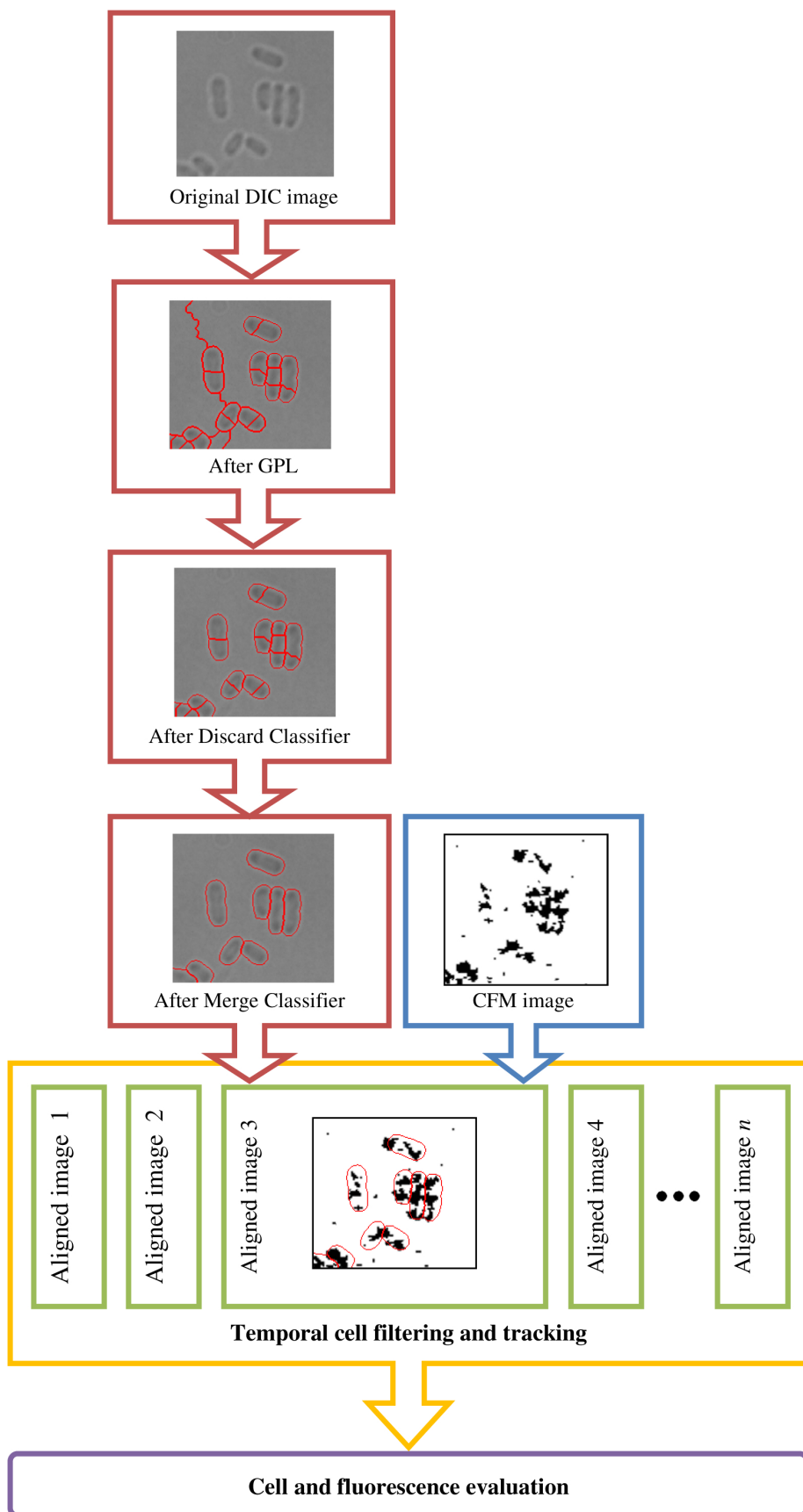


Figure 1.2: Developed algorithm schematics.



the evolution of the cells as well as evaluate the distribution of fluorescence within each cell over time.

## Brief state of the art

There are many references in literature to cell segmentation techniques. Most of those techniques use seeded region growing techniques, usually seeded watershed, as described in [12, 13, 14, 10, 2]. Usually seeds are obtained by thresholding. There are a vast literature referring to this technique, and some more examples can be found in [15, 16, 17].

Snakes or active contour models introduced by Kass *et al.*[18] are often used to segment cells, as in [19] or [20], but have the initialization and minimization drawbacks.

In order to overcome the over-segmentation of images, some techniques as the ones described in [12] can be applied. These techniques merge regions by combining a segment with a smaller size bigger than  $S$ , then it is merged with its smaller neighbour just if the shape factor together are smaller than its alone. A. Duarte *et al.*[21] developed a region merging technique based on the similarity of regions intensities, spacial locations and original sizes. J. Ning *et al.*[22] described a method to merge similar objects but the intervention of the user is needed.

Machine learning techniques are broadly used in cell segmentation methods. However, most of the times they are used to classify cells types [23] or to identify cells cycles phases, as in [24].

M. Tscherepanow *et al* [25] described a method to classify regions in "cells" and "non-cells". However, their method require an image with the background identified, distinguishing only cells from dirt or from cells clusters. Moreover, 107 features were extracted from the data for the classification, leading to a high computational cost process.

The segmentation techniques here mentioned will be described in Section 3.3.

This work was developed with the collaboration of the Laboratory of Biosystems Dynamics from Tampere University of Technology, Finland, which proposed this theme and gently provided all the used images.

This dissertation is organized into five more chapters. The next chapter introduces some of the most important aspects of machine learning, as well as the comparison between two of the most popular machine learning algorithms. The third chapter covers some of the image processing concepts used in this work, including the GPL algorithm. In the fourth chapter all the steps of the developed methodology to construct the proposed algorithm, the segmentation and the alignment phases are presented. Finally, chapters 5 and 6 describe the results and the conclusions of this work, respectively.

This research work also led to an article which was accepted for presentation at the Medical Signal & Information Processing (MEDSIP) conference in Liverpool, United Kingdom. This article can be found in Appendix 1.





# Machine Learning

The ability of learning is one of the main demonstrations of animal intelligence and since the emergence of computers researchers wonder if they are also able to learn[26, 27]. Meanwhile, computers have been used mainly as tools for data processing, but as the processing speed and the amount of data that can be processed by a computer have increased, researchers have discovered that computers can, in fact, be used as more intelligent information processing tools[28].

*Machine learning* is a term used for algorithms and techniques that "teach" a computer. It is a growing field of artificial intelligence and has been used in fields as diverse as business, agriculture, medicine or politics. This technology is based essentially in finding patterns/tendencies in data.

People have been seeking patterns in data ever since human life began. Hunters seek patterns in animal migration behaviour, farmers seek patterns in crop growth and politicians seek patterns in voter opinion with the main goal of predict future situations[29]. We do it almost all the time, and without conscious effort. In fact, most of our day-to-day activities are based on our success in performing various pattern recognition (PR) tasks, as when we read a book, we recognize the letters and words[30].

As the world grows in complexity, overwhelming us with the data it generates, an intelligent analysis of that data is a valuable resource that can lead to new insights, and, in commercial settings, to competitive advantages[29].

In a human sense, learn can be defined as "to get knowledge of something by study, experience or being taught"; "to become aware by information or from observation"; "to commit to memory"; "to be informed of or to ascertain" or "to receive instruction".

In informatics, learning or the acquisition of *knowledge* takes a slightly different meaning. In this context, it can be said that a computer program learns from experience  $E$  with

respect to some class of tasks  $T$  and performance measure  $P$ , if its performance at tasks  $T$ , as measured by  $P$ , improves with training experience  $E$ .

An example is handwriting recognition learning problem:

**Task** Recognition and classifying handwritten words within images;

**Performance** Percent of words correctly classified;

**Training Experience** A database of handwritten words with given classifications.

Some other examples of machine learning are the recognition of spoken words, the drive of an autonomous vehicle, the classification of new astronomical structures, or learning to play backgammon [26].

Before discussing machine learning algorithms, is important to understand the meaning of concepts, instances and attributes.

## 2.1 Concepts, Instances and Attributes

Basically, the concept is the "thing" to be learned[29]. For example, people learn general concepts such as "bird" or "car". Each concept can be viewed as describing some subset of objects or events defined over a larger set. The "bird" concept can be seen as describing the subset of animal with beak, wings and small proportions that flies. Alternatively, each concept can be defined as a boolean-valued function defined over this larger set. Back to the "bird" concept, this can be seen as a boolean function defined over all animals, whose value is true for birds and false for other animals[26].

Examples, or instances, are the data to be evaluated. Normally, each instance is an individual, independent example of the concept to be learned and is characterized by the values of a set of predetermined features or *attributes*. Instances are the input data of machine learning algorithms and belong, preferentially, to one single class. When this doesn't happen, they receive the nomination of *multilabeled instances*[29].

The value of an attribute for a particular instance is a measurement of the quantity to which the attribute refers. It can be *numeric* (or *continuous*) or *nominal* (or *categorical*). Note that the term continuous here is not the mathematical one, once integer-valued attributes are certainly not continuous in the mathematical sense. Nominal attributes can assume values in a pre-specified, finite set of possibilities.

The iris classification problem, a classic example for machine learning algorithms, is shown in Table 2.1. In this example, the concept to be learned is the type of iris, the attributes are Sepal length and width and Petal length and width. Instances are the lines of the table, each line representing an instance.

Other classic example is the weather problem present in Table 2.2. It is fictitious and concerns the conditions that are suitable for playing some unspecified game.

Table 2.1: Iris Data.

Sepal Length (cm)	Sepal Width (cm)	Petal Length (cm)	Petal Width (cm)	Type
5.1	3.5	1.4	0.2	<i>Iris setosa</i>
4.9	3.0	1.4	0.2	<i>Iris setosa</i>
4.7	3.2	1.3	0.2	<i>Iris setosa</i>
4.6	3.1	1.5	0.2	<i>Iris setosa</i>
5.0	3.6	1.4	0.2	<i>Iris setosa</i>
7.0	3.2	4.7	1.4	<i>Iris versicolor</i>
6.4	3.2	4.5	1.5	<i>Iris versicolor</i>
6.9	3.1	4.9	1.5	<i>Iris versicolor</i>
5.5	2.3	4.0	1.3	<i>Iris versicolor</i>
6.5	2.8	4.6	1.5	<i>Iris versicolor</i>
6.3	3.3	6.0	2.5	<i>Iris virtinica</i>
5.8	2.7	5.1	1.9	<i>Iris virtinica</i>
7.1	3.0	5.9	2.1	<i>Iris virtinica</i>
6.3	2.9	5.6	1.8	<i>Iris virtinica</i>
6.5	3.0	5.8	2.2	<i>Iris virtinica</i>

## 2.2 Types of Machine Learning

As said before, we can say that a machine is learning if it is improving its performance at some task through practice. If we consider the question of knowing whether or not the machine is learning (how it gets experience), the two possible answers to that question produce different types of machine learning: *supervised learning* and *unsupervised learning*.

**Supervised Learning** In this type of learning it is provided to the machine a set of examples with the correct classifications (concepts or *targets*) and, based on this training set, the algorithm learns to respond correctly to all possible inputs. The set of examples given to the machine is called *training set*[31]. This is the most common type of learning and is going to be detailed below.

**Unsupervised Learning** In this type of learning the correct classifications are not provided. Instead, the algorithm tries to identify resemblances between the inputs so that inputs that have something in common are categorised together[31]. Hence a concise description of the data can be a set of clusters or a probability density stating how likely it is to observe a certain object in the future[32].

The iris problem previously mentioned can be seen as a clustering problem if the iris type was omitted. In that case, the output would be as presented in Table 2.3. It is likely that all instances fall into natural clusters corresponding to the three iris types.

Table 2.2: Weather Data.

Outlook	Temperature (K)	Humidity (%)	Windy	Play
Sunny	85	85	false	no
Sunny	80	90	true	no
Overcast	83	86	false	yes
Rainy	70	96	false	yes
Rainy	68	80	false	yes
Rainy	65	70	true	no
Overcast	64	65	true	yes
Sunny	72	95	false	no
Sunny	69	70	false	yes
Rainy	75	80	false	yes
Sunny	75	70	true	yes
Overcast	72	90	true	yes
Overcast	81	75	false	yes
Rainy	71	91	true	no

### 2.2.1 Supervised Learning

In supervised learning the task is to find a deterministic function that maps any input to an output that can predict future input-output observations, minimizing errors as much as possible. Whenever asked for the target value of an object present in the training set, it can return the value that appeared the highest number of times together with this object in the training set. According to the types of the outputs, supervised learning can be distinguished in *classification* and *regression* learning[32].

**Classification learning** is a machine learning method that is presented with a set of classified examples from which it is expected to learn a way of classifying unseen examples. In this case the output is a *class* and the learning algorithm that solves the classification problem is called the *classifier*[32, 29].

**Regression learning** is a type of learning where there is no specified classes[29]. The output space is formed by the values of continuous variables (e.g. estimate a physical measure)[32].

## 2.3 Output Representations

An understanding of how knowledge is represented and interpreted is an useful insight into how data mining works. To exemplify the different output representations, the data from Table 2.4 is used as input.

**Table 2.3:** Iris Problem as a Clustering Problem.

<b>Sepal Length (cm)</b>	<b>Sepal Width (cm)</b>	<b>Petal Length (cm)</b>	<b>Petal Width (cm)</b>
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
7.0	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.9	3.1	4.9	1.5
5.5	2.3	4.0	1.3
6.5	2.8	4.6	1.5
6.3	3.3	6.0	2.5
5.8	2.7	5.1	1.9
7.1	3.0	5.9	2.1
6.3	2.9	5.6	1.8
6.5	3.0	5.8	2.2

**Table 2.4:** Mushroom quality problem data.

<b>Weight</b>	<b>Stalk damage</b>	<b>Dirt</b>	<b>Firmness</b>	<b>Quality</b>
heavy	hight	mild	hard	poor
heavy	hight	mild	soft	poor
normal	hight	mild	hard	good
light	medium	mild	hard	good
light	clear	clean	hard	good
heavy	clear	clean	soft	poor
normal	clear	clean	soft	good
heavy	medium	mild	hard	poor
heavy	clear	clean	hard	good
light	medium	clean	hard	good
heavy	medium	clean	soft	good
normal	medium	mild	soft	good
normal	hight	clean	hard	good
light	medium	mild	soft	poor

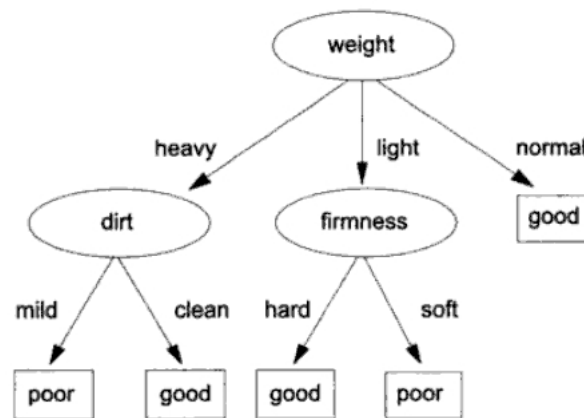


Figure 2.1: Decision tree for the mushroom data.

### 2.3.1 Decision Trees

One method of machine learning is to determine a test on an attribute that will discriminate between the instances in a data set. This test has the effect of splitting the data into two (or more) smaller data sets. By repeatedly splitting these smaller data sets, a series of attribute tests that represent the original data as a tree are obtained. An illustration of a decision tree is presented in Figure 2.1[33].

New instances are classified by starting at the root node of the tree, testing the addressed attribute for this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is then repeated for the sub-tree rooted at the new node[29, 26].

In Section 2.5, decision trees induction algorithms will be discussed.

### 2.3.2 Classification rules

An alternative to a decision tree is to represent knowledge as a set of rules. Rule sets can be either induced directly or produced from a decision tree. For example, the tree in Figure 2.1 can be readily transformed, generating a rule for each one of the leaves of the tree, into the following rules:

```

If weight=heavy AND dirt=mild then quality=poor.
If weight=heavy AND dirt=clean then quality=good.
If light=heavy AND firmness=hard then quality=good.
If light=heavy AND firmness=soft then quality=poor.
If normal=heavy AND dirt=mild then quality=good.
  
```

The conditions of the rule are the tests encountered from the leaf to the root of the tree (combined as conjunctions) and the classification given by the rule is the class label of the leaf[33].



## 2.4 Training and Testing

In the final phase of the data-mining process, when the model is obtained, it is important to validate it. Model validation is performed by verifying if models behave with satisfactory accuracy consistent with the objectives defined by the users, within its domain of applicability[34].

The data-mining results are validated by the testing process. Model testing relies on ascertaining if inaccuracies exist or revealing the existence of errors on the model[34].

### 2.4.1 Error rate and performance

For classification problems, the measurement of a classifier performance in terms of *error rate* is a common practice. The classifier predicts the class of each instance: if it is correct, it is counted as a *success*; otherwise, it is an *error*. The error rate is the proportion of errors made over a whole set of instances, and it measures the overall performance of the classifier.

The error rate on the training set is not likely to be a good indicator of future performance because the classifier has been learned from the very same training data, and therefore any performance estimate based on that will be optimistic, even hopelessly optimistic. This measurement is called the *resubstitution error* because it is calculated by resubstituting the training instances into a classifier that was built from them. Although it is not a reliable predictor of the true error rate on new data, it is nevertheless often useful to know.

To predict the performance of a classifier on new data, we need to assess its error rate on a dataset that played no part in the formation of the classifier. This independent dataset is called the *testing set*. It is assumed that both training and testing data are representative samples of the underlying problem[29].

There are no good guidelines available on how to divide the samples into subsets. No matter how the data is split, it should be clear that different random splits even with the specified size of training and testing sets, would result in different error estimates.

There are different techniques, usually called *resampling methods*, for splitting data sets into training and testing samples[34].

**Resubstitution Method** is the simplest method. All the available data is used for training as well as for testing. In other words, the training and testing sets are the same. Estimation of the error rate for this "data distribution" is optimistically biased (estimated error is often smaller than could be expected in real applications of the model), and therefore the method is rarely used in real-world data-mining applications. This is especially the case when the ratio of sample size to dimensionality is small.

**Holdout Method** is a method in which half the data, or sometimes two-thirds of the data, is used for training and the remaining data is used for testing. Training and

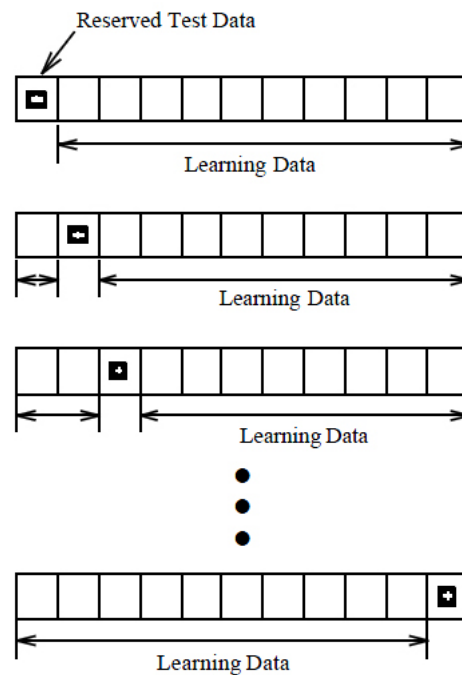


Figure 2.2: Cross-Validation method illustration from [36].

testing sets are independent and error estimation is pessimistic. Different partitioning will give different estimates. A repetition of the process, with different training and testing sets randomly selected and integration of the error results into one standard parameter will improve the estimate of the model.

**Leave-one-out Method** is a method where the learning algorithm is applied once for each instance, using all other instances as a training set and using the selected instance as a single item test set[35].

**Rotation Method or  $n$ -fold Cross-Validation** is an approach compromised between hold-out and leave-one-out methods. It divides the available samples into  $P$  disjoint subsets, where  $P \leq P \leq n$ .  $(P - 1)$  subsets are used for training and the remaining subset for testing. This is the most popular method in practice, especially for problems where the number of samples is relatively small[34]. This method schematics is shown in Figure 2.2.

**Bootstrap method** resamples the available data with replacements to generate a number of "fake" data sets of the same size as the given data set. The number of these new sets is typically several hundreds. These new training sets can be used to define bootstrap estimates of the error rate. Experimental results have shown that the bootstrap estimates can outperform the cross-validation estimates. This method is especially useful in small data set situations[34].

Generally, the larger the training sample, the better the classifier, although the outputs begin to diminish once a certain volume of training data is exceeded and the larger the

test sample, the more accurate is the error estimate[29].

### 2.4.2 Cost

The evaluations that have been discussed so far do not take into account the cost of making wrong decisions/classifications.

Let's take an example in which not considering the cost can lead to the construction of an useless classifier. The problem was to determine the exact day that each cow in a daily herd was in estrus, or "heat". Cows were identified by electronic ear tags, and various attributes were used such as *milk volume* and *milk chemical composition* (automatically recorded by a high-tech machine), and *milking order*, because cows herds are regular and generally cows arrive in the milking shed in the same order, except in unusual circumstances such as estrus. It is important to know when a cow is in estrus because they are fertilized by artificial insemination and missing a cycle will delay calving unnecessarily, causing complications down the line. In early experiments, machine learning schemes stubbornly predicted that each cow was never in estrus. Like humans, cows have a menstrual cycle of approximately 30 days. So, this "null" rule is correct about 97% of the time, a very good degree of accuracy. However, the main goal was to establish rules that predicted the "in estrus" situation more accurately than the "not in estrus" one. The costs of the two kinds of error were different. Evaluation by classification accuracy tacitly assumes equal error costs[29].

