Pedro Miguel Raminhos Ferreira

Licenciado em Engenharia Electrotécnica e de Computadores

Microscopy image segmentation by active contour models

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e Computadores

Orientador: Prof. Doutor José Manuel Matos Ribeiro da Fonseca, Prof. Auxiliar, FCT-UNL

Júri:
Presidente: Prof. Doutor André Teixeira Bento Damas Mora
Arguente: Prof. Doutor Yves Philippe Rybarczyk
Vogal: Prof. Doutor José Manuel Matos Ribeiro da Fonseca

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março de 2014

Microscopy image segmentation by active contour models

Acknowledgements

In first place, I would like to thank Professor José Manuel Fonseca for guiding and helping me pass all difficulties that emerged during this process.

To my parents and grandparents, to whom I owe everything and without them this work would mean nothing.

To all my friends, that accompanied me throughout this thesis, a big thank you.

Abstract

In this thesis a semi-automated cell analysis system is described through image processing. To achieve this, an image processing algorithm was studied in order to segment cells in a semi-automatic way.

The main goal of this analysis is to increase the performance of cell image segmentation process, without affecting the results in a significant way. Even though, a totally manual system has the ability of producing the best results, it has the disadvantage of taking too long and being repetitive, when a large number of images need to be processed.

An active contour algorithm was tested in a sequence of images taken by a microscope. This algorithm, more commonly known as snakes, allowed the user to define an initial region in which the cell was incorporated. Then, the algorithm would run several times, making the initial region contours to converge to the cell boundaries. With the final contour, it was possible to extract region properties and produce statistical data. This data allowed to say that this algorithm produces similar results to a purely manual system but at a faster rate.

On the other hand, it is slower than a purely automatic way but it allows the user to adjust the contour, making it more versatile and tolerant to image variations.

Keywords: segmentation, active contour, snakes

Resumo

Nesta dissertação, um sistema de análise de células semi-automático é descrito por processamento de imagem. Para desenvolver este sistema, um algoritmo de processamento de imagem foi estudado de modo a segmentar células de uma forma semi-automática.

O objectivo principal desta análise é aumentar a performance do processo de segmentação de imagens, sem que os resultados sejam demasiadamente afectados. Mesmo que um sistema completamente manual consiga produzir os melhores resultados, tem como desvantagem tornar-se repetitivo e longo quando o número de imagens a serem processadas seja muito grande.

Um algoritmo de contornos activos foi testado numa sequência de imagens tiradas por um microscópio. Este algoritmo, mais conhecido como snakes, permite ao utilizador definir uma região inicial que incorpora a célula. Depois, o algoritmo irá correr várias vezes fazendo com que o contorno da região inicial convergisse para os limites da célula. Com o resultado final, é possível extrair propriedades dessa região e produzir dados estatísticos. Estes dados permitiram dizer que este algoritmo apresenta resultados semelhantes a um sistema completamente manual mas a um ritmo mais elevado.

Por outro lado, é mais lento que um sistema completamente automático, mas permite ao utilizador fazer ajustes dos resultados, tornando este sistema mais versátil e tolerante a variações de imagens.

Palavras-Chave: segmentação, contornos activos, snakes

Table of Contents

List of Figures

List of Tables

Chapter 1

1. Introduction

1.1. Overview of the problem

The examination of cells by microscope has been the primary method for studying cellular function. Visual analysis can reveal biological mechanisms when cells are properly identified. In the meantime, advanced microscopes can now collect thousands of high resolution images of cells from time-lapse experiments. Even though several pioneering large screens have been scored through visual inspection by expert biologists, making the results to be very hard to replicate by a computer, for most applications, image cytometry (automated cell image analysis) is preferable to analysis by eye. The reasons will be discussed below.

Image cytometry is much less labor-intensive. With the appropriate software it is possible to obtain reliable results from a large-scale experiment in hours, versus the visual inspection that can take months. This is a critical advance since the presence of human error is reduced drastically, because there is no need to conduct routine and repetitive experiments.

Human-scored image analysis is qualitative. On the other hand, image cytometry produces consistent, quantitative measures for every single image. The automated analysis is also able to uncover samples of interest, hidden from the human eye, as well as draw some conclusions based on the quantitative measures of each image. This has proven to be a plus in cytometry profiling, since the identification of similar genes is easier to observe, thanks to the measuring of a large number of features.

The difference in feature scoring between human observers and image cytometry is huge. Human observers can typically score a few cellular features; image cytometry can obtain lots of informative measures of cells, such as intensity, localization, number, size, shape, etc. This is the reason why image-based analysis is high in information content, multiplexed and versatile. It is able to handle hundreds of thousands of distinct samples such as adherent cell types and time-lapse cell images.

Image cytometry is able to detect some features that a human observer cannot. Small but biologically significant differences, for example, a 10% increase in nucleus size, are not detected by the human eye. Pathologists have known for years that small changes in DNA or

protein texture can indicate profound and undetectable changes in cell physiology. This is a fact used to diagnose diseases as well as reveal the disease rate.

Quantitative image analysis measures each cell rather than the whole image. Since each cell's response is inhomogeneous, single cell data from different types of instruments is much more significant than population data for clustering genes, classifying protein localization and diagnosing disease. On the other hand, the single cell treatment reveals slight differences in measures that would have been undetected by the whole-population measures.

Many groups have benefited from automated cell image analysis, developing their own custom programs to attend their specific needs. Commercial software, such as Matlab, Java, etc. has been used to identify, measure, and track cells in images. However, most of the custom programs are not modular, making the routine of processing hundreds of thousands of images not practical, since the user has to interact directly with the code.

Commercial software for the pharmaceutical market has also been developed. The high cost of this software makes it impractical to test several programs for a new project. On the other hand, the proprietary nature of the code prevents researchers from knowing the program structure as well as not being able to modify it to attend specific needs.