Cost-Sensitive learning is a type of learning that takes the misclassification costs into consideration. The goal of this type of learning is to minimize the total cost. The key difference between cost-sensitive learning and cost-insensitive learning is that the first one treats different misclassifications differently. In other words, the cost for labelling a positive example as negative can be different from the cost for labelling a negative example as positive[35].

Considering a binary classification (i.e., positive and negative class), the cost of false positive (actual negative but predicted as positive; denoted as FP), false negative (FN), true positive (TP), and true negative (TN) can be given in a cost matrix, as shown in Table 2.5.

In Table 2.5, the notation  $C(i, j)$  is also used to represent the misclassification cost of classifying an instance from its actual class  $j$  into the predicted class  $i$  (1 is used for positive and 0 to negative). For multiple classes, the cost matrix can be easily extended by adding more rows and more columns, resulting in a *confusion matrix*.

Given the cost matrix, an example should be classified into the class that has the minimum expected cost. This is the minimum expected cost principle. The expected cost  $R(i|x)$  of classifying an instance  $x$  into class  $i$  can be expressed as

$$R(i|x) = \sum_j P(j|x)C(j, i),$$

where  $P(j|x)$  is the probability estimation of classifying an instance into class  $j$ [35, 29].

**Table 2.5:** An example of the cost matrix for binary classification by cost-sensitive learning.

	Actual Negative	Actual Positive
Predict negative	$C(0, 0)$ or TP	$C(0, 1)$ or FN
Predict positive	$C(1, 0)$ or FP	$C(1, 1)$ or TP

## 2.5 Decision Trees Induction

The two most commonly used systems for induction of decision trees classification are C4.5 and CART[37].

### 2.5.1 C4.5

#### 2.5.1.1 Construct Decision Trees

C4.5 algorithm is the Quilan's extension of his own ID3 algorithm for generating decision trees[38]. This algorithm uses a technique called *divide and conquer*. It starts with all the training samples at the root node of the tree. An attribute is selected to partition these samples. For each value of the attribute a branch is created, and the corresponding subset of samples that have the attribute value specified by the branch is moved to the newly created child node. The algorithm is applied recursively to each child node until all samples at a node are of one class. Every path to the leaf in the decision tree represents a classification rule. Note that the critical decision in such a top-down decision tree-generation algorithm is the choice of the attribute at a node.

So, according to this algorithm, suppose that we have the task of selecting a possible test with  $n$  outcomes ( $n$  values for a given feature) that partitions the set  $T$  of training samples into subsets  $\{T_1, T_2, \dots, T_n\}$ . Let the classes be denoted as  $\{C_1, C_2, \dots, C_k\}$ ,  $S$  is a set of samples and  $\text{freq}(C_i, S)$  stands for the number of samples in  $S$  that belong to class  $C_i$  (out of  $k$  possibilities), and let  $|S|$  denote the number of samples in the set  $S$ .

This algorithm uses a criterion called *gain* to select the attribute to be tested which is based on the information theory concept, the *entropy*. The following relation gives the computation of the entropy of the set  $S$ :

$$\text{Info}(S) = - \sum_{i=1}^k \frac{\text{freq}(C_i, S)}{|S|} \log_2 \left( \frac{\text{freq}(C_i, S)}{|S|} \right) \text{ bits} \quad (2.1)$$

Now considering a similar measurement after  $T$  has been partitioned in accordance with  $n$  outcomes of one attribute test  $X$ . The expected information requirement can be found as the weighted sum of entropies over the subsets:

$$\text{Info}_X(T) = - \sum_{i=1}^n \frac{|T_i|}{|T|} \text{Info}(T_i) \text{ bits} \quad (2.2)$$

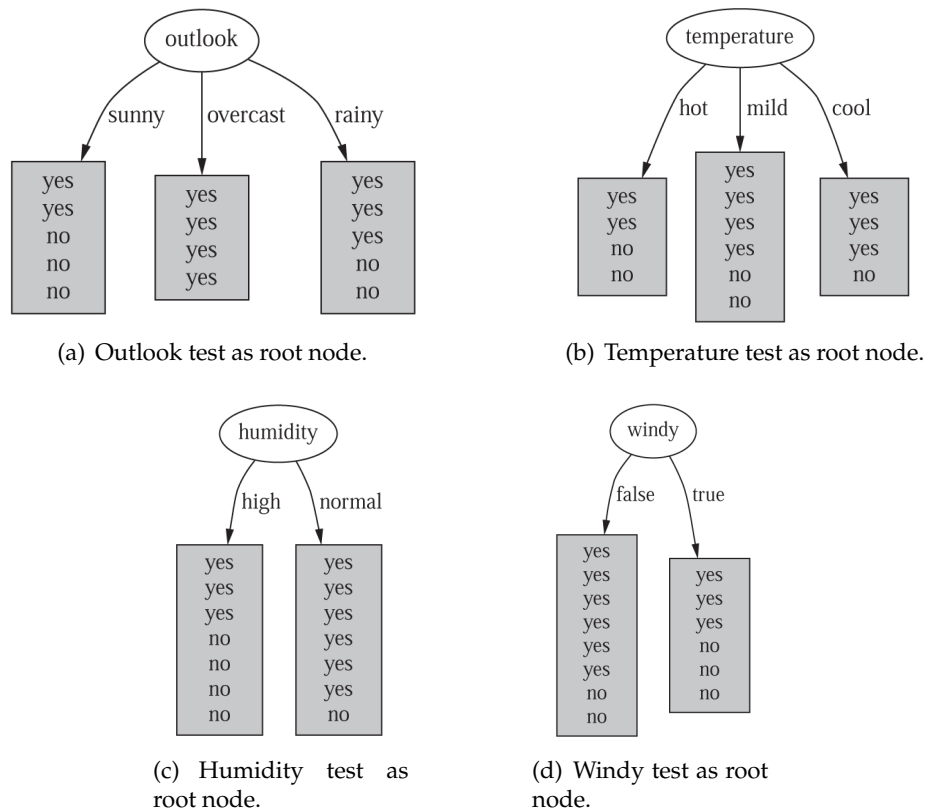
The quantity

$$\text{gain}(X) = \text{Info}(T) - \text{Info}_X(T) \text{ bits} \quad (2.3)$$

measures the information that is gained by partitioning  $T$  in accordance with the test  $X$ . The gain criterion selects a test  $X$  to maximize the *gain*.

To a better understanding of how to construct a decision tree with this algorithm we will consider again the weather problem, presented in Table 2.2. [39, 34]

The first problem is to select an attribute to place at the root node. Considering the weather problem, there are four different possibilities for the root node: *outlook*, *temperature*, *humidity* or *windy*. Trees that will result from selecting each attribute to the root node are shown in Figure 2.3.



**Figure 2.3:** Possibilities to the root node test, for the weather data tree.

So, starting with the *outlook* test which has  $n = 3$  possible outcomes. We have a set  $S$  that reaches that node with 14 samples, so  $|S| = 14$ . We have two different classes, *play*

and *don't play*. This way, Equation 2.1 becomes

$$\text{Info}(S) = - \left( \frac{9}{14} \log_2 \frac{9}{14} \right) - \left( \frac{5}{14} \log_2 \frac{5}{14} \right) = 0.940 \text{ bits}$$

Supposing now that  $T$  has been partitioned according to the 3 outcomes of *outlook* attribute, we have

$$\text{Info}(\textit{sunny}) = - \left( \frac{2}{5} \log_2 \frac{2}{5} \right) - \left( \frac{3}{5} \log_2 \frac{3}{5} \right) = 0.971 \text{ bits}$$

$$\text{Info}(\textit{overcast}) = - \left( \frac{0}{4} \log_2 \frac{0}{4} \right) - \left( \frac{4}{4} \log_2 \frac{4}{4} \right) = 0.0 \text{ bits}$$

$$\text{Info}(\textit{rainy}) = - \left( \frac{3}{5} \log_2 \frac{3}{5} \right) - \left( \frac{2}{5} \log_2 \frac{2}{5} \right) = 0.971 \text{ bits}$$

So,

$$\text{Info}_{\textit{outlook}}(T) = - \left( \frac{5}{14} 0.971 + \frac{4}{14} 0 + \frac{5}{14} 0.971 \right) = 0.693 \text{ bits}$$

This way,

$$\text{gain}(\textit{outlook}) = 0.971 - 0.693 = 0.278 \text{ bits}$$

Calculating the gain for all the other attributes, we obtain

$$\text{gain}(\textit{temperature}) = 0.029 \text{ bits}$$

$$\text{gain}(\textit{humidity}) = 0.152 \text{ bits}$$

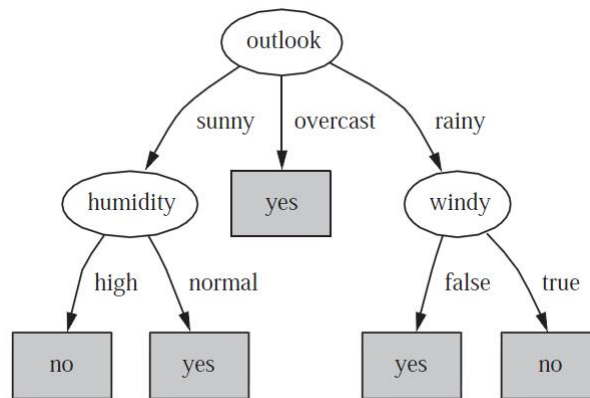
$$\text{gain}(\textit{windy}) = 0.048 \text{ bits}$$

Then we continue, recursively for all the other nodes and in the final of the process we would have the decision tree shown in Figure 2.4[34].

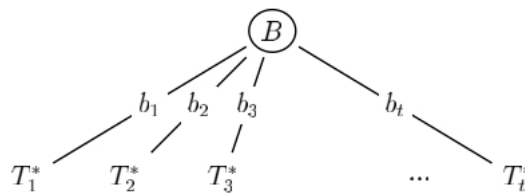
### 2.5.1.2 Pruning

A fully decision tree often contains unnecessary structure, and it is generally advisable to simplify it before it is deployed[29].

After a decision tree is produced by the divide and conquer algorithm, C4.5 prunes it in a single bottom-up pass[37].



**Figure 2.4:** Decision tree for the weather data.



**Figure 2.5:** Hypothetical decision trees.

To decide where to prune, it is necessary to estimate the error rate expected at a particular node given an independently chosen test set. We need to estimate the error at internal nodes as well as at leaf nodes[29].

Considering that an event occurs  $M$  times in  $N$  trials, the ratio  $M/N$  is an estimate of the probability  $p$  of the event. Deriving the confidence limits for  $p$  (for example, by a Bernoulli process[29])  $CF$ , the upper limit  $U_{CF}(M, N)$  is used as a conservative estimate of the error rate on unseen cases[37].

Let  $T$  be a non-leaf decision tree, produced from a training set  $S$ , as shown in the hypothetical tree from Figure 2.5, where each  $T_i^*$  has already been pruned. Further, let  $T_f^*$  be the subtree corresponding to the most frequent outcome of  $B$ , and let  $L$  be a leaf labelled with the most frequent class in  $S$ . Let the number of cases in  $S$  misclassified by  $T$ ,  $T_f^*$ , and  $L$  be  $E_T, E_{T_f^*}$ , and  $E_L$  respectively. The C4.5's tree pruning algorithm considers the three corresponding estimated error rates[37]:

- $U_{CF}(E_T, |S|)$ ,
- $U_{CF}(E_L, |S|)$  and
- $U_{CF}(E_{T_f^*}, |S|)$ .

Depending on whichever is lower, C4.5 either:

- leaves  $T$  unchanged,
- replaces  $T$  by the leaf  $L$  or

- replaces  $T$  by the subtree  $T_f^*$ .

## 2.5.2 CART

Classification and Regression Trees (CART) methodology was developed in the 80's by Leo Breiman, Jerome Friedman, R.A. Olshen and Charles Stone and was first presented in their paper from 1984[40]. It is a method to construct binary decision trees or, in other words, CART only asks *yes/no* questions[41]. The CART authors argue that binary splits are to be preferred to multiway splits because they fragment the data more slowly than multiway splits and repeated splits on the same attribute are allowed and, if selected, will eventually generate as many partitions for an attribute as required. Any loss of ease in reading the tree is expected to be offset by the improved predictive performance[42].

It is important to define some CART components. The first one is the *categorical outcome* or *dependent variable*. This variable is the same as the concept described in Section 2.1. The second component are the *predictors* or *independent variables* and they correspond to the attributes that were previously mentioned. However, the CART algorithm will decide if whether or not those attributes will be related to the outcome variable of interest[36].

An important practical property of CART is that the structure of its classification or regression trees is invariant with respect to monotone transformations of independent variables. One can replace any variable with its logarithm or square root value and the structure of the tree will not change[41].

CART methodology consists of tree parts:

1. Construction of the *maximum tree*
2. Pruning
3. Choice of the right tree size

The CART mechanism is intended to produce not one tree, but a sequence of nested pruned trees, each of which is a candidate to be the optimal tree.

Lets consider only the classification trees.

### 2.5.2.1 Construct the maximum tree

CART splitting rules are always couched in the form: "An instance goes left if CONDITION and goes right otherwise", where the CONDITION is expressed as "attribute  $X_i \leq C$ " for continuous attributes. For categorical or nominal attributes the CONDITION is expressed as membership in a list values. For examples as "An instance goes left if CITY is Chicago, Detroit, Nashville" and goes right otherwise[42].

To each node, CART software finds the best possible variable to split the node into two child nodes. In order to find the best variable, the software checks all possible splitting variables (or *splitters*), as well as all possible values of the variable to be used to split the



node. In choosing the best splitter, the program seeks to maximize the "purity" weighted average of the two child nodes. A number of different purity measures can be selected, loosely called *splitting criteria*, *splitting functions* or *splitting rules*[36].

The CART authors discuss examples using four splitting rules for classification trees (Gini, Twoing, Ordered Twoing and Symmetric Gini), but the monograph focuses most of its discussion on the Gini, which is similar to the better known entropy (information-gain) criterion, used by C4.5[42].

Let  $RF(C_j, S)$  denote the relative frequency of cases in  $S$  that belong to class  $C_j$ . The Gini index is defined as

$$I_{gini}(S) = 1 - \sum_{j=1}^x RF(C_j, S)^2$$

and the information gain due to a split is computed as

$$G(S, B) = I(S) - \sum_{i=1}^t \frac{S_i}{S} I(S_i)$$

where  $S_i$  are subsets partitioned of  $S$  by a test  $B$ [37].

By convention, splits on continuous variables send instances with larger values of the splitter to the right, and splits on nominal variables are defined by the lists of values going left or right.

In the diagram the terminal nodes are color coded to reflect the relative probability of response. A red node is above average in response probability and a blue node is below average[42].

CART also have ways to handle missing values, which is a feature that won't be specified due to its non-interest for this work.

### 2.5.2.2 Pruning

As mentioned above, the tree building process goes on until one of the following conditions is achieved:

1. There is only one observation in each of the child nodes;
2. All observations within each child node have the identical distribution of predictor variables, making splitting impossible;
3. An external limit on the number of levels in the maximal tree has been set by the user (*depth* option).

After achieving the maximal tree, an iterative pruning process is applied, resulting in a sequence of successively smaller pruned trees.

CART uses a *cost-complexity pruning* process. In this process, a nested sequence of subtrees of the initial large tree is created by "weakest-link cutting"[36].

This technique assumes that the bias in the resubstitution error of a tree increases linearly with the number of leaf nodes. The cost assigned to a subtree is the sum of two terms: the resubstitution error and the number of leaves times a complexity parameter  $\alpha$ . Formally,

$$R_\alpha = R(T) + \alpha \cdot \text{numberOfLeaves}$$

It can be shown that, for every value of  $\alpha$ , there is a unique smallest tree minimizing  $R_\alpha$ . Although  $\alpha$  runs continuously through a set of values, there is at most a finite number of possible subtrees. Thus, there is a sequence of trees minimizing  $R_\alpha$ , created by varying  $\alpha$  from 0 to infinity.

Pruning involves removing the split generating two terminal nodes and absorbing the two children into their parent, thereby replacing the two terminal nodes with one[42].

### 2.5.2.3 Select the best tree

At this point, we have to decide the tree to use. The goal in selecting the optimal tree, defined based on the expected performance on an independent set of data, is to find a correct complexity parameter  $\alpha$  so that the information in the learning dataset is fit but not overfit. Generally, finding this value for  $\alpha$  would require an independent set of data, but this requirement can be avoided by using the technique of cross validation[36] as described in Section 2.4.1.



# Image Processing

In this chapter some image processing aspects and techniques important to the development of this work, as histograms, interpolation techniques and segmentation methods are presented.

## 3.1 Digital Image Histogram

Intensity transformation functions based on information extracted from image intensity histograms play a basic role in image processing, in areas such as enhancement, compression and segmentation[11].

The histogram of a digital image with  $L$  intensity levels in the range  $[0,G]$  is defined as the discrete function

$$h(r_k) = n_k \quad (3.1)$$

where  $r_k$  is the  $k$ -th intensity level in the interval  $[0,G]$  and  $n_k$  is the number of pixels in the image whose intensity level is  $r_k$ [11].

The histogram provides a convenient summary of the intensities in an image, but it is unable to convey any information regarding spatial relationships between pixels. The histogram provides more insight about image contrast and brightness[43]:

- The histogram of a dark image will be clustered towards the lower gray level.
- The histogram of a bright image will be clustered towards the higher gray level.
- For a low-contrast image, the histogram will not be spread equally, that is, the histogram will be narrow.

- For a high-contrast image, the histogram will have an equal spread in the gray level.

This last two aspects are illustrated in Figure 3.1.

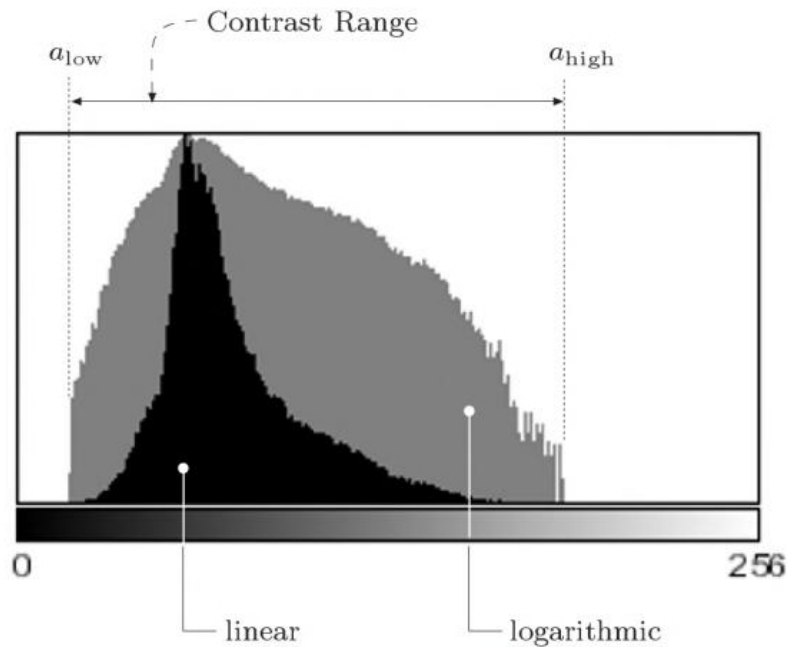


Figure 3.1: Histogram interpretation. From [44].

The equalisation of an image histogram is a process that attempts to spread out the gray levels of the image so that they are evenly distributed across their range[43]. This can be useful when the transformation of an image into a high-contrast one is desired. The equalisation process is also commonly used when a comparison between images with different intensity distributions is intended[45].

The histogram equalisation applies a point operation such that the histogram of the modified image is approximately a *uniform* distribution[45]. In general, the histogram of the processed image will not be uniform due to the discrete nature of the variables. Figure 3.2 shows an example of an histogram equalization.

## 3.2 Image interpolation

Interpolation techniques are essential to compute geometric operations on digital images, as translation, rotation and scaling[46]. The goal of these techniques is to obtain an optimal estimate for the value of the two-dimensional image function at any continuous position[47].

The general definition of a geometric operation is

$$g(x, y) = f[a(x, y), b(x, y)] \quad (3.2)$$

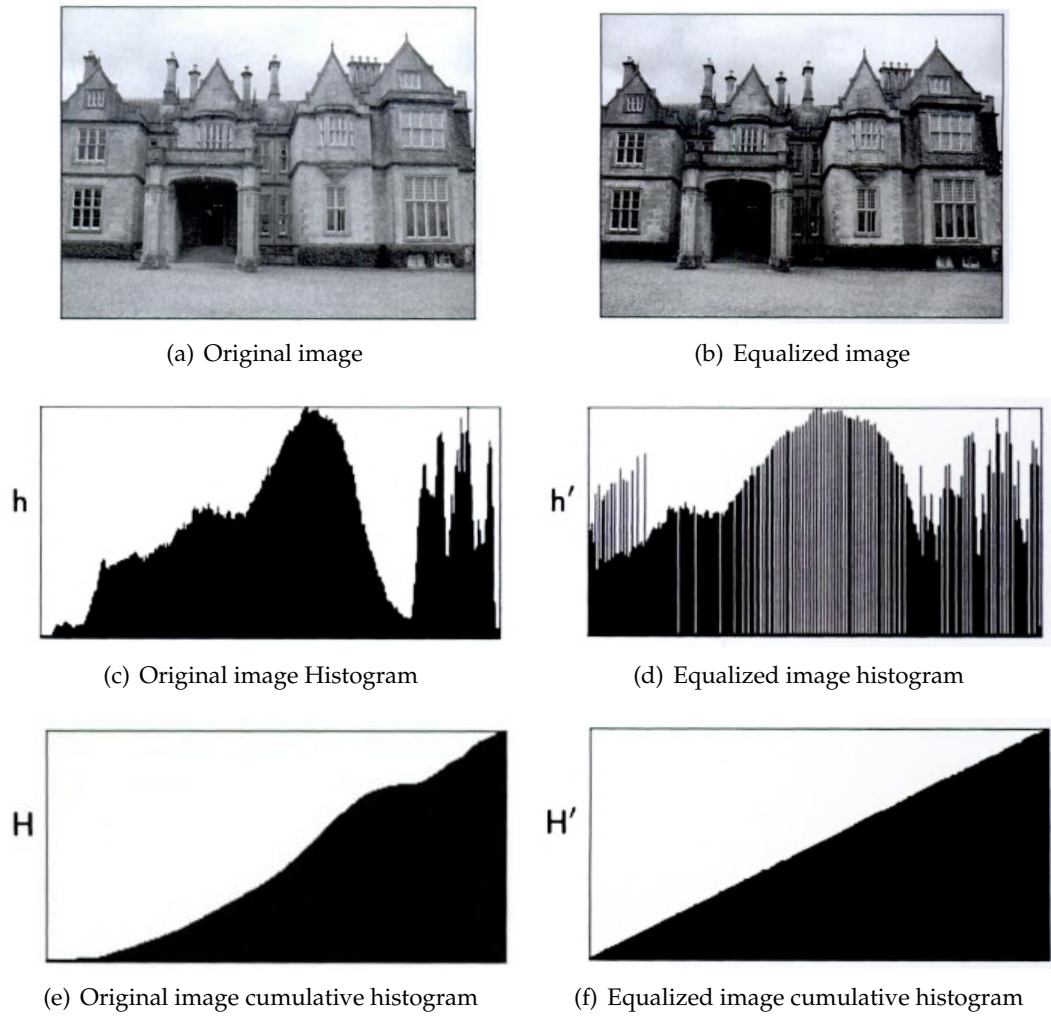


Figure 3.2: Histogram equalization results. Example from [44].

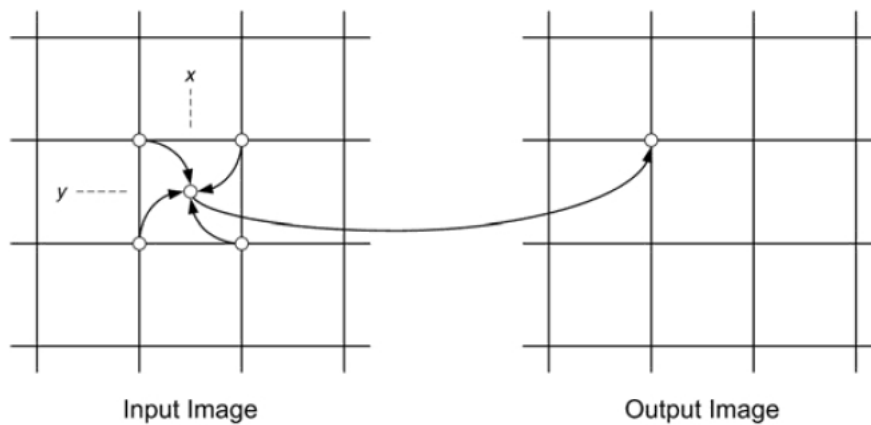


Figure 3.3: Interpolation illustration. From [46].

where  $f(x, y)$  is the input image and  $g(x, y)$  is the output image. The spacial transformation functions  $a(x, y)$  and  $b(x, y)$  specify the physical relationship between points in the input image and the corresponding points in the output image. This determines the effect that the operation will have on the image[46].

The output image is generated pixel by pixel. For each output pixel  $g(x, y)$ , the spacial transformation functions  $a(x, y)$  and  $b(x, y)$  point to a corresponding location in the input image. In general, this location falls between four adjacent pixels, as can be seen in Figure 3.3. The gray level that maps into the output pixel at  $(x, y)$  is uniquely determined by interpolation among these four input pixels[46].

In practice, the interpolation function should preserve as much detail as possible without causing visible artefacts[47].

Some of the most common interpolation methods for digital images are the 2D *nearest-neighbor*, *bilinear* and *bicubic* interpolations.

### 3.2.1 Nearest-neighbor

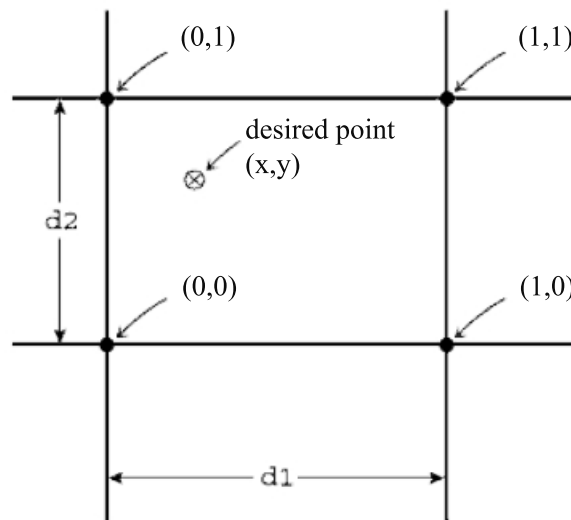
Nearest-neighbour is the simplest interpolation method to compute. Each interpolated point is simply the value of its nearest neighbor[48].

In this method, the closest pixel to a given continuous point  $(x_0, y_0)$  is found by rounding the  $x$  and  $y$  coordinates to integer values[45].

Nearest-neighbor produces strong blocking effects and, consequently, is rarely used for geometric image operations. However, in some situations, this effect may be intended; for example, if an image is to be enlarged by replicating each pixel without any smoothing[45].

### 3.2.2 Bilinear interpolation

Bilinear interpolation replaces every point with a weighted sum of the four points from the nearest  $2 \times 2$  set of pixels of the original image. The used weights for the sum are



**Figure 3.4:** Bilinear interpolation example. Adapted from [49].

determined by the distance from the location of the interpolated point to the location of the four points, where a closer distance corresponds to a larger weight[48].

In practice, bilinear interpolation approximates the continuous image by fitting a hyperbolic paraboloid through the four points. The hyperbolic paraboloid surface is given by

$$f(x, y) = ax + by + cxy + d$$

where  $a = [f(1, 0) - f(0, 0)]$ ,  $b = [f(0, 1) - f(0, 0)]$ ,  $c = [f(1, 1) - f(0, 0) - f(0, 1) - f(1, 0)]$  and  $d = f(0, 0)$ .

Figure 3.4 shows an example of this interpolation technique.

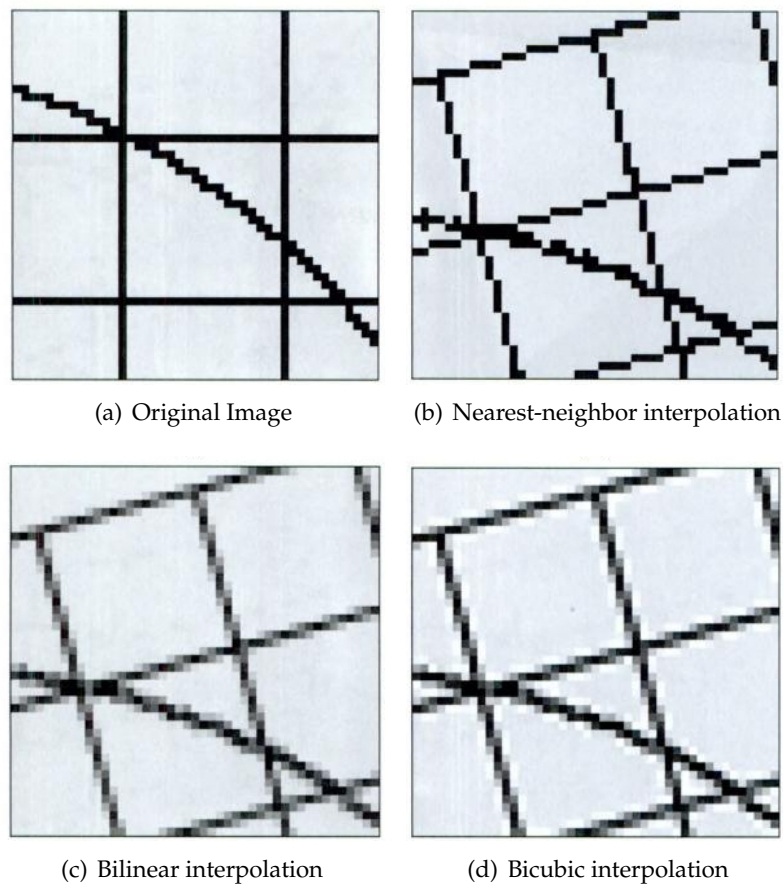
### 3.2.3 Bicubic interpolation

Bicubic interpolation works similarly to bilinear interpolation, except that bicubic interpolation replaces every point with a weighted sum of the 16 points from the nearest  $4 \times 4$  set of pixels of the original image[48].

With bicubic interpolation, the interpolated surface is given by

$$p(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j$$

The 16 coefficients  $a_{ij}$  are chosen to build the function and its derivatives continuous at the corners of the four-pixels square that contains the point  $(x, y)$ . This is done by solving 16 equations in the 16 unknown coefficients at each point. The equations are derived by setting the function and its three derivatives to their known values at the four corners[46].



**Figure 3.5:** Interpolation methods example. From [47].

Figure 3.5 shows the comparison between the three interpolation methods described above.

Despite the block effect of the nearest-neighbor interpolation, this method can be the most appropriate one in some situations, because no new pixel values are added, while in bilinear or bicubic interpolations intermediate pixel values are produced.

### 3.3 Image Segmentation

As it was stated before, image segmentation is the process of dividing an image into its constituents objects, or parts, isolating them from the image background.

There are many image segmentation techniques, being the most popular described next.

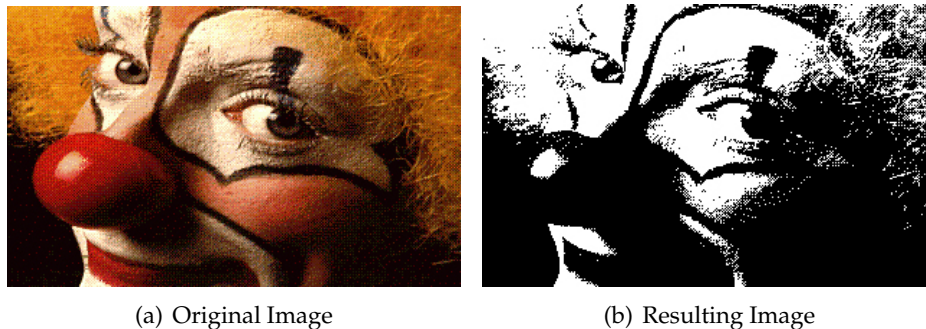
#### 3.3.1 Thresholding

Thresholding is based on the histogram of pixel intensities, implying that objects of interest are brighter or darker than other parts of the image[10]. It consists in convert an intensity image into a bi-dimensional one, "dividing" the image at a defined point,



named *threshold*. In order to get a satisfactory segmentation by thresholding, it is necessary to have a sufficiently uniform background. There are many background correction techniques but they may not always result in an image suitable for further analysis by thresholding. The transition between object and background may be diffuse, making an optimal threshold level difficult to find[10].

The most used thresholding method is the one proposed by Otsu [50] and an example is shown in Figure 3.6.



**Figure 3.6:** Otsu thresholding appliance example.

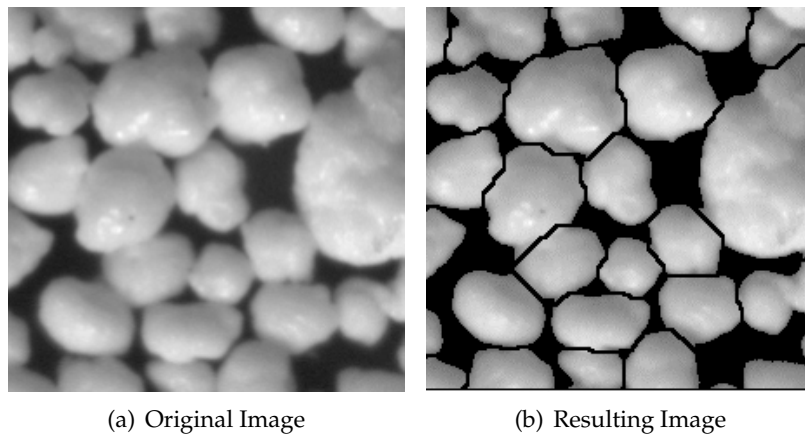
The algorithm assumes that the gray image has two classes of pixels, a foreground and a background, and the optimal threshold is selected by maximizing the separability of those classes [50].

### 3.3.2 Region Growing

Region Growing or region-based segmentation is a method that models the objects by connecting regions with similar pixels. One of the most common approaches of this method is to let regions grow from pre-defined small regions, known as *seeds*. Each region in the resulting segmented image will contain exactly one of the starting seeds. The problem with this approach to cell segmentation is that it is very difficult to construct a seeding method that puts exactly one seed in each cell[10].

One of the most popular region-method is the *watershed* algorithm[51]. The main difference between the watershed algorithm and any other ordinary region growing method is that the watershed algorithm works per intensity layer instead of per neighbour layer. If the intensity of the image is interpreted as an elevation in a landscape, the watershed algorithm will split the image into regions similar to the drained regions of that landscape. The watershed borders will be built at the crests in the image. In a gradient magnitude image, water will start to rise from minima representing areas of low gradient, i.e., the interior of the objects and the interior of the background, and the watershed borders will be built at the maxima of the gradient magnitude. However, if watershed segmentation is applied directly to the gradient magnitude image, it will almost result in over-segmentation, due to the intensity variations within both the objects and background[10].

Instead of letting "water" rise from every minimum in the image, water can be allowed



**Figure 3.7:** Watershed appliance example.

to rise only from places marked as seeds. Fully automatic foreground seeding is tricky, and when using morphological filtering, one often ends up with more than one seed per object or objects containing no seed at all. More than one seed per foreground object in many methods results in background seeds passing through foreground components, leading to incorrect segmentation results. Many seeded watershed segmentation methods are, therefore, based on manual seeding, requiring extensive user interaction[10]. In Figure 3.8 it can be seen an example of watershed appliance.

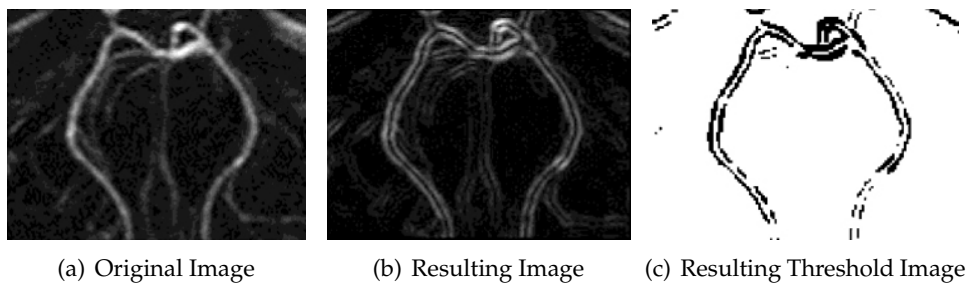
### 3.3.3 Point, line and edge-Based Segmentation

These techniques detect the three basic types of intensity discontinuities in a digital image: points, lines and edges. The most common way to look for discontinuities is to run a mask through the image. There are different masks and operators as Sobel, Roberts and Prewitt[52]. Other method for edge detection is the so-called snakes or active contour models, first described by Kass *et al.* in [18]. From a rough marking of the border or a seed inside the object of interest, a curve expands until it finds a string edge. The function describing the expansion consists of different energy terms attracting the curve to edges. Problems with those models consist in defining suitable energy terms and, again, constructing automatic seeding methods that are restricted to one unique seed per cell[10, 11].

### 3.3.4 Gradient Path Labelling (GPL)

*Gradient Path Labelling*(GPL) was developed by Mora *et al.* as part of a proposed methodology for Automatic Drusen Deposits Detection and quantification in retinal images by using digital image processing techniques[53].

Drusen are retinal abnormalities. They are visible in retinal images and their quantitative analysis is important in the follow up of the Age Regated Macular Degeneration.



**Figure 3.8:** Edge-based segmentation (Sobel) appliance example[52].

However, their evaluation is fastidious and difficult to reproduce when performed manually.

Considering that drusen are regional intensity maxima on retinal images and, in a gradient image, these regions have several ascending paths pointing towards them, the proposed algorithm for drusen detection is a novel segmentation method based on the labelling of these gradient path.

The first stage of this labelling procedure is a pixel level analysis, followed by a top-left to bottom-right direction. It starts assigning a new label to each pixel and determining its gradient azimuth using a  $3 \times 3$  *Sobel* operator, which is the direction to the ascending intensity.

The following step, *label propagation*, propagates this label following the gradient path until an already marked or outside image boundaries pixel is found. When the propagation process finishes on a different label, the two labels are tagged as *equivalents*, i.e., they are considered to belong to the same maximum.

The second stage of the labelling procedure is to *apply the equivalences*. Equivalent labels are grouped and replaced on the image by the smaller of each group, producing a segmented image with as many labels as detected objects.

Figure 3.9 shows an example of the algorithm described above.

When flat valleys or flat hills are encountered, not all gradient paths end on the same maximum pixel, resulting in an over-segmentation of the image. To solve this problem, a *merging* algorithm was introduced as the last stage of the labelling procedure.

The merging algorithm starts by creating a connectivity graph where *nodes* correspond to labels and *links* represent the adjacencies between them. Each node is characterized by the maximum intensity level of the pixel in its region and each link is set with the maximum intensity value of the border pixels between the two adjacent regions. If the difference between the link and the nodes is below a predefined threshold ( $\Delta_a$ ), the two corresponding regions are merged.

Figure 3.10 shows an example of the merging algorithm appliance. Figure 3.10(b) shows the segmentation of a small region of the image shown in Figure 3.10(a) and Figure 3.10(c) shows its corresponding connectivity graph. The connectivity graph represents the connections between segmentation areas and is labelled with the minimum

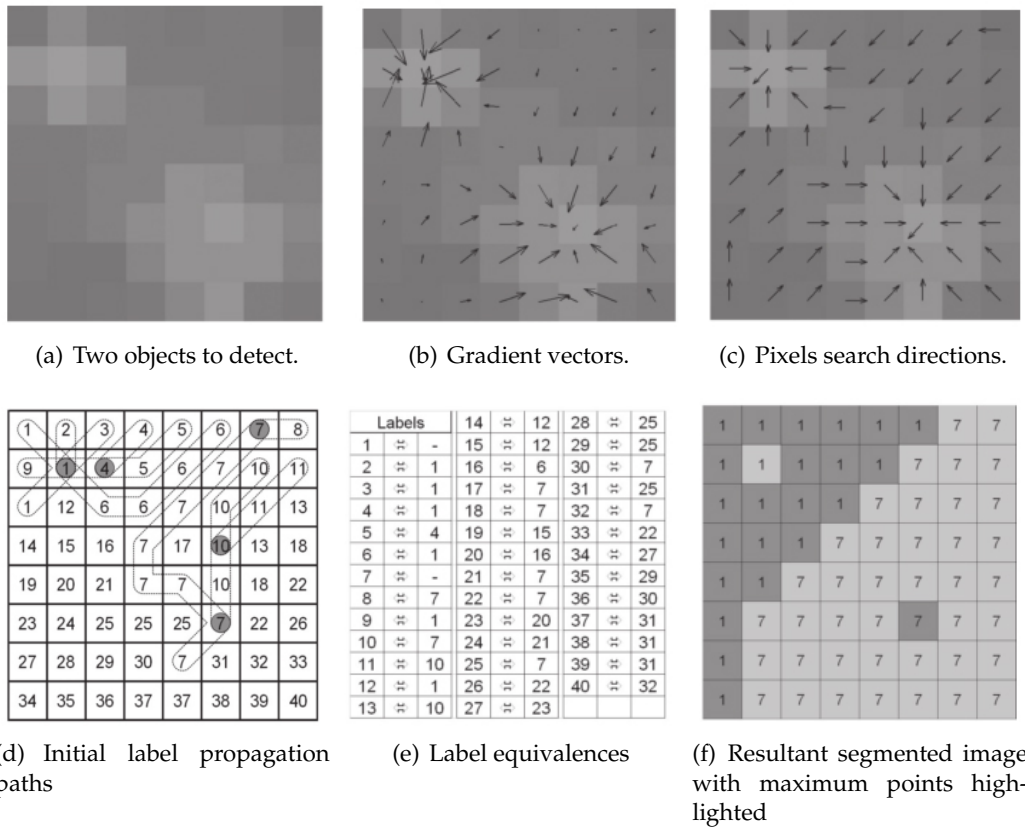


Figure 3.9: An example showing the procedures of the objects detection algorithm.