High throughput analysis has been the main concern for the scientific community. It has been impractical unless someone develops a customized solution for a determined situation or the commercial packages are used for a limited set of cell types. The need for an open-source, flexible, powerful, high throughput cell image analysis platform has never been so high.

1.2. Overview of the solution

The mathematical description of object contours in an image (segmentation) has been target of investigation in the past few years. This kind of operation is significantly important to many image processing areas such as medical application, automatic surveillance and quality control. The mathematical description will allow the separation of the interest regions such as cells, objects or even people from the image and can be used to calculate shape factors of those regions such as area, perimeter and shape.

The test images used in this application were taken by a microscope and saved has a sequence in order for posterior analysis. The images contain several cells that can be together or not, and the goal is to separate each cell from the background of the image. This will be done in a semi-automatic way, where the user will give the application an initial contour (deformable

surface) for each cell and the application will try to adjust this deformable surface to the image boundaries, through a series of parameters and factors that can be changed accordingly, in order to obtain the desired outcome.

This technique has its disadvantages. The main one is that the initial contour is given by the user, subject to human error, making it difficult to reproduce and to compare results, as well as it can take a long time, depending on the number of images that have to be processed.

The main goal of this work is to use a contour mapping method, known as Snakes GVF[1]–[3], to identify semi-automatically cells in microscopic images. This algorithm was chosen because it is a simple and efficient solution to segment a region from an image, even though it depends from an initial contour. Other segmentation algorithms[4] are able to segment all image, however, it is very complicated to control the precision level, because of thresholdings. On the other hand this algorithm has its limitations, like the quality of the results, as well as when several regions are near each other, as you will be able to see further ahead.

These limitations were attenuated with a more active interaction with the user, making it easier for him to perfect the segmentation results in a fast and simple way. This way, we can obtain a solution that is quicker than manual segmentation as well as more perfect than automatic methods.

This thesis is organized in the following way; in chapter 2 the state of the art for automatic image contour segmentation is presented. In chapter 3, the methodologies used in the different stages of the system are described, giving a theoretical approach of the algorithms. In chapter 4, the system interface and functionalities are described. In chapter 5, the results obtained are shown and analyzed and in chapter 6 the conclusions and future work are presented.

Chapter 2

2. The State of the Art

The state of the art was studied considering the automatic detection of microscopic images contours. The automatic evaluation of micro-structures is a very specific area which makes the development of applications poorly documented and divulgated in the scientific world.

2.1. Automated differential blood count system

To evaluate the health of a patient, one of the most used medical exams is the Hemogram, which counts the blood particles (red blood cells, white blood cells and platelets).

The hematologist needs two types of counts in order to make his diagnostic. The first one is a CBC (Complete Blood Count) and the second one is a DBC (Differential Blood Count). The CBC is made by a flux cytrometry medical instrument which gives raw information about cell composition. The DBC is done through manual procedure, making it a longer process but a more precise and reliable one.

The automatic hemogram[5] is an application of automatic differential blood cell count that exists to ease the job of the hematologist.

The most relevant phase of this application is the segmentation. The goal is to find the cells through their contours. The contour detection is achieved by an active contour algorithm or snakes[1], which consists in a list of 2D points that evolve from initial positions to the desired contour, taking in consideration the internal and external forces of the image.

In figure 2.1 the segmentation steps of this model can be viewed.

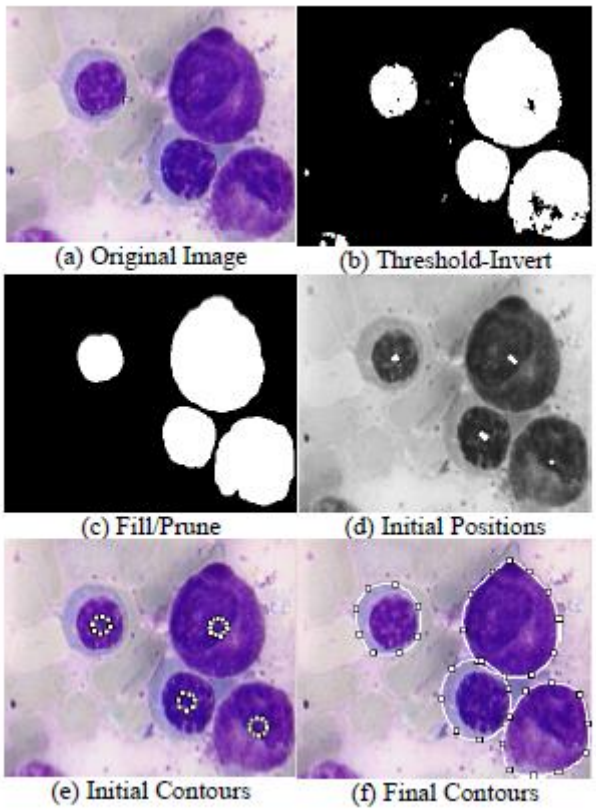

Figure 2.1 - Segmentation stages.

There are two major limitations to this algorithm. The first one is if the initial contour, made manually, is too far apart from the desired outcome. This will make the snake to not hold the desired contour. The second one is if the micro-structure has concavities. The snake does not adapt to this kind of regions. The latter does not influence this work since the cells are convex and elliptical.

2.2. Hybrid System for Cell Detection in Digital Micrographs

This system[6] emerged because of the need to surpass performance issues limited by singular approaches to the micro-structures contour detection problem, such as Artificial Neural Networks (ANN), morphologic operators and adjust algorithms based on models. As a result, a hybrid system was designed, that combines the Hough transform with the multi-layer perceptron neural networks (MLP) obtained through the back propagation algorithm.

The Hough transform calculates the candidate cell positions, making a map of all points that belong to the contour of a single object. The more regular the object contour is, the bigger

the number of points mapped in the same position of the array is. The candidate positions are then inserted in the neural network trained by the MLP.

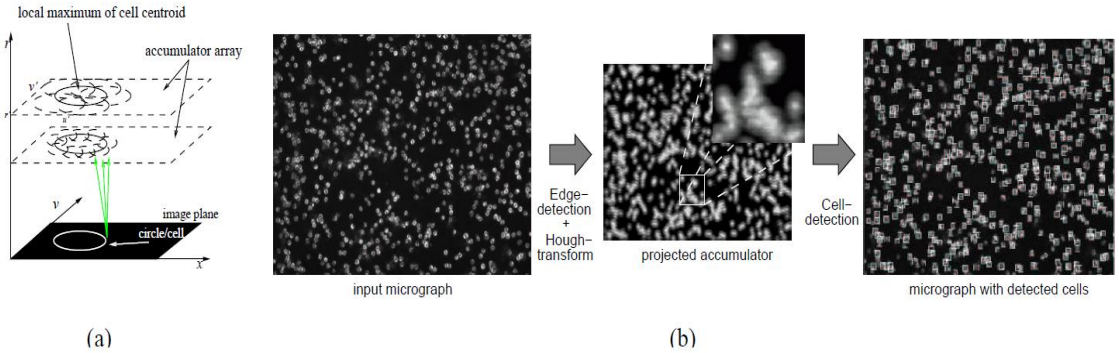

Figure 2.2 - Hybrid system for cell detection. (a) Mapping of the contour pixels to cells. (b) Cell detection through the use of MLP.

This approach made a significant improvement to the performance of micro-structure contour detection. However, this system still depends on a set of standard images for the neural network to learn. The other drawback of this system is that in order to get good results, the contours have to be somewhat regular, because of the Hough transform.

2.3. Neural network architecture for automatic segmentation

This architecture[7] was created for an automatic segmentation of fluorescence micro-structures. In the first stage, the position of the fluorescence cells is detected by a neural network. Its visual knowledge is acquired through a set of images given by the user. In a later stage, the system detects the cells contours using a neural network model.

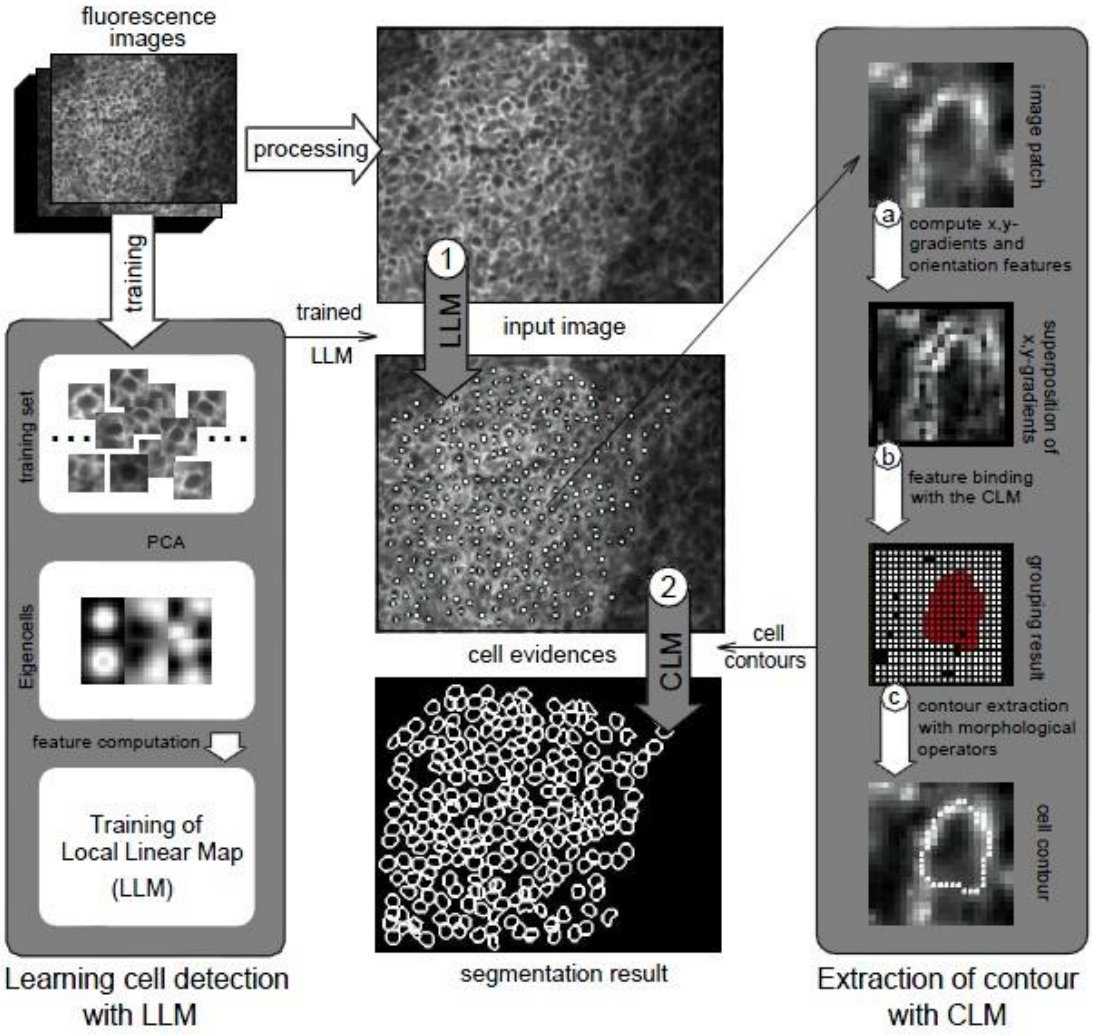

Figure 2.3 - Neural network architecture for automatic segmentation.

The detection of the cells position is made through a neural classifier which is trained to calculate the evidence values for each point of the image. These values represent the probability of a point being occupied by a fluorescence cell. The contour of each cell is then calculated through the separation of the cell from its environment. This is possible using a figure-ground segmentation with a Competitive Layer Model (CLM)[8]. An example of this application in fluorescence micro-structures can be seen in figure 2.4.