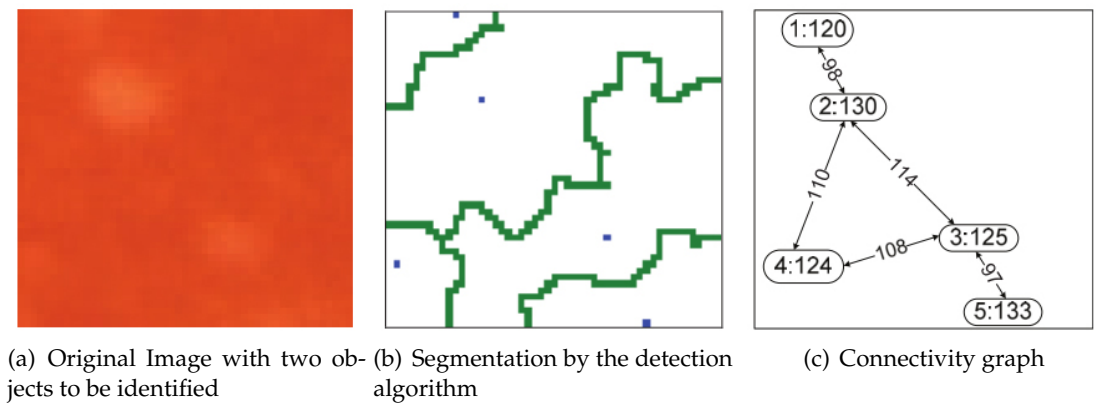


Figure 3.10: GPL Merging algorithm.

intensity border value on each link. Each node contains the label and the intensity maximum of the segmentation area that it represents.

The step-by-step GPL algorithm is shown in Figure 3.11, where TL means top-left coordinates and BR means bottom-right coordinates.

The implemented GPL algorithm allows the adjustment of a set of parameters, in such a way that the segmentation process could be adopted to other type of images.

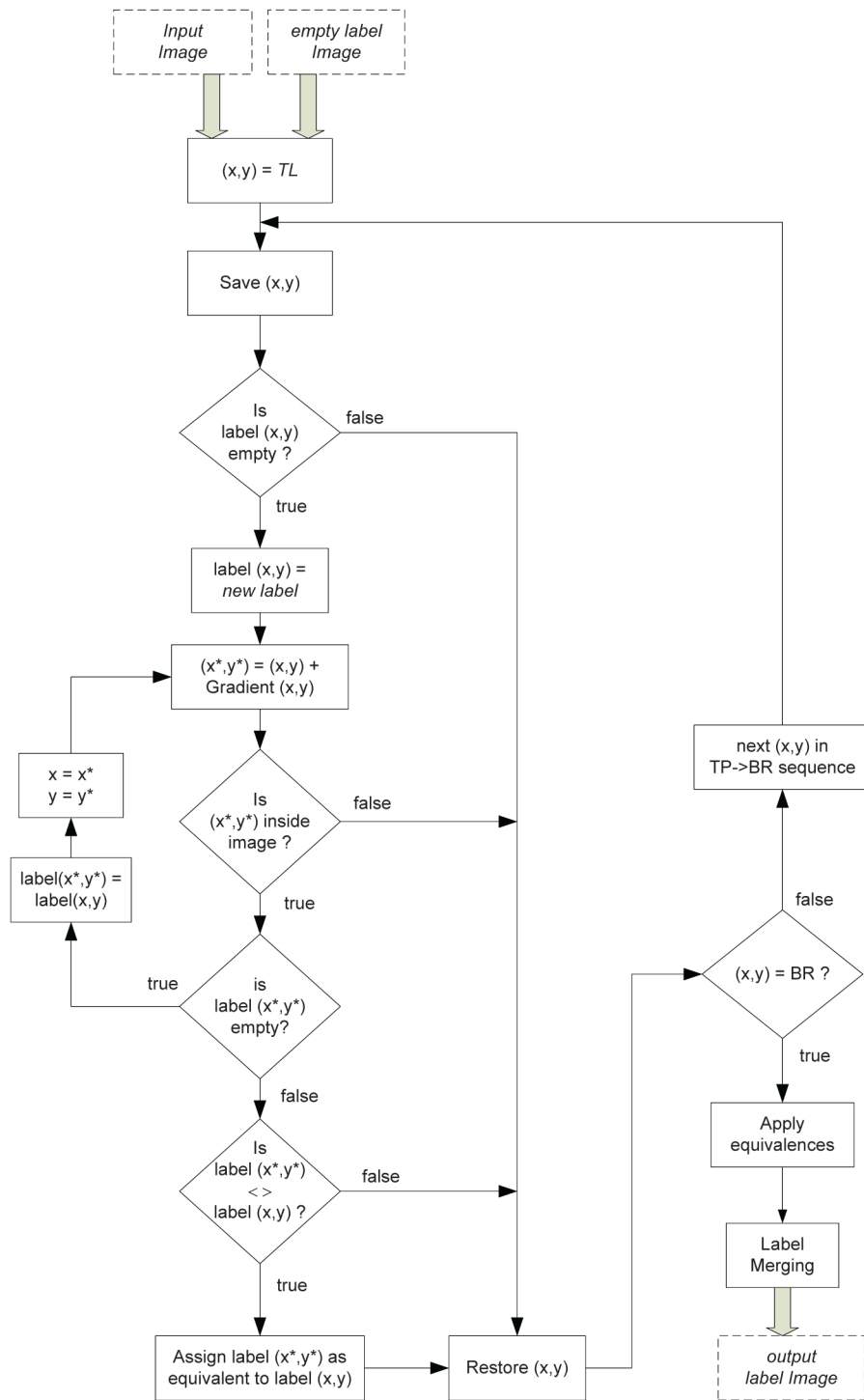


Figure 3.11: The GPL algorithm step-by-step.

# 4

## Methodology

In this chapter the main steps of the cells segmentation process of DIC images and the main steps of the alignment of those images with the correspondent fluorescence ones are presented.

The proposed algorithm was developed in MATLAB R2012a, with Windows Vista<sup>TM</sup> Home Premium, Inter(R) Core(TM)2 Duo CPU T7250 @ 2.00 GHz 2.00 GB RAM. NVIDIA GeForce 8400M GS.

Images from 5 different time series were used, with 13 Differential Interference Contrast (DIC) images and 13 Confocal Fluorescence Microscopy (CFM) images each.

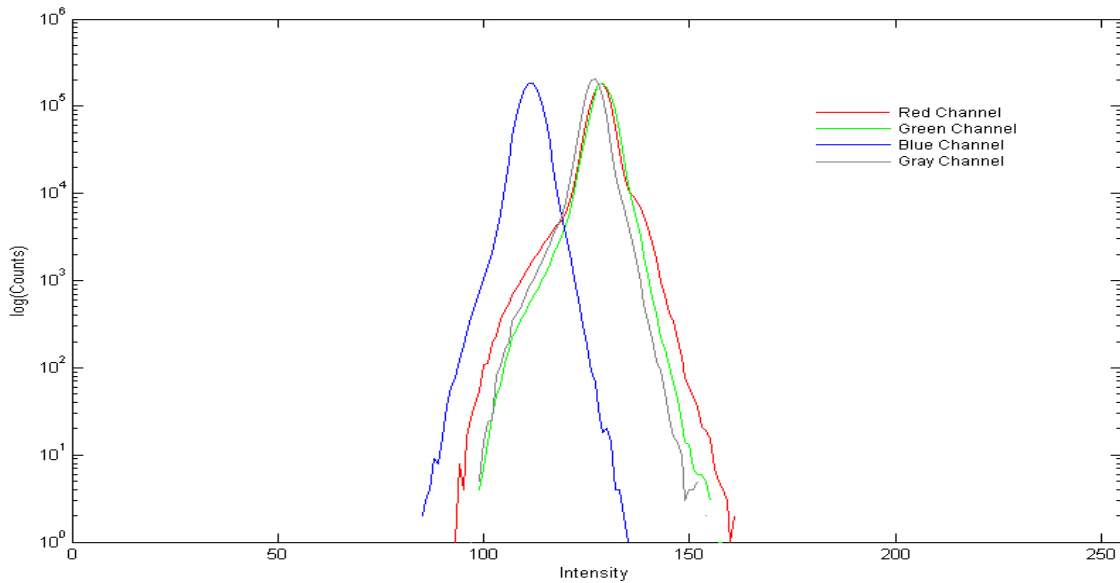
### 4.1 Cell Segmentation

#### 4.1.1 GPL

The first step of the cell segmentation process was to run images in Gradient Path Labelling algorithm. Its parameters were adjusted to produce a segmentation as accurate as possible, but once a perfect segmentation was not possible, it was opted by a moderate over-segmentation.

First of all it was necessary to choose the color channel that the GPL would use. The options were the red, the green and the blue channel or the intensity (gray) channel. The criteria used was choosing the channel with more contrast.

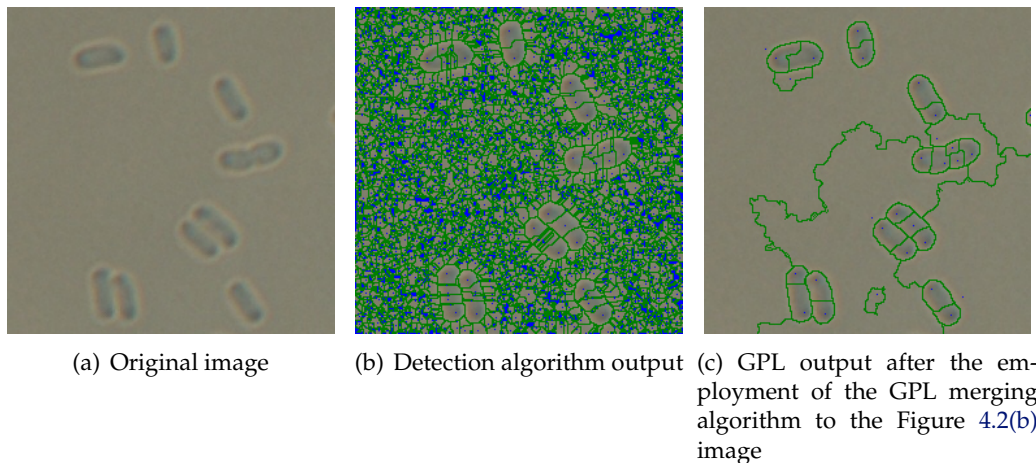
As discussed in Section 3.1, one way to compare images contrast is to compare their histogram width. Considering two bi-dimensional images, the one with more contrast would have a larger histogram. Thus, by computing the histogram of each channel it was possible to infer which had more contrast .



**Figure 4.1:** Different image channel and respective histograms.

As it can be seen in Figure 4.1, which refers to one of the used images on this study, the different channels had different histograms widths. For the image to which the histograms of this figure refers, the red channel would be the chosen one to use in GPL. Once the channel was chosen, the GPL algorithm could be applied.

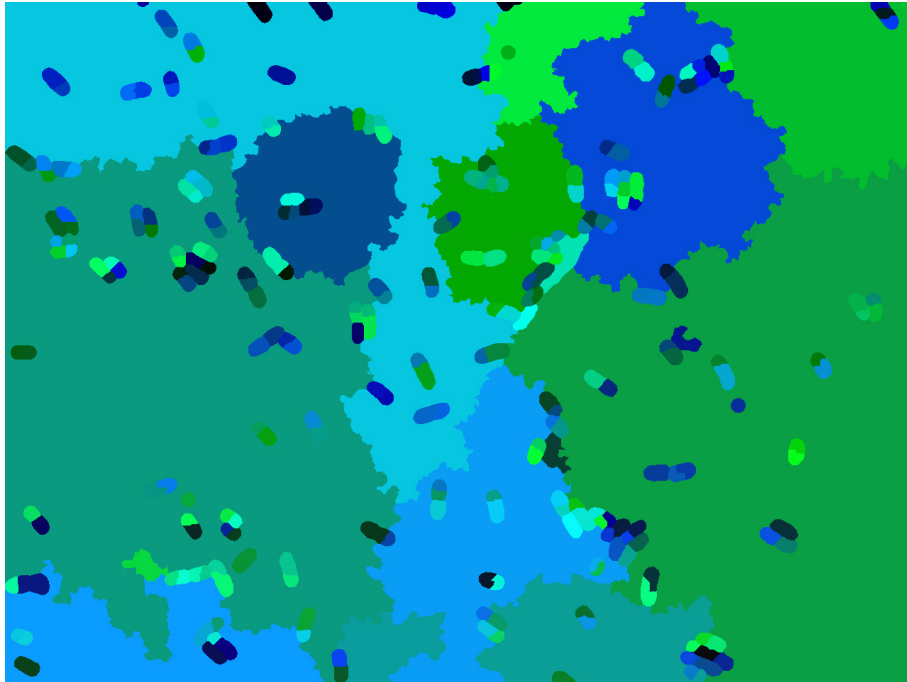
The GPL process described in Section 3.3.4, when applied to a DIC image, is shown in Figure 4.2.



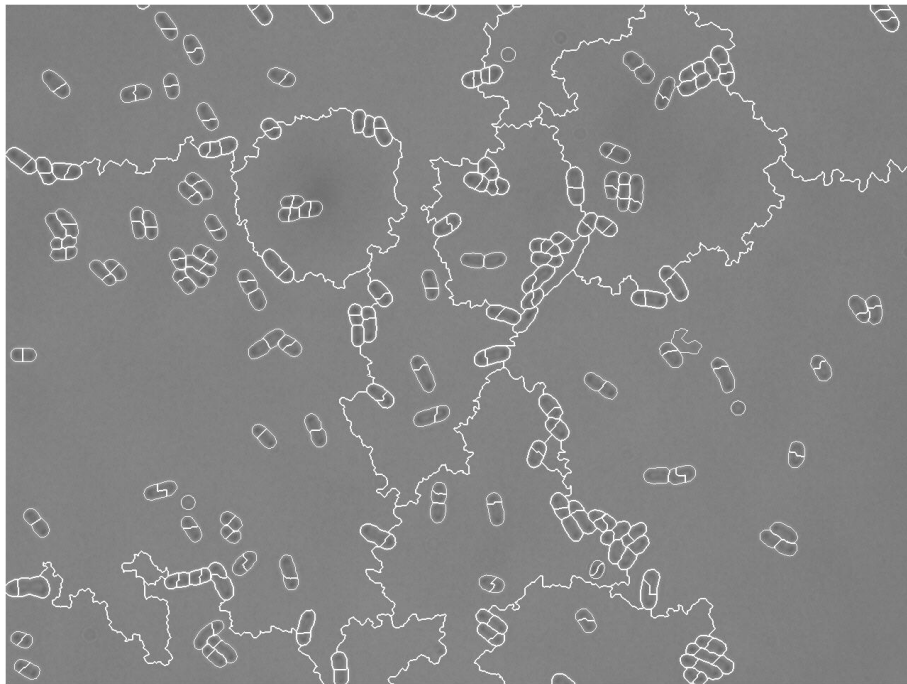
**Figure 4.2:** Example of the GPL algorithm processing a DIC image.

The output returned by the GPL is a three-dimensional labelled image as shown in Figure 4.3. Those images were converted in MATLAB to a bi-dimensional labelled image, as the one shown in Figure 4.4, where different regions were limited by a white line.





**Figure 4.3:** An example of the GPL output.



**Figure 4.4:** GPL resulting in a over-segmented image.

### 4.1.2 Classifiers Building

To build the discard and the merge classifiers it was used the CART for Windows software, version 4.0. The chosen options were the standard ones: Gini as the splitting method, the same probability for both classes, 10-fold cross validation as method for testing tree, the best suggested tree is the minimum cost one, regardless the tree size. The minimum node size was changed, so the parent nodes must have at least 2 cases and the terminal nodes must have at least one case.

#### 4.1.2.1 Discard Classifier

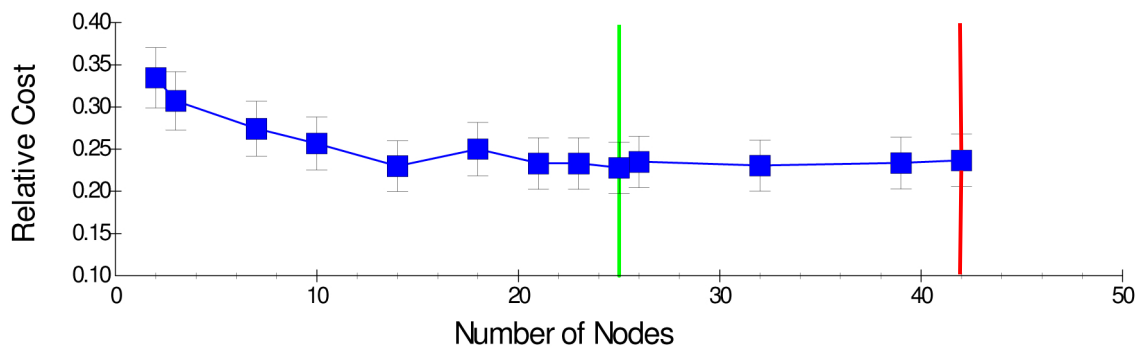
To build the discard classifier training set, 6 different over-segmented images were used and 50 "to discard" instances and 50 "to keep" instances from each image were selected, making a total of 600 instances. To each instance the following features were saved:

- Area (A);
- Perimeter (P);
- Shape factor given by  $\text{Perimeter}^2 / (4 * \pi * \text{Area})$  (S);
- Histogram width (HW);
- Variance (V);
- Ratio of the contour intensity over the inside intensity (R)

The choice of these attributes can be justified by closely observing the regions of the over-segmented image. As we can see from Figure 4.4, the area, for example, should be an important attribute once the background regions are significantly bigger than the cell regions. This aspect is reflected also in regions perimeter and that is why this feature is also considered. The shape factor is a measure of the circularity of the regions, being 1 to circles. This is an important feature for two reasons: first, the background regions have a shape factor much bigger than cell regions. The other reason is that by using this feature, air bubbles (which are approximately circles having a shape factor close to one) on images can be discarded since most of the cell regions are not circle shaped. Like it was stated in Section 3.1, histogram width is a measure of the region contrast. Typically, cell regions have more contrast than background regions, which are much uniform. Other measure of this uniformity is the regions variance. This attribute reflects the fact that region cells are much less uniform than background regions. This feature has shown to be the most important one. For a given training set, CART shows a graph of the variables importance, as shown in Figure 4.5. The ratio of contour intensity over the inside intensity of regions was used because the cell regions have a brighter contour comparing to their inside zone while background regions don't.

Variable	Score	
V	100.00	
R	70.18	
A	68.43	
S	60.32	
P	58.05	
HW	54.19	

**Figure 4.5:** Variables from discard classifier importance values, given by CART.



**Figure 4.6:** Discard trees set suggested by CART to the discard classifier.

All instance attributes were saved in *.txt* file and then converted to *Excel 2003-2007* in order to be read by the CART software.

The proposed set of trees by CART are shown in Figure 4.6, where the smaller cost tree is marked with a vertical green line and the chosen tree is marked with a vertical red line. The suggested tree was not chosen because, after the classifier construction, this last tree showed to perform better than the best cost tree.

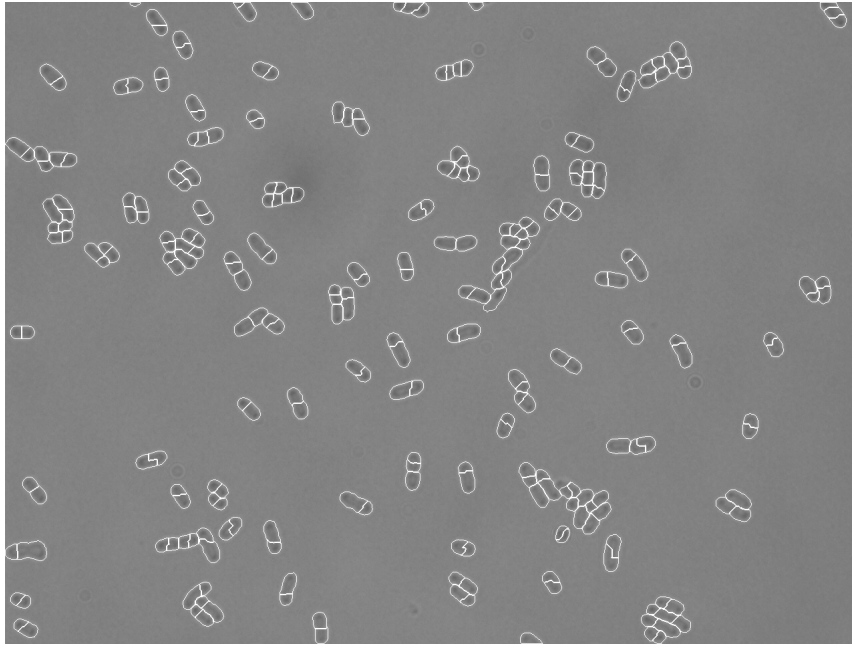
The chosen tree structure is shown in Figure 4.7(a), in which, as mentioned in Section 2.5.2.1, terminal nodes are color coded to reflect the relative probability of response: a red node is above average and a blue node is below average in response probability. In Figure 4.7(b), it can be seen the splitters of the decision tree used to build the discard classifier.