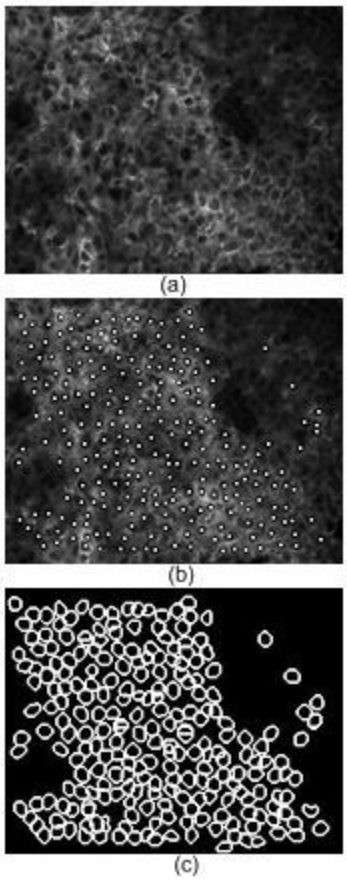

Figure 2.4 - Example of segmentation. (a) Image sample. (b) Computation of the position of the cells. (c) Contour extraction from the focus points.

According to this study, even though the micro-structures has a lot of noise and the fluorescence cells vary in shape and size, the system is able to detect a minimum of 95% of cells. However, this system depends on a set of examples to achieve a decent success rate.

2.4. Contour detection of human kidney by Markov random fields

This method[9] uses an automatic contour detection of the kidney through a probability model of Markov field deformation in relation to a standard model of the kidney contour. This model is adjusted, manually or automatically, to the image and is smoothly deformed following the borders of the ultra-sound image and guided by an empirical model of distribution of the echographical data. The level of smoothness is imposed by the spectrum of allowed deforming values and a standard distribution.

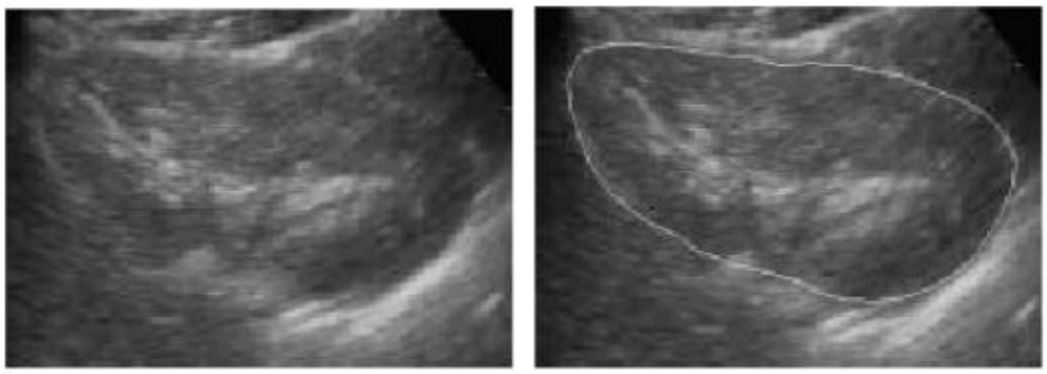

Figure 2.5 - Automatic contour detection through a probability model of Markov field deformation.

The disadvantages of this application are the dependence of a standard kidney model design and the results only affecting the external structure of the kidney. It is not possible to analyze internal structures of the kidney, even though the external contour restricts the analysis to a smaller area.

2.5. Tumor cell identification using feature rules

The identification of tumor cells has been object of study for several years. One of the fundamental elements of this identification is the quantitative analysis of the tumor cells to evaluate the disease's dissemination degree. In traditional pathology, carcinogenic cells are introduced in non-human test subjects in order to analyze tissues, counting the number of affected cells manually. This process turns out to be rather slow and counter-productive.

There have been several tries to automate this process of counting carcinogenic cells, but all of them had limited success because of the rather complex nature of the images. This work[10] proposes a robust local adaptive thresholding and dynamic water immersion algorithms to segment regions of interest from background.

However, because of the histological noise in the images, a large number of false positives is obtained. To enhance this identification of carcinogenic cells, meaningful features are extracted from the segmented regions. Three classifiers are used to generate a set of rules that allows differentiating carcinogenic cells from the histological noise.

2.6. High throughput, subpixel precision analysis of bacteria

"*Bacteria display various shapes and rely on complex spatial organization of their intracellular components for many cellular processes*"[11]. It was with this in mind that MicrobeTracker was developed. This software is able to segment cells with subpixel precision and does cell lineage tracking.

The software is divided in two phases. The first one consists in separating the cells or cell clusters from the background, applying edge-detection algorithms and watershed transforms (fig. 2.6). In the second stage, a variant of active contour models is used in order to refine the cell outlines obtained in the first stage. The cell contour is adjusted smoothly through the action of image forces until it converges to the cell boundaries.

On the other hand, when used to analyze time-lapse sequences of growing cells, it is even more powerful. This happens because it takes advantage of time-dependent information to resolve complicated images. Basically, for each time frame, the original shape for the next frame is taken from the previous one. This process ensures the identity of each cell over time, allowing the program to keep track of the history and genealogy of the cells.

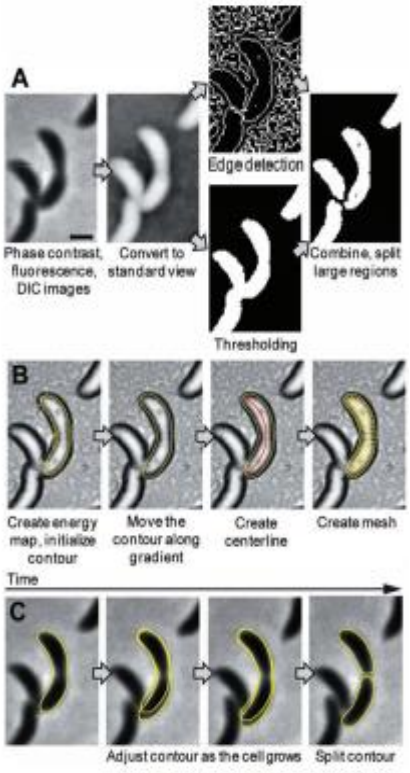

Figure 2.6 - MicrobeTracker operations. (A) Image preparation using inversion, thresholding and edge detection algorithms. (B) Active contour model. (C) Cell contour in image sequences.

2.7. Measuring single-cell gene expression dynamics in bacteria

"*Quantitative single-cell time-lapse microscopy is a powerful method for analyzing gene circuit dynamics and heterogeneous cell behavior*"[12]. The implemented protocol involves seeding and growing bacteria and imaging the results, through an automated microscopy system. The images are then reviewed and analyzed using a custom Matlab analysis system named Schnitzcells.

The Schnitzcells software allows users to perform analysis of time-lapse images of fluorescent proteins in living cells. Image analysis is done in four stages (fig. 2.7):

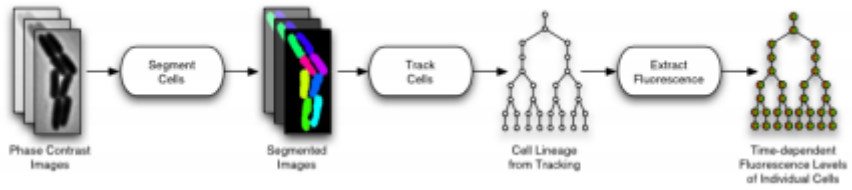

Figure 2.7 - Schnitzcells data flow overview.

- Preparing for analysis, specifying directories and parameters;
- Segmenting each image to define cell boundaries, manually correcting the segmentation;
- Tracking cells to obtain various cell data and to create a cell lineage tree, making cell division events possible to recognize;
- Extracting fluorescence intensity data from cells.

The segmentation of cells is a multi-stepped process. First, it applies edge detection to generate an initial segmentation, then splits long or clumped cells and finally identifies too small objects as false positives and discards them.

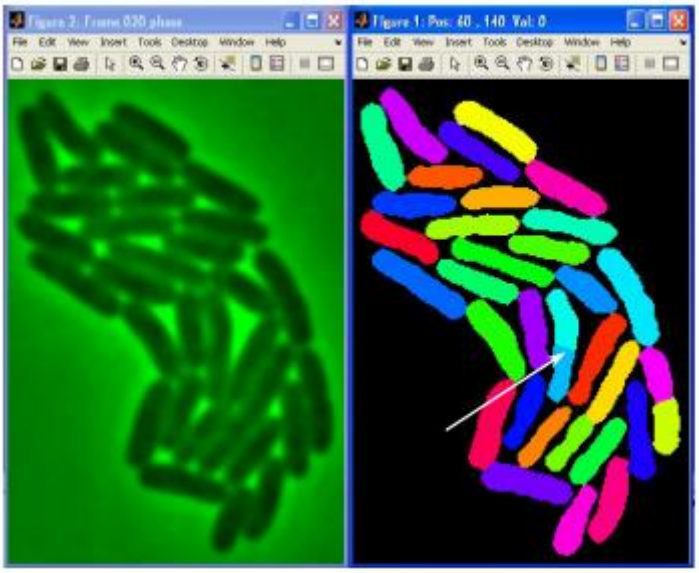

Figure 2.8 - Schnitzcells segmentation result. On the left, the phase image and on the right is the mask output of the automated cell-segmentation.

2.8. CellProfiler – software for biological image analysis

CellProfiler is a modular image analysis software that is capable of handling hundreds of thousands of images[13]. The software contains methods for many cell types and is an open-source program. This allows sharing, testing and development of new methods by image analysis experts. The platform has advanced algorithms for image analysis, flexible design in order to adapt to new phenotypes, and is open-source, making it easy to modify or improve.

The image analysis process is accomplished just by pointing and clicking using CellProfiler's graphical user interface (fig. 2.9 (a)). The software uses the concept of a sequence of modules, where each module processes images in its own way, for example, image preparation, object segmentation and measurement.