#### 4.1.2.2 Merge Classifier

At this point the over-segmented image only had the segments of interest (see Figure 4.8). This way, it was necessary to merge them to form cells instead of cell segments. For that, a second classifier to decide which segments were to be merged and which were to be kept apart was built.

Like for the discard classifier, to build the merge classifier 6 different over-segmented images were used (after a discard action by the discard classifier) and 50 "to merge" instances and 50 "do not merge" instances were selected.





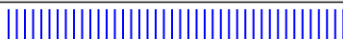



**Figure 4.8:** Over-segmented image without background segments.

Similarly to the building process of the discard classifier, a training set with the following attributes was created:

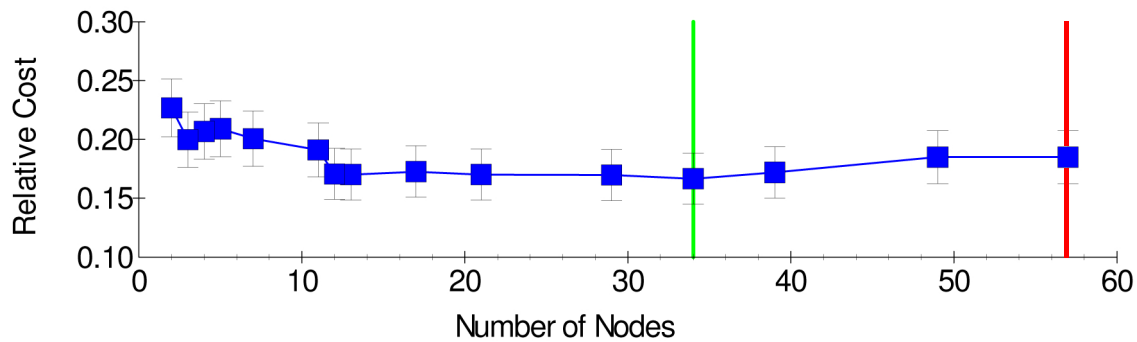
- Shape factor of the two segments analysed together (S).
- Variance of the image in the contact area between the two segments (V).
- Length of the contact zone between the two segments (CL).
- Ratio between intensity of the image in the contact zone and the intensity of the image in the contour of the segments (R).

The shape factor was used again because, as it can be easily seen from Figures 1.1(a) and 4.8, cells have a relatively regular elliptical shape. This way, computing the shape factor of the two segments to merge (or not) is an important attribute. As it was stated before, the regions belonging to a cell have a bigger variance. This way, evaluating the variance of the contact zone between two segments can be a good indicator if those segments are to be merged or not. So, two segments which barely touch each other, probably are not to be merged, and thus, the length of the contact zone between segments is also taken into account. It was also mentioned before that cells have a brighter contour, comparing to the inside zones. This way, the ratio between the two segments contact zone and its contour when aggregated, can be a decisive attribute. In fact, this last attribute is the most important one, as can be seen in Figure 4.9. Once more, all this instances attributes values were saved in a *.txt* file and then converted to *Excel 2003-2007*.

After these steps, the data was loaded into CART which suggested the set of trees shown in Figure 4.10.

Variable	Score	
R	100.00	
S	65.27	
V	58.33	
CL	31.68	

**Figure 4.9:** The given importance values by CART to the variables from the merge classifier.



**Figure 4.10:** Trees set suggested by CART to build the merge classifier.

The chosen tree was the last one of the set of trees suggested by CART, for the same reasons mentioned when the discard classifier was previously discussed. Its structure and splitters are shown in Figure 4.11.

## 4.2 Errors Correction

With the employment of the constructed classifiers, some errors might arise which can be solved comparing each image with the previous and the next ones.

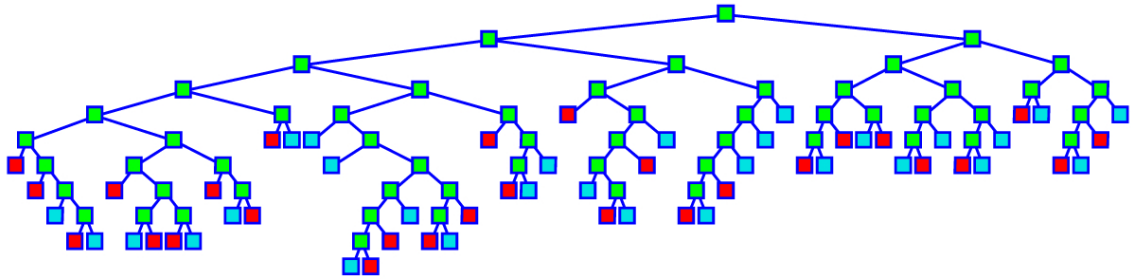
### 4.2.1 Discard Errors

It were considered two different types of discard errors: a cell segment that was incorrectly discarded (error type 1) or a background segment that was accepted as a cell segment (error type 2).

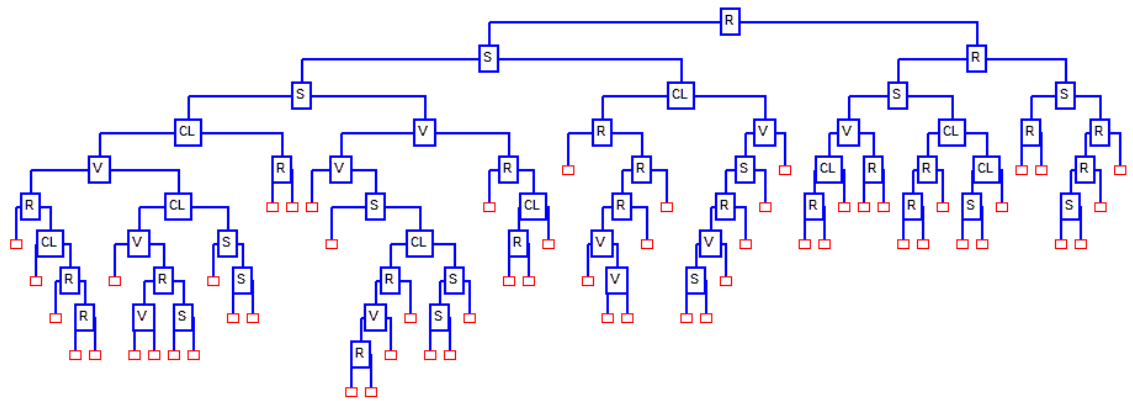
An example of type 1 discard error is shown in Figure 4.12. This type of error can usually be solved by analysing each discarded segment.

It was considered that a cell segment has correspondence in the previous or the next image if, by overlapping both images (current and next or previous images), that segment intersects in more than 50% of its area on any cell of the other image. It is important that this percentage of interception is not too large because sometimes little misalignments occur between consecutive images.

This way, if one segment that was discarded has correspondence in the previous and in the next image, it was probably misclassified and by recovering that segment, the problem is solved.

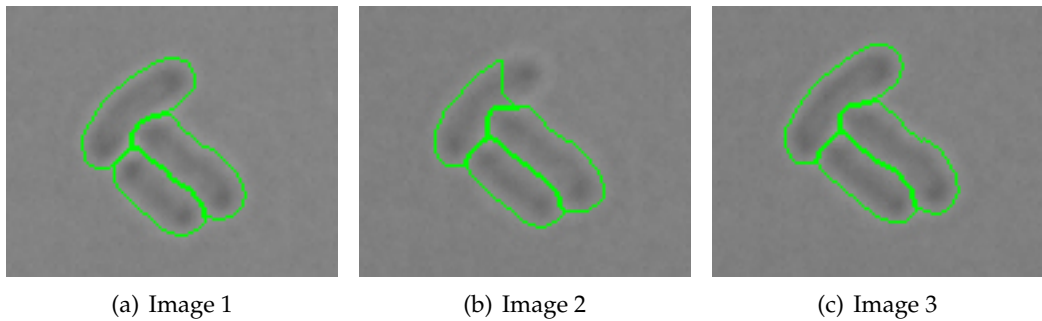


(a) Merge classifier tree structure, where the decision nodes are represented in green. The blue terminal nodes represent the class *not merge* and in red is the class *merge*.



(b) Merge classifier tree splitters.

**Figure 4.11:** Merge tree used to build the discard classifier.



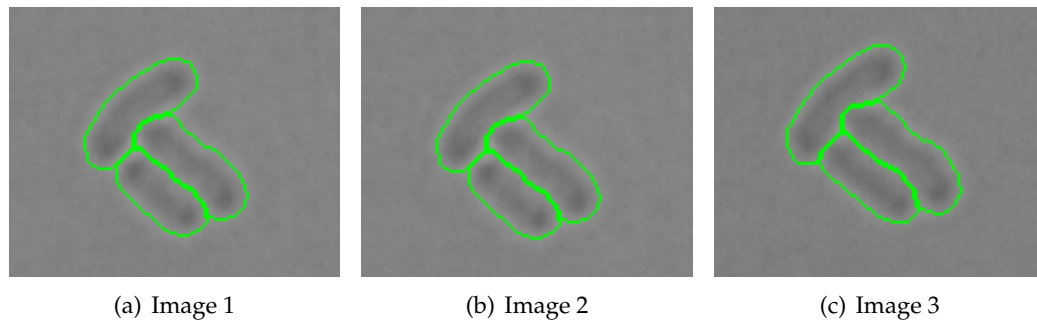
(a) Image 1

(b) Image 2

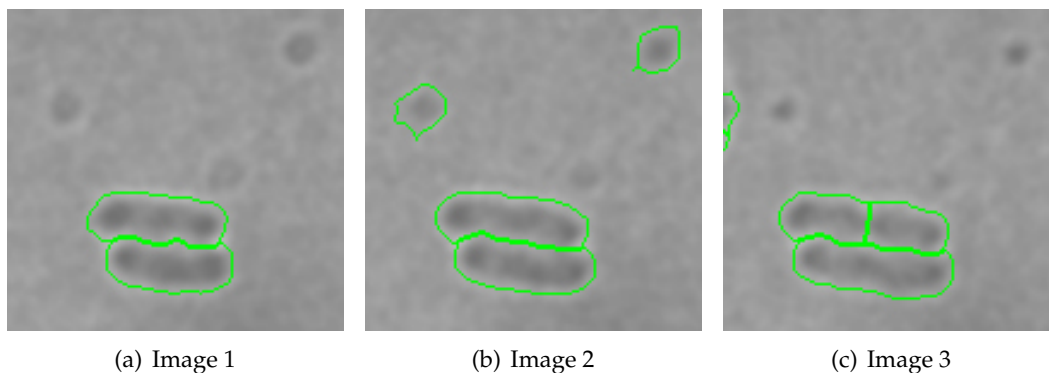
(c) Image 3

**Figure 4.12:** A discard type 1 error.

The result after this correction is shown in Figure 4.13.



**Figure 4.13:** A discard type 1 error solved.



**Figure 4.14:** A discard type 2 error.

The other type of discard error is shown on Figure 4.14. This type of error can be solved similarly to the error previously described but instead of searching for segments which were wrongly discarded, the the algorithm searches for segments that were accepted but do not have correspondence in the previous and in the next image.

The first image of each time series is compared with the next two images and the last image is compared with the two previous ones.

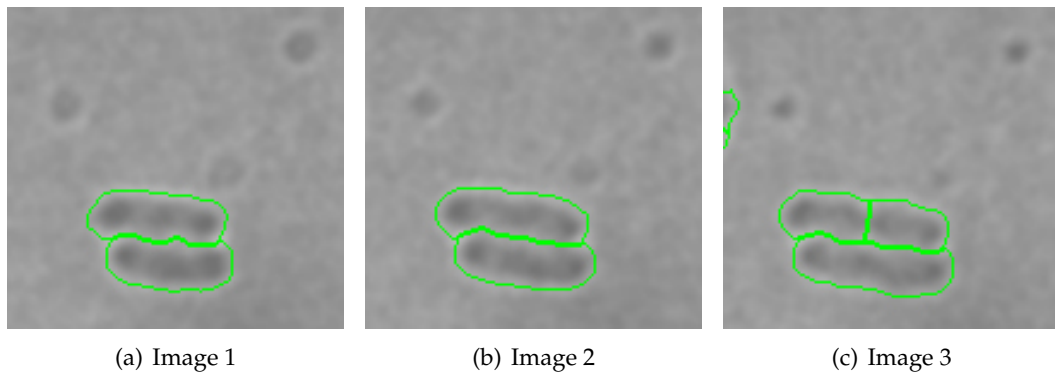
### 4.2.2 Merge Errors

Merge errors can also be separated into two types of error. The first type occurs when two segments that do not belong to the same cell are merged. The other type of error occurs when two segments of one cell are not merged. Some of those errors can be solved by comparing the current image with the previous one.

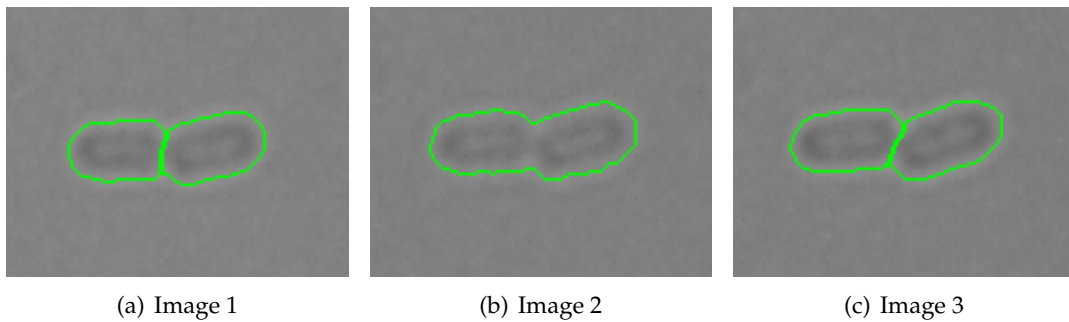
In order to find these errors, each image is compared with the previous one (except for the first image). If one cell of the current image has more than one correspondent in the previous image, then, probably is a merge error. That situation occur in scenarios similar to the ones illustrated in Figures 4.16(b) and 4.18(c).

Those scenarios might correspond to two different situations: the current cell was





**Figure 4.15:** A discard type 2 error solved.

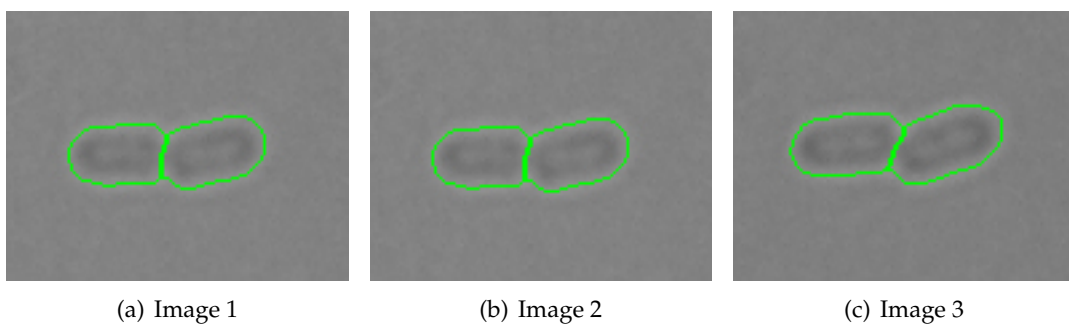


**Figure 4.16:** A merge type 1 error.

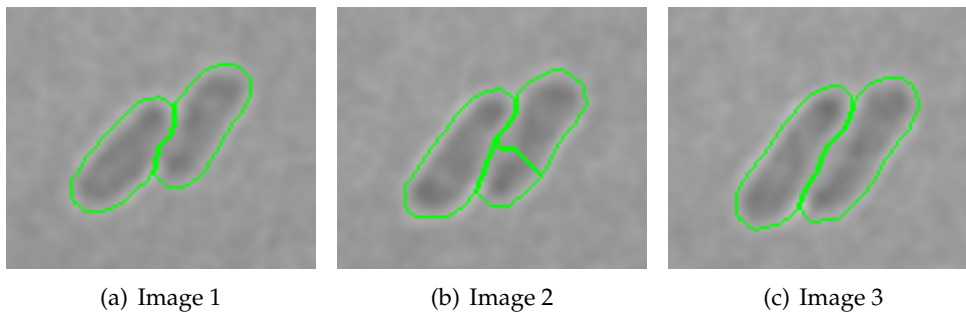
incorrectly merged (error type 1), as show in Figure 4.16(b), or the correspondences were incorrectly left separated (error type 2), as shown in Figure 4.18(c).

There are two ways to correct the first type of errors. The first one is to join the segments of the previous image and the other is to split the cell of the current image into two segments.

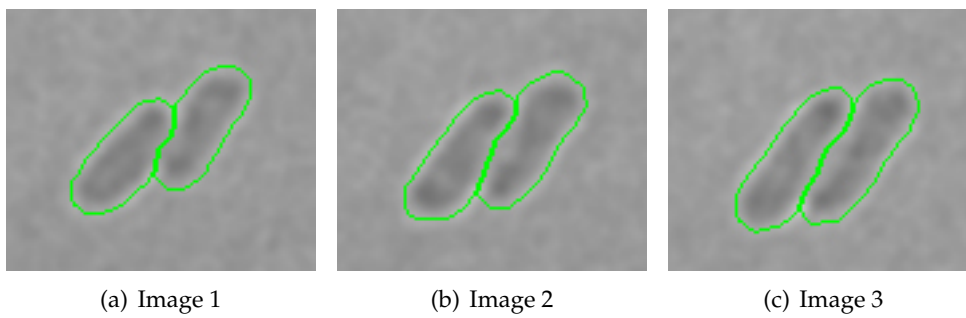
Consider that  $a_p$  is the mean cell areas of the previous image and  $a_c$  is the mean cell areas of the current image.  $a_{ps1}, a_{ps2}, \dots, a_{psn}$  represent the areas of the  $n$  cell segments of the previous image that intersect the cell under analysis of the current image, which has an area of  $a_{c1}$ .



**Figure 4.17:** A merge type 1 error solved.



**Figure 4.18:** A merge type 2 error.

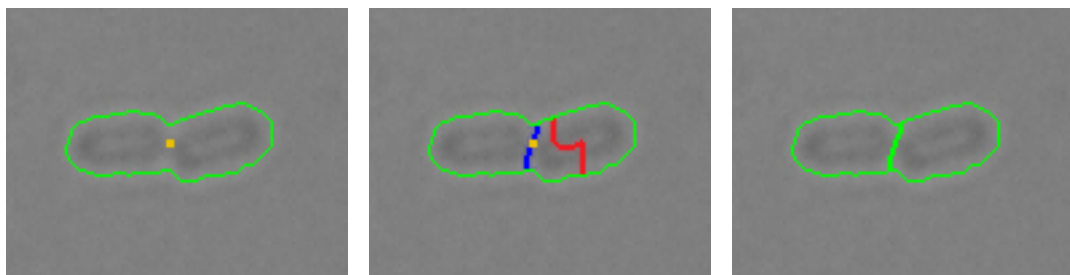


**Figure 4.19:** A merge type 2 error solved.

Considering the two previously mentioned hypotheses, the first one is applied if:

$$|a_p - \sum a_{ps1}, a_{ps2}, \dots, a_{psn}| < |a_c - \frac{a_{c1}}{2}|$$

Otherwise, the second hypothesis is applied. In this case, the cell is divided by the contact region whose mean distance of the pixel to the centroid of the cell is lower, as shown in Figure 4.20, which represents the situation of Figure 4.16.



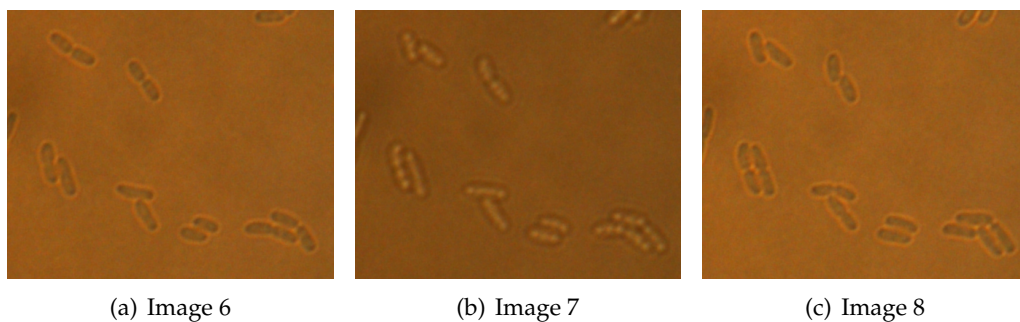
(a) Merge error and cell centroid in yellow. (b) The two possible cell division zones in red and blue. (c) The chosen of the division by the closer zone to the centroid.

**Figure 4.20:** Illustration of the used cell division criteria.

### 4.2.3 Inverted Images

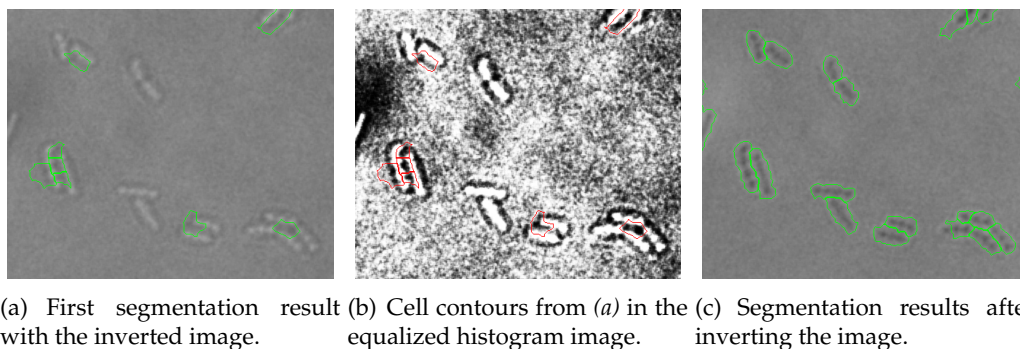
In some of the time series, there were images that were inverted relating to the other images, as shown in the example of Figure 4.21. These inversions were due to the focusing problems related to the image acquisition process. When this takes place, the GPL detects objects that do not correspond to cells, which are darker in a brighter background in images used to built the classifier. The GPL result obtained with the example in Figure 4.21(b) is shown in Figure 4.22(a).

To ascertain if an image is inverted, it is calculated the mode of the pixels belonging to cells, in an image with equalized histogram (see Figure 4.22(b)). If it is 255 (white), an inverted image is observed, with cells brighter in a darker background, as in Figure 4.22(a).



**Figure 4.21:** Inverted time series image.

This way, the image is inverted and the analysis is done again. The example shown in Figure 4.22(a) results in Figure 4.22(c) after the image is inverted and analysed again.



**Figure 4.22:** GPL results of an inverted time series image.

## 4.3 Cells Tracking

Once the cells are detected, it is important to track them over time. This is important in order to know when a cell divides and how it evolves over time.

The first step of the cell tracking algorithm was, was to identify the corresponding cells of each image on the previous one.

As it was mentioned before, a cell in an image  $i$  corresponds to a cell in the next image  $i+1$  if the area of their intersection is bigger than 50% in one of the two cells. Each corresponding cell on a previous image will be called from now on as *parent* or *ascendant*.

Once identified each parent cell, if two different cells have the same parent it means that a division occurred.

Thus, a label is attributed to each cell in the first image. Then, beginning in the second image, each cell will receive the label of its parent if a division has not occurred. Otherwise, each "sister-cell" receives a new different label. Repeating this process to all images of the time series, one has each life of one cell labelled with a specific label.

Assigning a color to each label, we can visualize the evolution of the cells of a time series, like in Figure 4.23.

When a cell divides, its descendants are coloured with a slightly different tone in order to better follow the daughters of a cell. After this step, the DIC image should be correctly segmented.

## 4.4 Images Alignment

The previously described algorithm allows us to know where each cell is in a certain image. After that, it is necessary to align the DIC image with its correspondent fluorescence image.

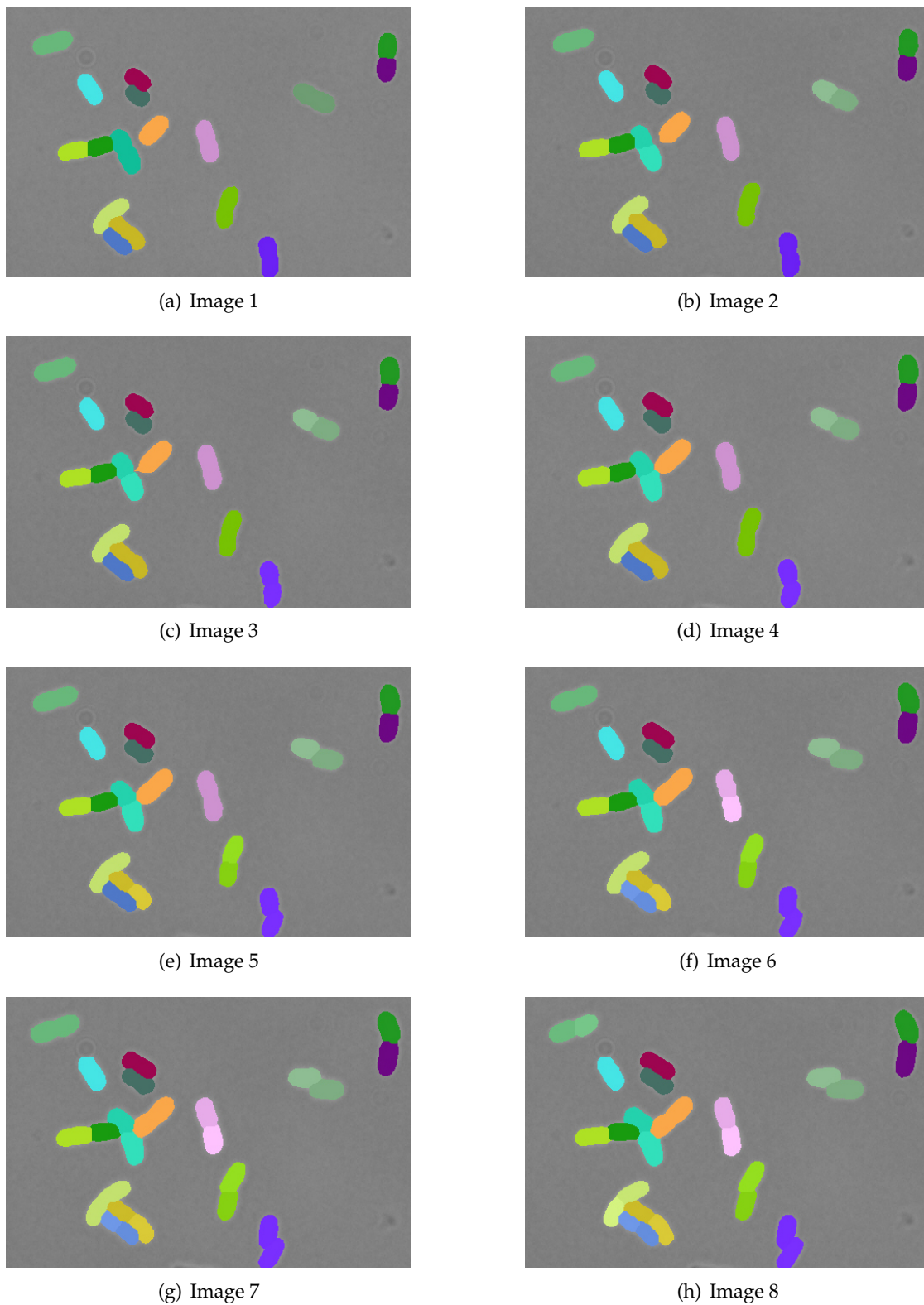
To accomplish that a method that looks for the relative position between DIC and fluorescence images that maximizes the "fluorescence dots" within cells contours was developed. In order to accomplish this, an *exhaustive search* of the parameters angle, scale and relative position of the images and counting, to each position, the fluorescence dots within cells contours was performed.

To enable fluorescence visualization on confocal images, it was necessary to apply a threshold since fluorescence confocal images are usually very dark. In order to obtain a better distinction between the fluorescence and the background, the fluorescence image was thresholded using the Otsu method. Once the image was converted into a binary one, the fluorescence dots previously mentioned correspond to the black pixels. An example of a resulting confocal fluorescence image is shown in Figure 4.26(c). Furthermore, it was applied a  $5 \times 5$  median filter to better distinguish cell zones from background. An example is shown in Figure 4.26(d).

This procedure is only for visualization and position searching proposes, once the fluorescence quantification (explained in Section 4.5) will be done in the original confocal image.

As it can be seen in Figure 4.25, the DIC image only covers a part of the fluorescence image. The relative position, scale and angle between the two types of images is not fixed. Therefore, each time series must be analysed individually.

In order to align the DIC and the CFM images, two methods were developed: the *automatic alignment* and the *semi-automatic alignment*.



**Figure 4.23:** Cell tracking example.

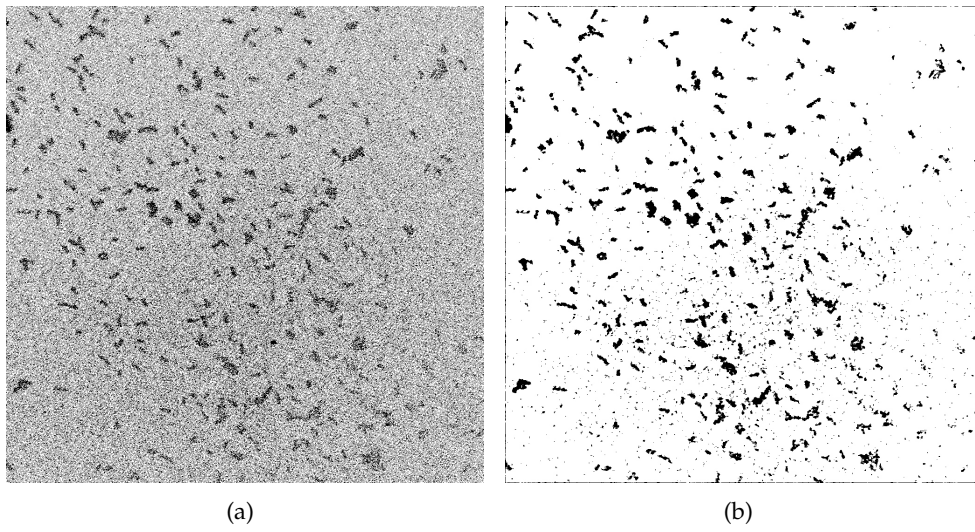


Figure 4.24: Otsu threshold (a) and median filter (b) appliance to the fluorescence image.

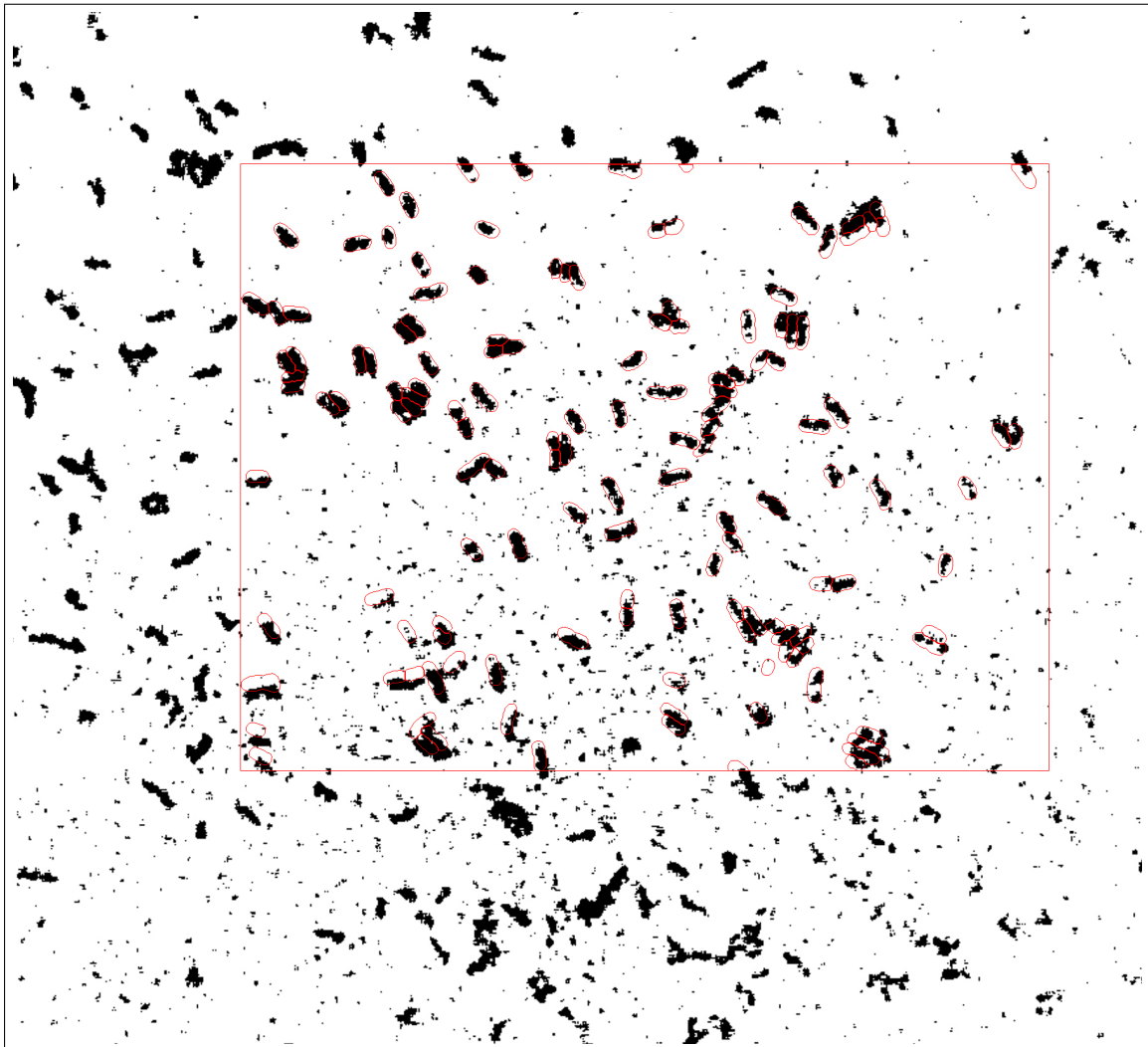


Figure 4.25: Example of the alignment between the DIC (the one bounded by the red square) and the correspondent CFM images.

### 4.4.1 Automatic Alignment

The automatic alignment doesn't require any intervention from the user.

In order to maximize the performance of this method, a two phase method was developed: first, the search for an approximate position by bigger steps and once that position is found, the search in its surroundings for the best position that maximizes the fluorescence within cells contours.

Images can vary in relative scale and position. In a small number of images the rotation also changed and therefore the search was done for both angle 0 and 180 degrees.

Let  $fl_y$  be the fluorescence image height and  $dic_y$  the DIC image height.

For the first phase of this method, the scale ranging is built as

$$\min \left( 2.5 \frac{dic_y}{fl_y}, \frac{dic_y}{fl_y} \right) \leq \text{biggerScale} \leq \max \left( 2.5 \frac{dic_y}{fl_y}, \frac{dic_y}{fl_y} \right)$$

in 0.1 steps.

Considering that the position  $(x, y)$  relates to the upper-left corner of the DIC image and that  $fl_x$  and  $dic_x$  are the widths from the CFM and DIC images, respectively, we obtain:

$$1 \leq x \leq \max(1, fl_x - dic_x)$$

and

$$1 \leq y \leq \max(1, fl_y - dic_y).$$

The steps in the  $x$  and  $y$  directions depend on the scale already applied to the CFM image and they divide it into a maximum of 150 columns and 150 lines, respectively. The steps are therefore given by  $\text{step}_x = fl_x \lceil fl_x / 150 \rceil$  and  $\text{step}_y = \lceil fl_y / 150 \rceil$ .

These limits are illustrated in Figure 4.26.

After finding the best set of values that maximize the fluorescence within cells contours, the best fit in the surroundings of the previously found position is searched. On this second phase of the process, considering  $\text{scale}_0$ ,  $x_0$  and  $y_0$  the parameters already found, the ranges of the new search was done by

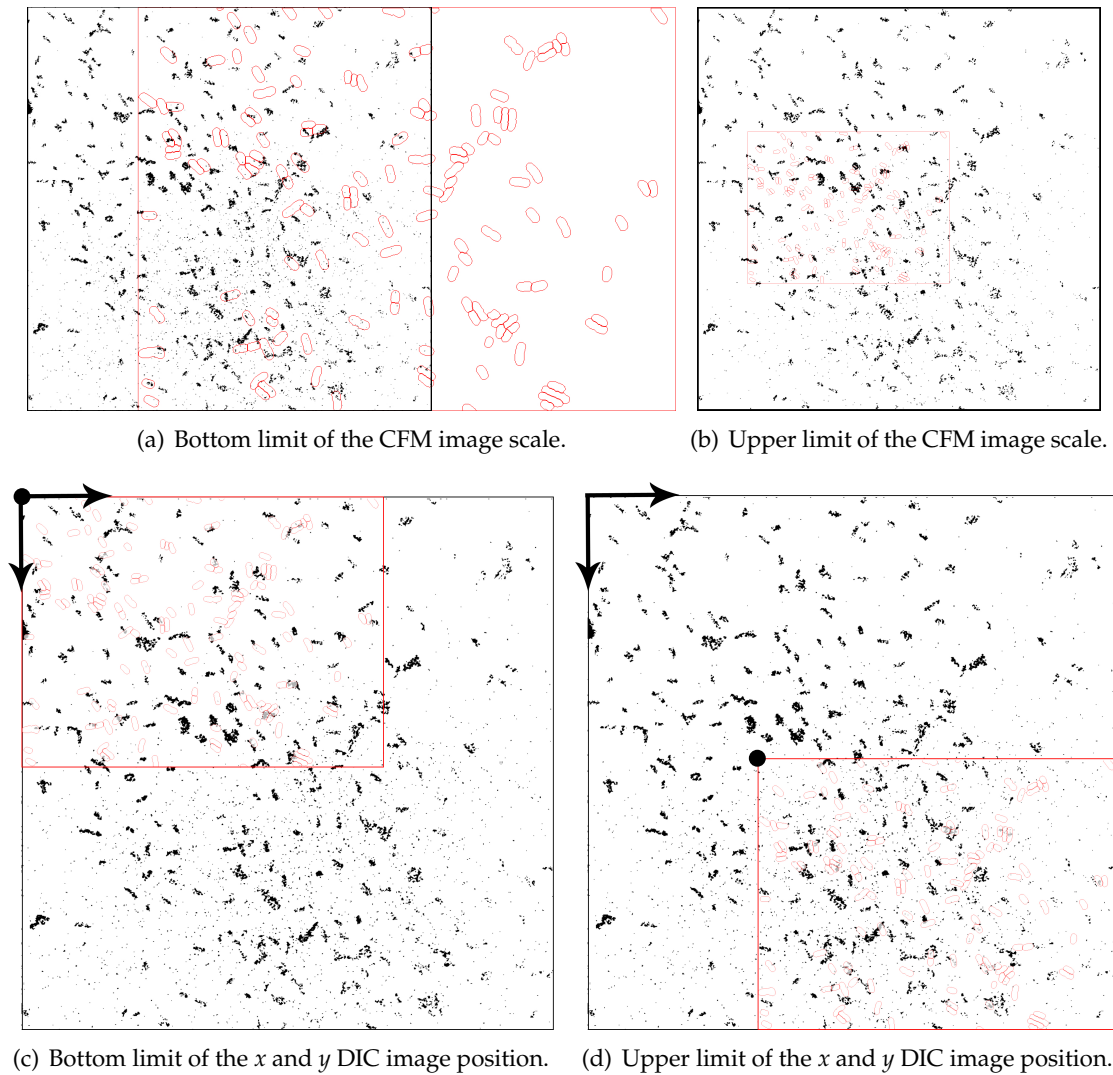
$$\text{scale}_0 - 0.1 \leq \text{scale}_0 \leq \text{scale}_0 + 0.1$$

in 0.01 steps and

$$x_0 - \text{step}_x \leq x_0 \leq x_0 + \text{step}_x$$

and

$$y_0 - \text{step}_y \leq y_0 \leq y_0 + \text{step}_y$$



**Figure 4.26:** Limits of exhaustive-search parameters.

in unitary steps.

#### 4.4.2 Semi-Automatic Alignment

This method differs from the one mentioned above because the first position is given by the user. Only the second phase is computed, using smaller steps to find the best position around the one that was given by the user.

The user has at his disposal two different sliders that allows him to range the scale and angle between the two images. Clicking in the image, he can change the  $y$  and  $x$  positions.

In the final stage of the process an image like the one in Figure 4.27 is obtained.



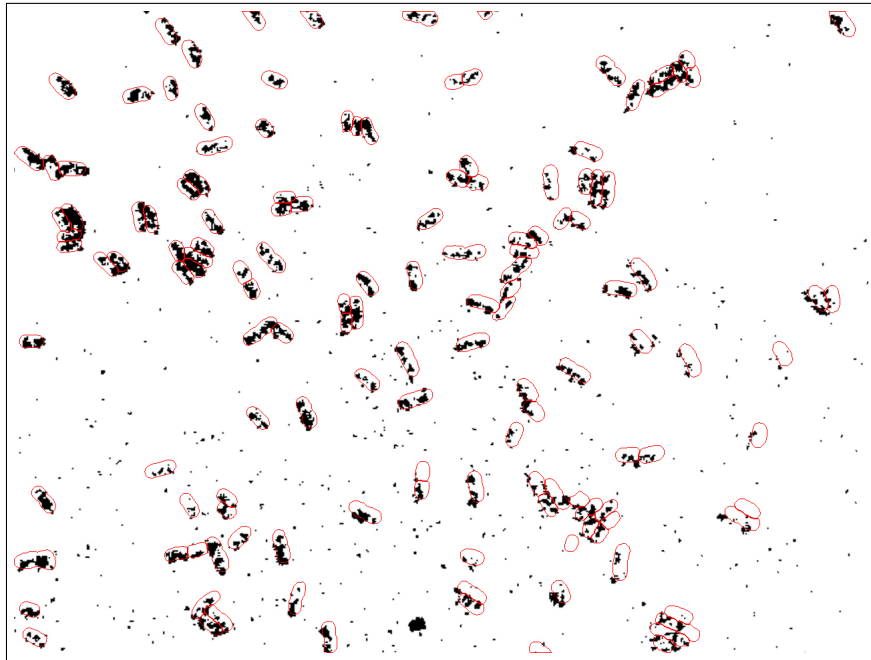


Figure 4.27: An example of the DIC image aligned with the correspondent fluoresce image.