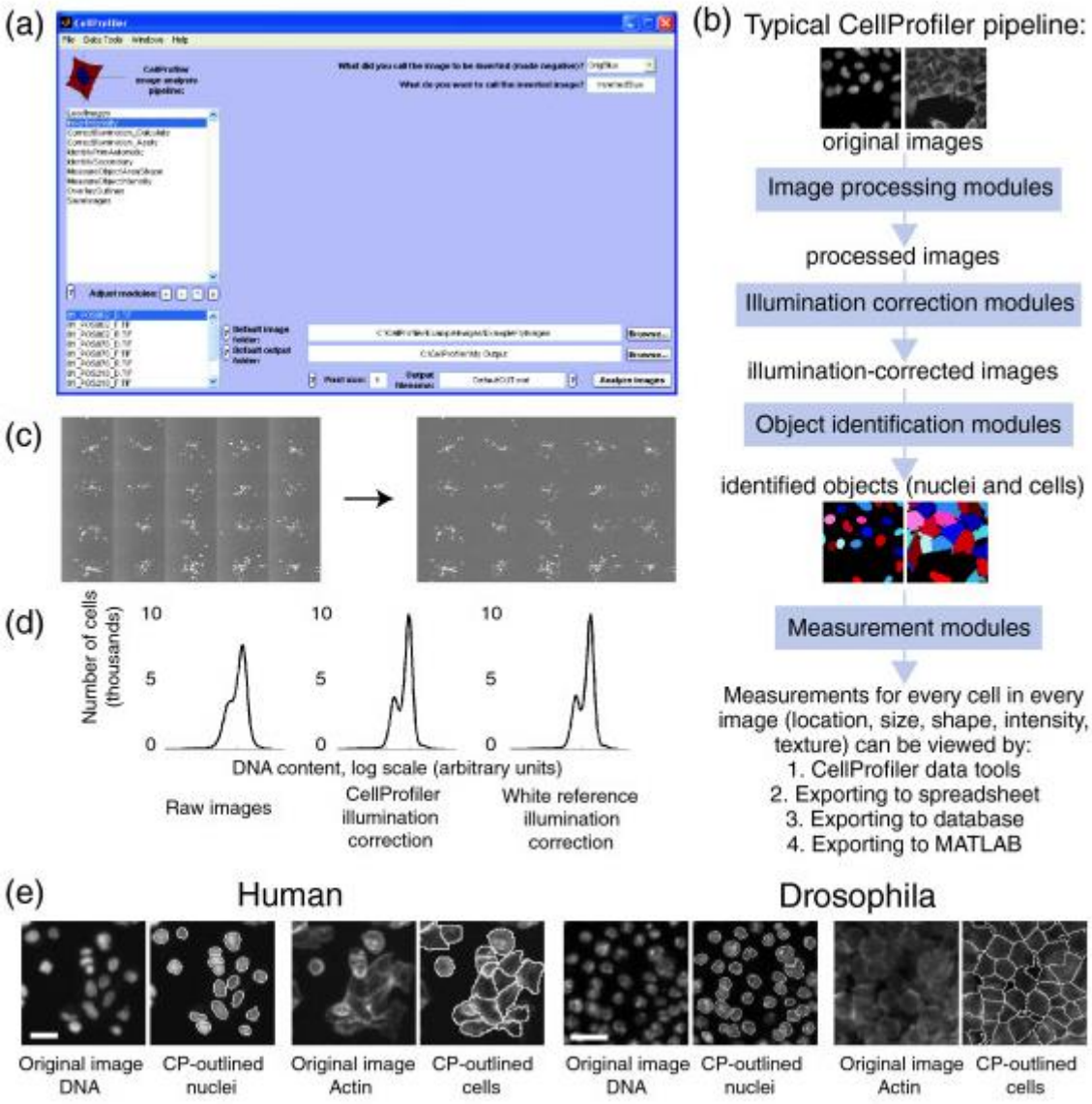

Figure 2.9 - CellProfiler overview and features.

In figure 2.9 it is possible to see the CellProfiler graphical user interface (a), in (b) a typical CellProfiler module sequence is displayed, in (c) it is possible to see the result of the illumination correction module, in (d) the impact of the previous correction is shown, because it reduces noise in quantitative measurements, and finally in (e) shows the identification of human HT29 (left) and Drosophila (right) cells. The cells in the image borders are intentionally excluded from the process.

Chapter 3

3. Methodology

To face the problems above, an automatic system was proposed to perform the segmentation of cell images. The system uses image processing algorithms to analyze a series of images and calculates the region properties of the cell, giving relevant information to the user. The main objective is to create an interface that makes the analysis of these images easier for the user.

The segmentation process is achieved by using active contours[2], [3]. The user will be able to execute several iterations of the algorithm in order to obtain the most accurate contour.

3.1. System Architecture

The several stages of the system are represented in figure 3.1.

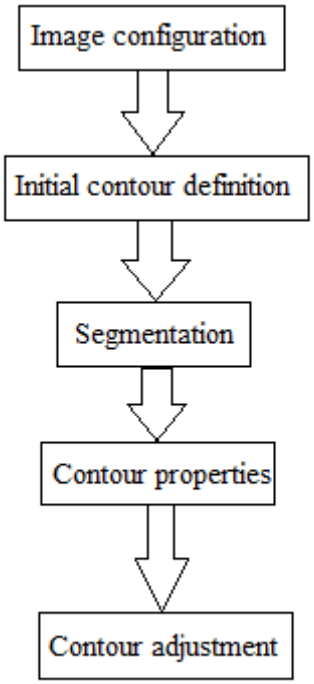

Figure 3.1 - System architecture.

Image configuration

This is the starting stage of the system. It is responsible for the noise reduction and preparation of the images for the segmentation phase. The pre-processing includes a conversion of the image to gray-scale, the use of a Gaussian Blur filter and calculation of the image gradient.

Initial contour definition

As the name indicates, in this stage a set of initial points will be established so that the algorithm can start operating.

Segmentation

In this stage, the contours of the cells are obtained using the Gradient Vector Flow[2] algorithm, with the help of the user.

Contour properties

After the segmentation has occurred, the user will be able to obtain some information, such as the area, perimeter, centroid, radius, from the region that was obtained.

Contour adjustment

In this final stage, the user will be able to make some adjustments to the contour in order to obtain a more accurate and precise result.

All of these stages will be described in detail in the next sections.

3.2. Image configuration

In the image configuration stage, three operations are executed, gray-scale conversion of the image, contour calculations, needed for the algorithms used in this work and GVF field calculation, described in 3.5.2. These operations are made so that the algorithms can run more rapidly.

To calculate the contours, a Gaussian Blur[14] filter was applied to reduce the noise of the images. The cell images are obtained through a microscope and are relatively low quality (low signal noise relation). This filter is used in programs such as Adobe Photoshop, Paint.NET

and Inkscape to clear the image noise and reduce the detail level. Applying a Gaussian blur filter is the same as doing a convolution of the image with a normal distribution. Since the Fourier transform of a Gaussian is another Gaussian, applying a Gaussian Blur filter as the same effect as applying a low-pass filter to the image.

The next step is to obtain the gradient of the image, using a Sobel mask. This gives the intensity of the contour for the various pixels of the image. In vector calculus, the gradient of a scalar field is no more than a vector field whose direction is determined by the maximum grow rate of the scalar field and the amplitude is equal to the absolute value of the variation rate.

3.3. Initial contour definition

The initial contour definition can be done in two ways. The first one is manually, through a set of points that create a polygon that is similar to the boundary of the cell (figure 3.2). The other way is to provide the set of points, previously calculated, loading them into the system, through the interface.

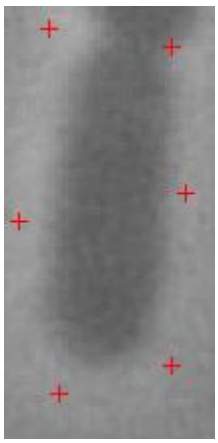

Figure 3.2 - Initial contour definition.

3.4. Segmentation

The segmentation is a rather complex process that depends of the content and noise of the images. The classic methods of edge detection[15] or thresholding[16], based on local operations and image intensity, are unreliable when obtaining results in the majority of

situations. More complex algorithms are needed; ones that analyze the content of the image in a global manner. In this group, the active contour algorithms stand out and are used in several applications such as deformable models[17], segmentation[18] and tracking[19].

In this work, the segmentation process is done using active contours, applied to the filtered images. This process will also try to keep track of the cells since they are not stationary.

3.4.1. Active Contour

An active contour is defined by a list of 2D points, given manually or automatically, designated control points. The control points evolve through a set of iterations over the image by the application of external forces (data dependent) and internal forces (contour dependent). The internal forces make the contour deformation level vary through a set of parameters of elasticity and flexion, and the external forces (calculated based on the image gradient) force the contour to the borders of the object.

The traditional snake is a curve $x(s) = [x(s), y(s)]$ defined to minimize the energy given by the equation

$$E = \int_0^1 [\frac{1}{2}(\alpha|x'(s)| + \beta|x''(s)| + E_{ext}\big(x(s)\big)]ds \qquad (1)$$

where α and β are weights that control the tension and the rigidity of the snake, respectively and are applied to the first and second derivative of x(s).

The external energy is obtained from the image $I(x, y)$ and should have lower values in the points of interest. Our goal is to make the contours closer to each other. A good choice is$E_{ext}(x, y) = -|\nabla(G_\sigma(x, y) * I(x, y))|^2$, where $G_\sigma(x, y)$ is the Gauss function with standard deviation σ and ∇ is the gradient operator, because the contour points are the ones who present greater gradient value and less external energy value.

A snake that minimizes (1) should satisfy the following Euler equation

$$\alpha \cdot x''(s) - \beta \cdot x'''(s) - \nabla E_{ext} = 0 \qquad (2)$$

which can be view as a force equilibrium, where the internal force is responsible for shrinking and smoothing the snake and the external for pulling the snake to the contour points.

To solve (2), the snake has to become dynamic, treating x like a t function.

$$x_t(s, t) = \alpha \cdot x''(s, t) - \beta \cdot x'''(s, t) - \nabla E_{ext} \quad (3)$$

A numeric solution can be found making it a discrete equation and solving it until $x(s,t)$ stabilizes. For more information, see[1].

However, the use of active contours poses two main difficulties[3]. The first one is that the initial contour as to be in the proximity of the object boundary, so that the snake can converge to the desired object. The other problem is the fact that active contours have problems when treating objects with concavities.

In figure 3.3 there is an example of the traditional snake algorithm being applied onto concave objects.

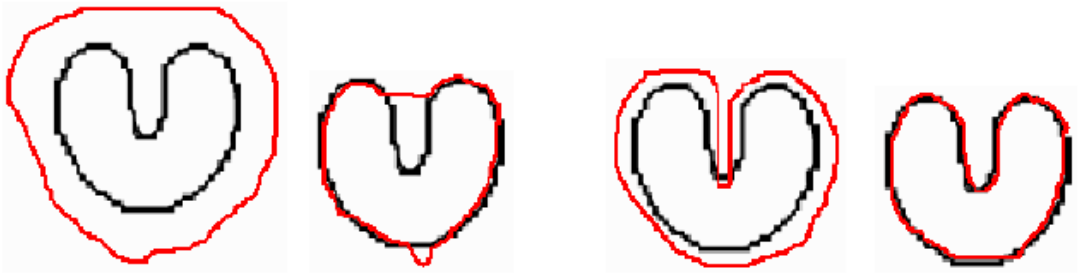

Figure 3.3 - Traditional snake algorithm applied to concave objects.

The traditional snake needs the initial contour to be designed near the desired contour, because of the weak external forces intensity (see figure 3.4). On the other hand, the difficulty of snakes getting near concave contours is verified, because in these areas the external forces are pointed horizontally, making the contour near to the vertical contours and not progressing to the concavity.

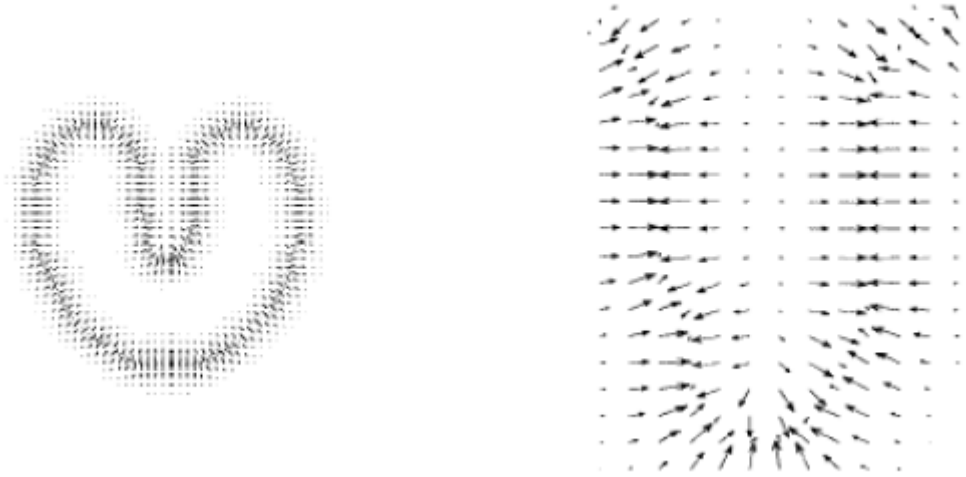

Figure 3.4 - External forces in traditional snake algorithms.