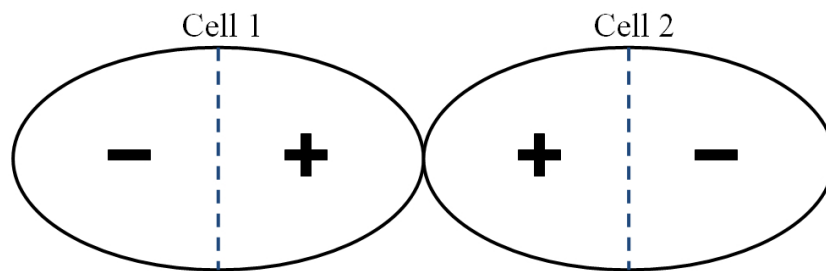


Figure 4.28: Cell positive and negative pole location.

## 4.5 Fluorescence Quantification

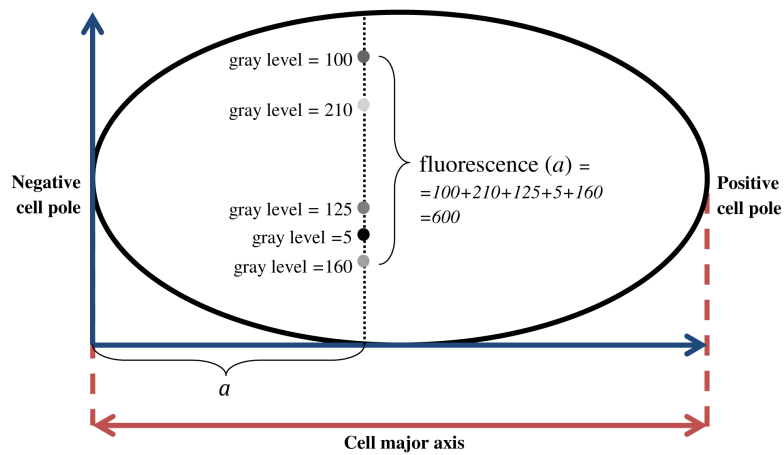
Fluorescence quantification is done over cells major axis, in the direction of their younger pole. The younger pole (or positive pole) of a cell is the pole created after the division that originated it, as shown on both "sister" cells of Figure 4.28.

The fluorescence, for each pixel of the major axis, is the sum of the gray intensity of the confocal image in its vertical direction until the cell contours are reached as shown in Figure 4.29.

### 4.5.1 Poles Determination

To compute cells poles it is necessary to distinguish between both cell poles: the pole which is closer to the division point and the one which is further.

Consider the example of the cell just divided in two "sister" cells, as shown in Figure 4.30(a). Observing just the upper cell, shown in Figure 4.30(b), the new pole of the



**Figure 4.29:** Fluorescence quantification along the major axis of cells, in negative-positive pole direction.

cell will be the lower one. The cell is then rotated to orientate that pole to the right and to align its major axis with the horizontal axis. Furthermore, to allow comparisons of relative fluorescence distribution between cells, their size is normalized to 50 pixels, as in the cell represented on Figure 4.30. For both rotation and resizing the bi-cubic interpolation was used.

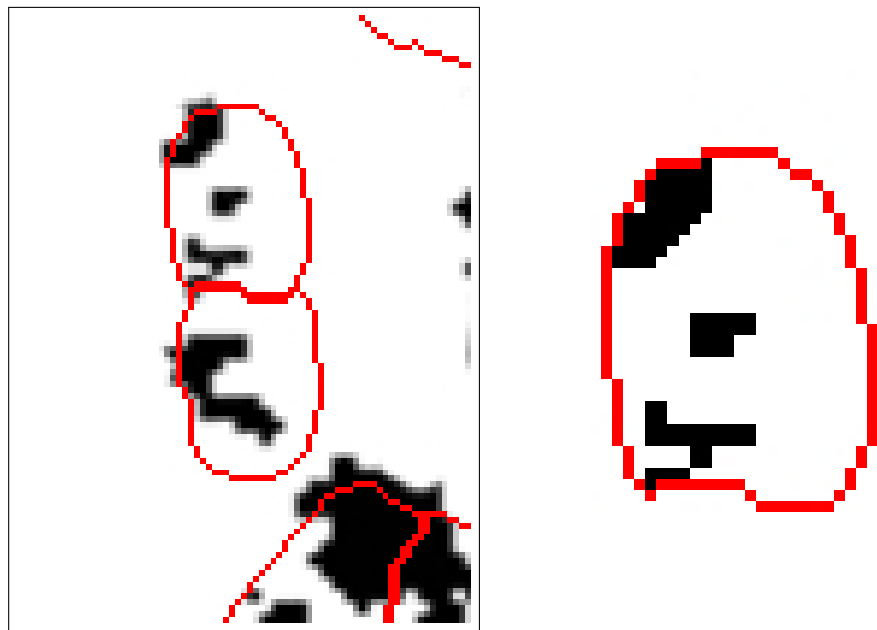
For each cell fluorescence quantification it was saved an entry in a *.txt* file with the format `0, 0, 0, 1, 2, 3, 4, 4, 6, 7, 9, 8, 7, 7, 5, 4, 2, 0, 0, . . .`, corresponding each number to the vertical gray intensity of one of the 50 pixels from the cell major axis. This allows the data to be analysed externally, e.g., in *Excel*.

The scheme of the fluorescence quantification algorithm is shown in Figure 4.31.

## 4.6 Interface

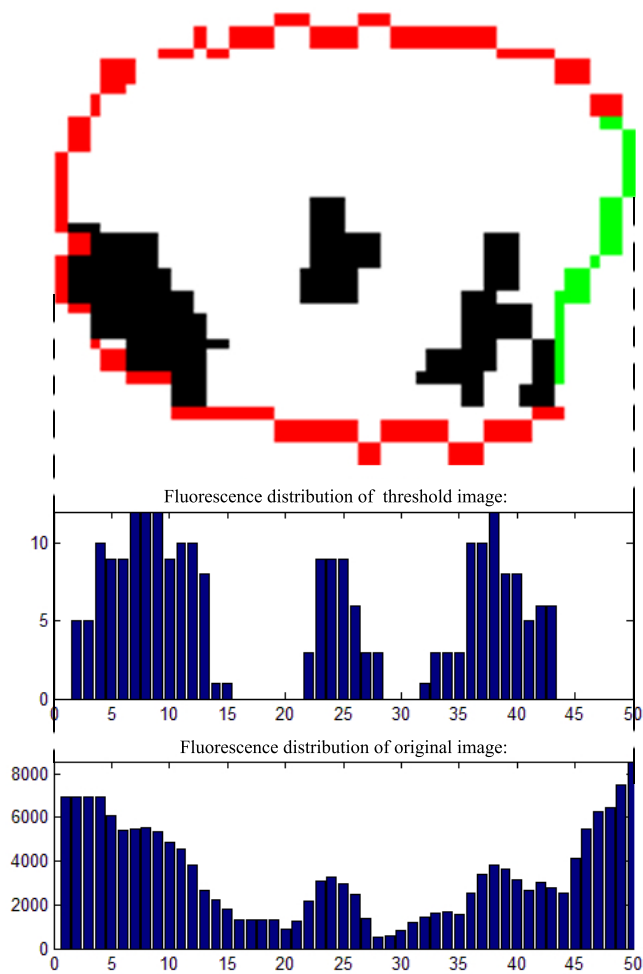
In Figure 4.32 the developed interface is shown, where the marked buttons and sliders have the following functions:

- 1 Load images (single shots or time series) from the disc;
- 2 Run the cells detection algorithm;
- 3 Choose one of the following view modes: *Original Images*, *Cells Contours*, *Colored Cells* or, after the fluorescence quantification, *Fluorescence*.
- 4 Align the images using the semi-automatic mode;
- 5 Align the images using the automatic mode;
- 6 Manually adjust the CFM image scale;
- 7 Manually adjust the CFM image angle;



(a) Two "sisters" cells.

(b) The upper cell from (a).



(c) Cell from (b) rotated, resized to 50 pixels of major axis, with the positive pole in green and correspondent fluorescence quantification distribution.

**Figure 4.30:** Fluorescence quantification process.

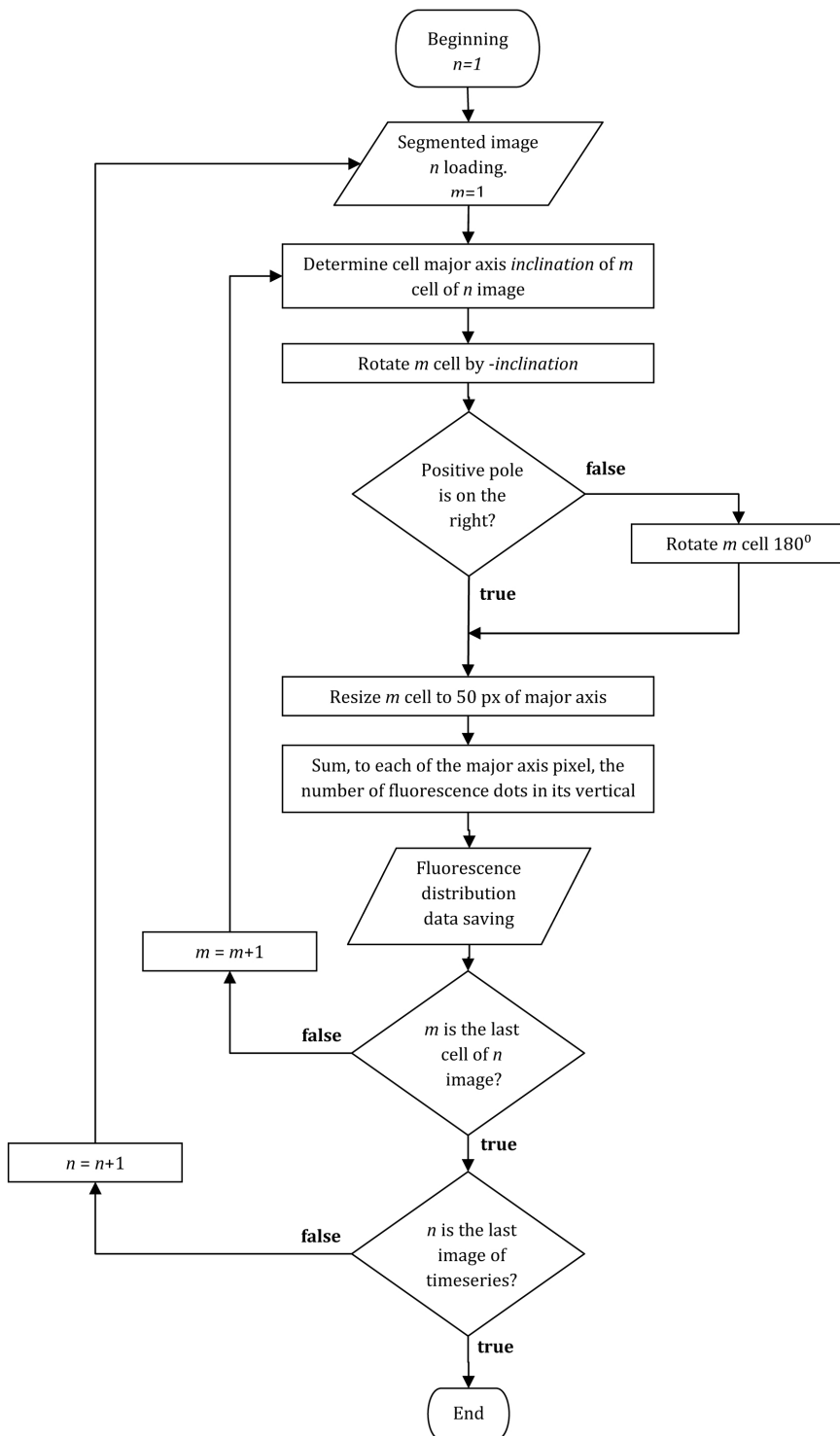


Figure 4.31: Fluorescence quantification algorithm scheme.



Figure 4.32: Developed interface.

- 8 Accept the current scale and angle values and start small adjustments around it;
- 9 Save current image;
- 10 If the alignment and fluorescence quantification are already done, the resulting fluorescence distribution data is exported;
- 11 Detach the current image to allow the use of tools such as zooming;
- 12 Navigate through the previous time series image.
- 13 Navigate through the next time series image.

Figure 4.33 shows an image of the developed interface during the cells detection phase.

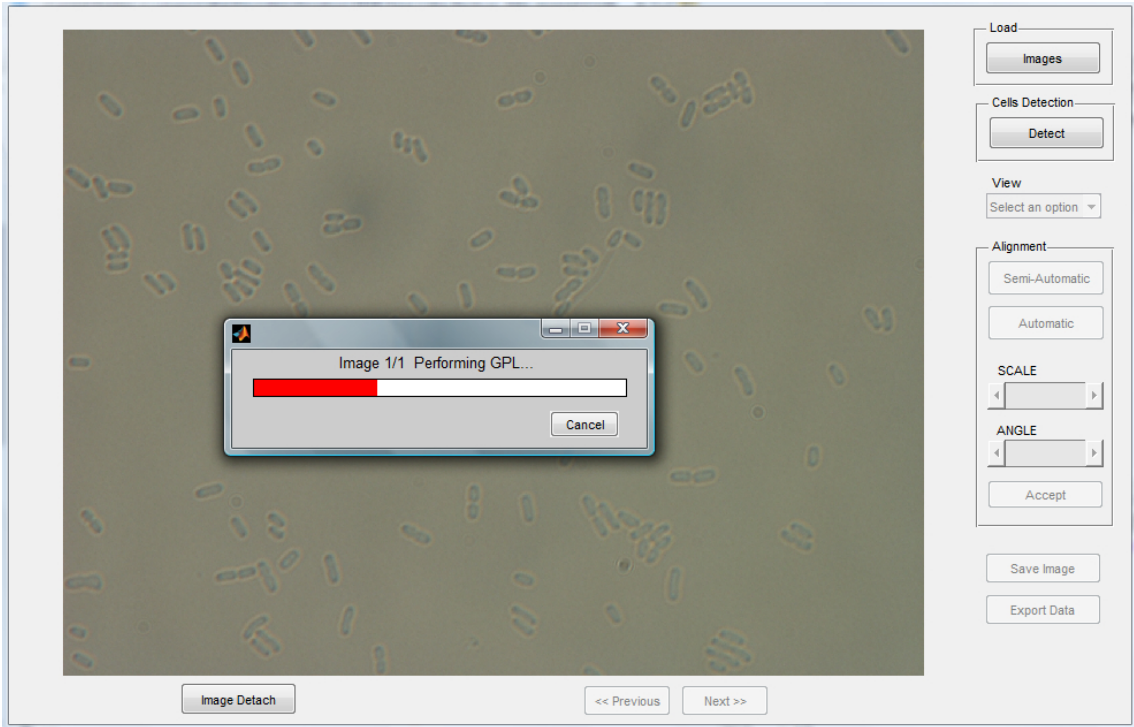


Figure 4.33: Developed interface showing a waiting bar while cells detection is running.



# Results and Discussion

## 5.1 Results

As it was mentioned in Section 2.4.1, machine learning algorithms can calculate the error rate by different methods. Using *n-fold* cross-validation (with  $n = 10$  in this case), CART predicted a success rate of 90,14% and 90,87% for the discard classifier and the merge classifier, respectively.

The classifiers were built from 6 different images, using 600 examples each. Those 6 images were chosen from 5 time series, with 13 images each. In this chapter the results of the classifiers on all the 65 available images are shown.

The classifiers, segmentation and tracking processes error rate at each one of the time series are shown in Tables 5.1 to 5.5. The columns in gray refer to images which were inverted and DD, DWD and DER refers to the number of decisions, wrong decisions and error rate of the discard classifier, respectively. Likewise, MD, MWD and MER refers to the number of the decisions, wrong classifications and error rate of the merge classifier, respectively.

The classifier error rates were calculated by the ratio of wrong decisions over the total number of decisions. The segmentation error rate (SER) was computed by the number of cells that the segmentation algorithm detected (SNC) over the real number of cells (RNC) in the image.

The tracking algorithm error rate (TER) was calculated by the ratio of the number of cells in the first image that were correctly segmented and followed (and all its descendants) until the last time series image over the total number of cells in the first image.

Summarizing and averaging the classifiers error rates, the discard classifier performed

**Table 5.1:** Time series n.1 error rates

Image n.	1	2	3	4	5	6	7	8	9	10	11	12	13
<b>DD</b>	293	298	313	310	322	344	329	362	415	426	433	415	495
<b>DWD</b>	2	1	2	3	6	8	3	2	3	8	4	7	5
<b>DER (%)</b>	0.68	0.34	0.64	0.97	1.86	2.33	0.91	0.55	0.72	1.88	0.92	1.69	1.01
<b>MD</b>	273	285	307	299	318	349	345	392	494	484	528	528	686
<b>MWD</b>	0	3	1	2	3	4	6	5	8	10	9	20	9
<b>MER (%)</b>	0.00	1.05	0.33	0.67	0.94	1.15	1.74	1.28	1.62	2.07	1.70	3.79	1.31
<b>RNC</b>	141	145	150	154	162	174	182	193	221	228	237	257	288
<b>SNC</b>	142	147	154	160	167	181	188	200	225	235	249	267	297
<b>SER (%)</b>	0.71	1.38	2.67	3.90	2.09	4.02	3.30	3.63	1.81	3.07	5.06	3.89	3.13

**DER mean = 1.12 % ± 0.61%**

**MER mean = 1.36% ± 0.92%**

**SER mean = 3.05% ± 1.18%**

**TER=29/142=20,42%**

**Table 5.2:** Timeseries n.2 error rates

Image n.	1	2	3	4	5	6	7	8	9	10	11	12	13
<b>DD</b>	700	786	837	837	847	847	868	916	797	829	860	983	836
<b>DWD</b>	16	21	28	29	28	20	22	27	24	24	30	31	40
<b>DER (%)</b>	2.29	2.67	3.35	3.46	3.31	2.36	2.53	2.95	3.01	2.90	3.49	3.15	4.78
<b>MD</b>	78	92	95	112	147	171	176	206	214	231	334	402	297
<b>MWD</b>	3	5	3	3	4	6	6	3	12	6	8	15	5
<b>MER (%)</b>	63.85	5.43	3.16	2.68	2.72	3.51	3.41	1.46	5.61	2.60	2.40	3.73	1.68
<b>RNC</b>	61	65	72	78	79	88	95	105	108	129	148	144	141
<b>SNC</b>	63	69	76	81	84	91	100	113	120	135	159	158	155
<b>SER (%)</b>	3.28	6.15	5.56	3.85	6.33	3.41	5.26	7.62	11.11	4.65	7.43	9.72	9.93

**DER mean = 3.10% ± 0.65%**

**MER mean = 3.25% ± 1.24%**

**SER mean = 6.48% ± 2.56%**

**TER=25/63=39,68%**

**Table 5.3:** Time series n.3 error rates

Image n.	1	2	3	4	5	6	7	8	9	10	11	12	13
<b>DD</b>	851	782	813	695	807	802	818	809	840	888	781	834	964
<b>DWD</b>	11	18	18	19	23	13	26	29	16	25	17	21	49
<b>DER (%)</b>	1.29	2.30	2.21	2.73	2.85	1.62	3.18	3.58	1.90	2.82	2.18	2.52	5.08
<b>MD</b>	59	78	111	91	135	140	137	187	278	325	339	428	364
<b>MWD</b>	3	5	3	3	4	6	6	3	12	6	8	15	5
<b>MER (%)</b>	5.08	6.41	2.70	3.30	2.96	4.38	3.21	1.08	3.69	1.85	2.36	3.50	1.37
<b>RNC</b>	39	57	59	65	68	66	76	91	105	105	122	154	158
<b>SNC</b>	46	61	64	68	74	72	84	94	112	118	129	158	167
<b>SER (%)</b>	17.95	7.02	8.47	4.62	8.82	9.09	10.53	3.30	6.67	12.38	5.47	2.60	5.70

**DER mean = 2.64% ± 0.96%**

**MER mean = 3.22% ± 1.48%**

**SER mean = 7.91% ± 4.11%**

**TER=28/46=60,87%**



**Table 5.4:** Time series n.4 error rates

Image n.	1	2	3	4	5	6	7	8	9	10	11	12	13
<b>DD</b>	1917	1286	439	665	557	1131	849	553	1535	435	589	605	444
<b>DWD</b>	6	4	2	5	2	60	21	5	24	7	8	9	5
<b>DER (%)</b>	0.31	0.31	0.46	0.75	0.36	0.53	2.47	0.90	1.56	1.61	1.36	1.13	0
<b>MD</b>	39	42	41	25	59	92	158	136	154	145	222	260	133
<b>MWD</b>	3	3	4	3	4	3	6	4	5	1	10	12	10
<b>MER (%)</b>	7.69	7.14	9.76	6.00	6.78	3.26	3.80	2.94	3.25	0.69	4.50	4.62	7.52
<b>RNC</b>	28	31	29	31	36	51	66	68	77	75	88	105	114
<b>SNC</b>	31	33	34	35	39	53	71	72	82	82	94	107	123
<b>SER (%)</b>	10.71	6.45	17.24	12.90	8.33	3.92	7.58	5.88	6.49	9.33	6.82	2.86	7.89

**DER mean = 1.02% ± 0.66%**

**MER mean = 5.23% ± 2.50%**

**SER mean = 8.19% ± 3.79%**

**TER=10/31=32,26%**

**Table 5.5:** Time series n.5 error rates

Image n.	1	2	3	4	5	6	7	8	9	10	11	12	13
<b>DD</b>	2286	641	1231	611	655	635	702	1479	743	645	659	712	862
<b>DWD</b>	31	3	15	14	2	4	6	24	16	11	2	3	18
<b>DER (%)</b>	1.36	0.47	1.22	2.29	0.31	0.63	0.85	1.62	2.15	1.71	0.30	0.42	2.09
<b>MD</b>	117	127	141	139	154	218	256	488	297	445	577	606	724
<b>MWD</b>	4	4	2	4	3	7	10	31	7	9	15	29	26
<b>MER (%)</b>	3.42	3.15	1.42	2.88	1.95	3.21	3.91	6.35	2.36	2.02	2.60	4.79	3.59
<b>RNC</b>	87	86	87	97	97	103	123	192	168	174	191	225	241
<b>SNC</b>	91	92	96	101	105	112	128	196	175	184	204	239	250
<b>SER (%)</b>	4.60	6.98	10.34	4.12	8.25	8.74	4.07	2.08	4.17	5.75	6.81	6.22	3.73

**DER mean = 1.19% ± 0.74%**

**MER mean = 3.20% ± 1.31%**

**SER mean = 5.83% ± 2.34%**

**TER=22/91=24,18%**

with an error rate of  $1.81\% \pm 0.98\%$  and the merge classifier with  $3.25\% \pm 1.37\%$ . The proposed segmentation method detected  $93.71\% \pm 2.06\%$  of the cells. The tracking algorithm detected  $64.52\% \pm 16.02\%$  of cells in the tested images.

The presented alignment algorithm, in both semi-automatic and automatic methods, shown to align correctly all the images tested. However, the process is very slow, specially if the automatic mode is chosen..

## 5.2 Results Discussion

The results shown that the presented segmentation algorithm has a mean success rate of  $93.71\% \pm 2.06\%$  of cells correctly segmented in each image and that  $64.52\% \pm 16.02\%$  of the cells on first image are correctly followed. As it can be seen, the standard deviation of this last value is elevated which indicates that the performance significantly varies from time series to time series. In fact, the available images were very different from each other, being the time series 1 the one with a better resolution and less noise, allowing a tracking success rate of 79,58% cells correctly segmented and followed.

Despite the noise, other time series also had some inverted images which hindered the segmentation and the tracking process, resulting in a decrease of the success rate. Moreover, cells location and size also varied significantly between consecutive images, which detracted the tracking process. Besides, those 4 time series had also "dark bubbles" regions which sometimes were confused with cells. All these factors decreased the success rate of the presented tracking and segmentation algorithms.

The tracking algorithm followed  $64.52\% \pm 16.02\%$  of bacteria present in the first image. This value is lower than the desired and is mainly due to the fact that there were small misalignments between consecutive images.

The alignment method, as said before, shown good results although being computationally heavy and consecutively, very slow. This issue is lightly softened in the semi-automatic method. As it was mentioned before in Section 4.4, the alignment process has two phases: the search for an approximate position using larger steps (or given by the user, in the semi-automatic mode) and once found that position, search in its surroundings, the position that maximizes the fluorescence within cells contours.



## Conclusion

This work consisted on the development of an image processing tool to assist the study of *in vivo* proteins expression in E.coli cells.

A novel cell segmentation method for Differential Interference Contrast microscopy was developed. This method was based on a previous segmentation by the Gradient Path Labelling algorithm which produces an over-segmented image. That over segmentation was solved by machine learning techniques. In order to decide which of the over-segmented image regions were to discard and which were to merge, two classifiers were built, based on morphological and intensity attributes.

A method to align a segmented DIC image with a Confocal Fluorescence Microscopy image acquired simultaneously (in which the fluorescence emitted by cells was visible) was developed. An exhaustive-search of the relative position, scale and angle between both images in order to maximize the fluorescence within cells contours was undertaken.

The obtained results showed that the proposed algorithm had accomplished the goals that were previously set. It showed a good performance in the automatic detection of bacteria in DIC images, since  $93.71\% \pm 2.06\%$  of the cells were correctly detected. Moreover, the developed algorithm overcame the inverted images issue.

The tracking algorithm followed  $64.52\% \pm 16.02\%$  of bacteria. As it was previously mentioned, this result is lower than the desired and it is mainly due to the fact that there are small misalignments between consecutive images.

Finally, the alignment algorithm correctly aligned all the tested images. However, it is a very slow method.

In order to solve these last two issues, some additional work is needed.

## 6.1 Future Work

The low number of cells correctly followed from the first image until the last one of the time series, could be improved by, for example, correcting the small misalignments between consecutive images before performing the tracking algorithm through the employment of image registration techniques. Moreover, if the DIC images could be acquired with shorter intervals, the differences between consecutive images would be lower and consequently the tracking would have better results.

Although it returned good results, the alignment algorithm, is computationally heavy and slow. In order to overcome this problem, a new technique is being developed using the correlation of Fourier transforms. Because DIC and CFM images are not acquired exactly at the same time, cells locations change a little in that time fraction causing misalignments between both images. To overcome that situation, an algorithm is being developed which aligns both images by blocks, minimizing those misalignments.

The obtained fluorescence distributions over time can be used to calculate some bias in those distributions. Those bias can be, for example, between the old and the new pole of the cell or between two sister-cells in the moment of division. Those calculations could be, in a future work, included in the algorithm here described.

# Bibliography

- [1] X. Chen, X. Zhou, and S. Wong. Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy. *IEEE transactions on bio-medical engineering*, 53(4):762–6, April 2006.
- [2] A. Kuijper and B. Heise. An automatic cell segmentation method for differential interference contrast microscopy. *2008 19th International Conference on Pattern Recognition*, pages 1–4, December 2008.
- [3] J. Yu, J. Xiao, X. Ren, K. Lao, and X. Xie. Probing gene expression in live cells, one protein molecule at a time. *Science (New York, N.Y.)*, 311(5767):1600–1603, March 2006.
- [4] J. R. Albani. *Structure And Dynamics Of Macromolecules: Absorption And Fluorescence Studies*. Elsevier, 2004.
- [5] M. Müller. *Introduction to Confocal Fluorescence Microscopy*. Tutorial Texts in Optical Engineering. SPIE Press, 2006.
- [6] R. Ali, M. Gooding, M. Christlieb, and M. Brady. Advanced phase-based segmentation of multiple cells from brightfield microscopy images. In *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on*, pages 181–184, may 2008.
- [7] X. Cui, M. Lew, and C. Yang. Quantitative differential interference contrast microscopy based on structured-aperture interference. *Applied Physics Letters*, 93:3, 2008.
- [8] C. Preza, D. L. Snyder, and J. Conchello. Theoretical development and experimental evaluation of imaging models for differential-interference-contrast microscopy. *Journal of the Optical Society of America. A, Optics, image science, and vision*, 16:2185–99, September 1999.

- [9] M. Sumbali. *Principles Of Microbiology:M&S*. McGraw-Hill Education (India) Pvt Limited, 2009.
- [10] E. Bengtsson, C. Wählby, and J. Lindblad. Robust Cell Image Segmentation Methods 1. 14(2):157–167, 2004.
- [11] R. C. González, R. E. Woods, and S. L. Eddins. *Digital Image Processing Using Matlab*. Pearson Prentice Hall, 2004.
- [12] X.i Chen, X. Zhou, and S. Wong. Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy. *IEEE transactions on bio-medical engineering*, 53(4):762–6, April 2006.
- [13] A. Pinidiyaarachchi and C. Wählby. Seeded Watersheds for Combined Segmentation and Tracking of Cells. volume 3617 of *Lecture Notes in Computer Science*, pages 336–343. Springer Berlin / Heidelberg, 2005.
- [14] M. Wang, X. Zhou, F. Li, J. Huckins, R. King, and S. Wong. Novel cell segmentation and online SVM for cell cycle phase identification in automated microscopy. *Bioinformatics (Oxford, England)*, 24(1):94–101, January 2008.
- [15] U. Pal, K. Rodenacker, and B. Chaudhuri. Automatic cell segmentation in cyto- and histometry using dominant contour feature points. *Analytical cellular pathology : the journal of the European Society for Analytical Cellular Pathology*, 17(4):243–50, January 1998.
- [16] C. Solórzano, E. Rodriguez, A. Jones, D. Pinkel, W. Gray, D. Sudar, and S. Lockett. Segmentation of confocal microscope images of cell nuclei in thick tissue sections. *Journal of Microscopy*, 193(3):212–226, March 1999.
- [17] E. Battenberg and I. Bischofs-pfeifer. A System for Automatic Cell Segmentation of Bacterial Microscopy Images. *Cell*, pages 1–4, 2006.
- [18] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.
- [19] P. Bamford and B. Lovell. Unsupervised cell nucleus segmentation with active contours. *Signal Processing*, 71(2):203–213, December 1998.
- [20] C. Zimmer, E. Labruyere, V. Meas-Yedid, N. Guillen, and J.-C. Olivo-Marin. Segmentation and tracking of migrating cells in videomicroscopy with parametric active contours: a tool for cell-based drug testing. *Medical Imaging, IEEE Transactions on*, 21(10):1212–1221, oct. 2002.
- [21] A. Duarte, A. Sánchez, F. Fernández, and A. Montemayor. Improving image segmentation quality through effective region merging using a hierarchical social meta-heuristic. *Pattern Recognition Letters*, 27(11):1239–1251, August 2006.

- [22] J. Ning, L. Zhang, D. Zhang, and C. Wu. Interactive image segmentation by maximal similarity based region merging. *Pattern Recognition*, 43(2):445–456, February 2010.
- [23] N. Theera-Umpon and S. Dhompongsa. Morphological Granulometric Features of Nucleus in Automatic Bone Marrow White Blood Cell Classification. *Information Technology in Biomedicine, IEEE Transactions on*, 11(3):353–359, 2007.
- [24] P. S. Hiremath. Automatic Identification and Classification of Bacilli Bacterial Cell Growth Phases. *Image Processing*, pages 48–52, 2010.
- [25] M. Tscherepanow, F. Zollner, and F. Kummert. Classification of Segmented Regions in Brightfield Microscope Images. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 972–975, 2006.
- [26] T. Mitchell. *Machine Learning*, volume 4. McGraw-Hill Science/Engineering/Math, June 1990.
- [27] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann Publishers, San Francisco, 1988.
- [28] Y. Anzai. *Pattern Recognition and Machine Learning*. Academic Press, 1992.
- [29] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining - Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, third edition, 2011.
- [30] S. K. Pal and A. Pal. *Pattern Recognition: From Classical to Modern Approaches*. World Scientific, 2001.
- [31] S. Marsland. *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC machine learning & pattern recognition series. Taylor & Francis, 2009.
- [32] F. Camastra and A. Vinciarelli. *Machine Learning for Audio, Image and Video Analysis: Theory and Applications*. Advanced Information and Knowledge Processing. Springer, 2007.
- [33] L.M.M. Tijskens, L.A.T.M. Hertog, and B.M. Nicolaï. *Food Process Modelling*. Woodhead Publishing Series in Food Science and Technology. Woodhead, 2001.
- [34] M. Kantardzic. *Data Mining: Concepts, Models, Methods and Algorithms*. John Wiley & Sons, Inc., Hoboken, NJ, USA, July 2011.
- [35] C. Sammut and G. I. Webb. *Encyclopedia of Machine Learning*. Springer reference. Springer, 2011.
- [36] R. J. Lewis. An Introduction to Classification and Regression Tree ( CART ) Analysis. (310), 2000.
- [37] R. Kohavi and R. Quilan. Decision Tree Discovery. 3(Hunt 1962), 1999.

- [38] D. T. Larose. *Discovering Knowledge in Data: An Introduction to Data Mining*. John Wiley & Sons, 2004.
- [39] J. R. Quinlan. Induction of Decision Trees. pages 81–106, 2007.
- [40] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *CART: Classification and Regression Trees*. Wadsworth, Belmont, 1984.
- [41] R. Timofeev. Classification and Regression Trees ( CART ) Theory and Applications by. 2004.
- [42] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, December 2007.
- [43] S. Esakkirajan and T. Veerakumar. *Digital Image Processing*. Tata McGraw Hill Education, 2011.
- [44] W. Burger and M.J. Burge. *Principles of Digital Image Processing: Fundamental Techniques*. Undergraduate Topics in Computer Science. Springer, 2009.
- [45] W. Burger and M.J. Burge. *Digital Image Processing: An Algorithmic Introduction Using Java*. Texts in Computer Science. Springer, 2007.
- [46] Q. Wu, F. Merchant, and K. R. Castleman. *Microscope Image Processing*. Academic Press. Elsevier/Academic Press, 2008.
- [47] W. Burger and M.J. Burge. *Principles of Digital Image Processing: Core Algorithms*. Undergraduate Topics in Computer Science. Springer, 2009.
- [48] J.J. Goldberger and J. Ng. *Practical Signal and Image Processing in Clinical Cardiology*. Springer, 2010.
- [49] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.
- [50] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, January 1979.
- [51] S. Beucher and C. Lantuejoul. Use of Watersheds in Contour Detection. In *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, Rennes, France., 1979*.
- [52] I. N. Bankman. *Handbook of Medical Image Processing and Analysis*. Academic Press Series in Biomedical Engineering. Elsevier/Academic Press, 2008.



- [53] A. D. Mora, P. M. Vieira, A. Manivannan, and J. M. Fonseca. Automated drusen detection in retinal images using analytical modelling algorithms. *Biomedical engineering online*, 10(1):59, January 2011.





## **Appendix 1**

## Segmentation and tracking of *Escherichia coli* expressing *tsr-venus* proteins from combined DIC/Fluorescence images

C. Queimadelas<sup>1</sup>, J. Rodrigues<sup>1</sup>, A-B. Muthukrishnan<sup>2</sup>, A. Mora<sup>1</sup>, A.S. Ribeiro<sup>2</sup>, J.M. Fonseca<sup>1</sup>

<sup>1</sup>Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Lisboa, Portugal

<sup>2</sup>Laboratory of Biosystem Dynamics, Tampere University of Technology, Finland

### Introduction

Recent studies using microbes as model organisms rely on microscope imaging which needs to be complemented with reliable and fast methods of computer assisted image processing. These methods aim at facilitating the extraction of information from images of bacterial populations with single cell resolution, by avoiding manual analysis, which is fastidious, time consuming and subject to observer variances. To isolate single cells in microscopy images, image segmentation techniques are essential. However, segmentation of nontrivial images is one of the most difficult tasks in image processing. Here, we propose a method to assist the study of the *in vivo* kinetics of protein expression, one protein at a time, from confocal fluorescence microscopy images, complemented with differential interference contrast microscopy images (DIC), of *Escherichia coli* cells expressing *tsr-venus* proteins.

### Methodology

*E. coli* K-12 strain SX4 expressing *tsr-venus* was generously provided by S. Xie. For growth and induction conditions see [1]. Following induction, images were collected every 5 min for 1 h under the fluorescence confocal microscope and, in parallel, every 10 min by DIC.

In order to calculate the amount of fluorescence on each cell, we need to first segment the cells on the DIC image and then compute the level of fluorescence inside each cell on the fluorescence image.

When we apply any segmentation technique, the balance between under and over segmentation of cells is a challenge. On our work, we apply Gradient Path Labelling (GPL) [2] for cell segmentation (Figure 1.a). Before reducing over-segmentation by merging, it is essential to discard segments that do not correspond to cells, such as background and undesired objects (air bubbles and other artefacts). The decisions of what segments to discard or merge are based on an intelligent morphological algorithm that uses decision trees generated by the Classification and Regression Trees (CART) algorithm. To build the CART training set, images are manually analyzed by an expert that identifies whether a segment belongs or not to a cell and identifies which of them need to be merged with other cell segments. Based on the training set including images measurements such as area, perimeter and pixels intensity and the correspondent expert decision, CART executes a statistical analysis of the data and produces two automatic classifiers (for “cells to discard” and “cells to merge”) that are then used to produce the final result.

After the proper cell segmentation and merging, the resulting image (Figure 1.b) is projected onto the fluorescence image (Figure 1.c). Despite that the two types of images are taken almost simultaneously, they do not match in orientation and in area covered. The alignment and scaling of the DIC image with the fluorescence image is done by exhaustive searching over the first image of the time series.



Figure 1: Example of image processing steps: (a) Original DIC image (b) DIC image segmented by GPL showing moderate over-segmentation ; (c) Detail of the desired segmentation of DIC image after applied the intelligent cell segment selection and merge step; (d) Detail of a fluorescence image superimposed with the contours of the cells detected from the DIC.

After the segmentation of the DIC image the amount of fluorescence inside the cells is calculated for each possible position of the DIC over the fluorescence image. The position and scaling that maximizes the fluorescence inside the cells is chosen as the initial projection. For each of the subsequent images on the same time series a limited search around the last projection is done to maximize the fluorescence inside the cells and accommodate possible variations during the experiment. With the cell contours identified in fluorescence images, the area within each contour, the correspondent fluorescence area and its distribution over the major cell axis are calculated, allowing

the evaluation of the fluorescence emitted by each cell, from which one can extract protein numbers based on the known intensity from an individual *tsr-venus* protein.

## Results

The methodology proposed for cell segmentation and tracking was evaluated by quantifying the number of misclassifications of the classifiers and evaluation of the fraction of cells detected over time. Table 1 shows the results achieved by the selection classifier over the 13 images of the time series used for testing the algorithm. As it can be seen, the error on the selection decisions is always below 1,5% with an average error of  $0,78 \pm 0,26\%$ . Table 2 presents the error of the merge classifier over the same time series. The merge classifier achieves an error below 2% on all images with an average error of  $0,91 \pm 0,48\%$ . The resulting segmentation was evaluated by comparing the number of cells detected by the user with the number of cells detected by the proposed methodology (see Table 3). The error achieved by the cell segmentation algorithm is below 6% on all images with an average error of  $3,19 \pm 1,93\%$ . The error percentage achieved by both classifiers and the final error of the cells segmentation meets the requirements of the project. Nevertheless, comparison with other segmentation algorithms will be done in the near future.

	Image												
	1	2	3	4	5	6	7	8	9	10	11	12	13
<b>Total number of decisions</b>	453	426	444	452	463	500	521	539	581	634	583	637	696
<b>Positive decisions</b>	306	312	331	340	351	378	370	419	462	476	479	482	573
<b>Negative decisions</b>	147	114	113	112	112	122	151	120	119	158	104	155	123
<b>False positive decisions</b>	4	3	2	3	4	5	3	3	4	4	4	5	4
<b>False negative decisions</b>	0	3	0	0	0	0	0	0	2	0	0	0	0
<b>Misclassification rate</b>	0,88%	1,41%	0,45%	0,66%	0,86%	1,00%	0,58%	0,56%	1,03%	0,63%	0,69%	0,78%	0,57%

Table 1 – Selection classifier results

	Image												
	1	2	3	4	5	6	7	8	9	10	11	12	13
<b>Total number of decisions</b>	660	668	754	774	818	916	920	1078	1260	1314	1334	1416	1756
<b>Positive decisions</b>	342	326	376	380	386	410	378	458	492	502	490	448	560
<b>Negative decisions</b>	318	342	378	394	432	506	542	620	768	812	844	968	1196
<b>False positive decisions</b>	4	3	5	7	12	6	11	10	2	3	9	16	4
<b>False negative decisions</b>	0	1	1	2	2	4	3	1	0	1	2	5	5
<b>Misclassification rate</b>	0,61%	0,60%	0,80%	1,16%	1,71%	1,09%	1,52%	1,02%	0,16%	0,30%	0,82%	1,48%	0,51%

Table 2 – Merge classifier results

	Image												
	1	2	3	4	5	6	7	8	9	10	11	12	13
<b>Cells manually detected</b>	145	151	159	164	174	172	183	198	212	223	242	259	301
<b>Cells automatically detected</b>	143	154	155	161	177	183	186	205	224	233	241	270	318
<b>Misclassification rate</b>	1,38%	1,99%	2,52%	1,83%	1,72%	6,40%	1,64%	3,54%	5,66%	4,48%	0,41%	4,25%	5,65%

Table 3 – Final cell segmentation results

## Conclusions

The automatic analysis of sequences of images taken at regular intervals will allow characterizing the kinetics of protein production in live *E. coli* cells, one event at a time, as well as aid the inference with statistical methods of the duration of the underlying steps of transcription and translation, such as the open complex formation and translation elongation. For this, DIC images will be automatically segmented and aligned with fluorescence images allowing tracking and fluorescence evaluation on individual cells. The proposed method was shown to be able to extract the information necessary for this aim, in a timely fashion, and thus can be used to support studies of gene expression dynamics using *tsr-venus* proteins to assess the kinetics of expression of the gene of interest.

## References

- [1] J. Yu, J. Xiao, X. Ren, K. Lao, and X. S. Xie, "Probing gene expression in live cells, one protein molecule at a time.," *Science (New York, N.Y.)*, vol. 311, no. 5767, pp. 1600-3, Mar. 2006.
- [2] A. D. Mora, P. M. Vieira, A. Manivannan, and J. M. Fonseca, "Automated drusen detection in retinal images using analytical modelling algorithms.," *Biomedical engineering online*, vol. 10, p. 59, Jan. 2011.