3.4.2. Gradient Vector Flow (GVF)

The active contour algorithm used in this work, pretends to overcome, in a more satisfying form, these difficulties through the definition of a new force field. This algorithm makes the contour evolve over a non-irrotational field of external forces, calculated from the cell images, named Gradient Vector Flow(GVF)[2].

The GVF is a vector field whose elements (2D vectors) point towards the borders and its module is minimal in the homogeneous regions. In the neighbor points, the vectors are perpendicular to their boundary and its module is going to grow until it is near it. Another feature of the field is the large range spectrum.

In homogeneous regions the vector module is reduced; however even in the farthest areas of the boundary, it continues to point towards it.

Figure 3.5 - GVF algorithm application.

As it can be seen in figure 3.5, the results are much better than the traditional snakes. The external forces have their intensity raised and point towards the concavity interior (figure 3.6).

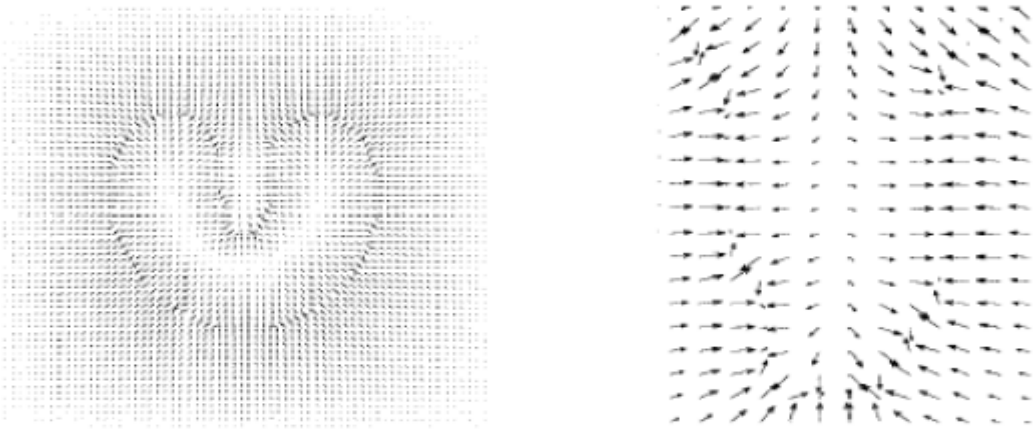

Figure 3.6 - GVF algorithm external forces.

To obtain the new external field, an edge map is defined for each image of the sequence. The edge map is calculated through the convolution of the image with a Gaussian filter, $f(x,y) = |\nabla\sigma(x,y) * I(x,y)|$, with $\sigma = 1$, and is greater near the interest borders. This result is used to calculate the GVF field through the evolution of the active contours. It is important to notice that greater σ values, makes the range of the active contours higher, however the borders of the image are less clear.

The new vector field $v(x,y) = [u(x,y), v(x,y)]$ described looks to minimize the energy function

$$\varepsilon = \iint \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2 |v - \nabla f|^2 dxdy \quad (4)$$

where ∇f is the gradient of the edge map and is looking to smooth the outcome where there is no relevant data. This way, when ∇f is lower, the energy is dominated by the partial derivatives of the field, resulting in a smoothing operation. On the other hand, when ∇f is higher, the second part of the integration dominates the function and is minimized by $v = \nabla f$. The parameter μ is a regularization factor of the GVF field and has to be adjusted accordingly to the present noise in the image. The GVF field can be calculated using Euler equations[20],

$$\mu \nabla^2 u - (u - f_x)\left(f_x^2 + f_y^2\right) = 0 \qquad (5)$$

$$\mu \nabla^2 v - \left(v - f_y\right)\left(f_x^2 + f_y^2\right) = 0 \qquad (6)$$

where ∇^2 is the Laplace operator. It is possible to notice that in the homogeneous region, where the edge map has fixed value, the second term of both equations disappears because the gradient of $f(x,y)$ is zero. This way, the regions u and v are determined by the Laplace equation. This produces the desired effect of "filling-in" the field with information given by the boundaries of the object. This fact explains the reason why vectors of the GVF field point towards the concavities of the object.

The equations 5 and 6 can be solved treating u and v as time functions and solving the generalized diffusion equations[2].

Now let's have a look to the mathematical description of the active contour. Being this described by the following curve $c(s) = [x(s), y(s)]$, where $s \in [0,1]$, the optimal curve has to minimize the following energy function,

$$E = E_{ext} + E_{int} \qquad (7)$$

where

$$E_{int} = \frac{1}{2}\int_0^1 [\alpha |c'(s)|^2 + \beta |c''(s)|^2] ds \quad (8)$$

and α and β determine the tension and rigidity level of contour, respectively. The external energy E_{ext} depends on the filtered image and is minimal near the boundaries. The stationary condition is obtained through the Euler equation,

$$F_{ext} + F_{int} = 0 \qquad (9)$$

where

$$F_{ext} = -\nabla E_{ext} \qquad (10)$$

$$F_{int} = \alpha c''(s) - \beta c''''(s) \qquad (11)$$

The internal force F_{int} prevents the snake from stretching and bending while the external force F_{ext} pulls the snake to the image contour.

The Euler equation is solved iteratively, starting from an initial contour $c(s,0)$ and evolves through time to $c(s,t)$, where t is a discrete time variable and gives the number of iterations. The final contour $c(s)$ depends from the initial contour $c(s,0)$. Mathematically, the snake is made dynamic, treating x as a time function.

$$x(s,t) = \alpha x''(s,t) - \beta x''''(s,t) - \nabla E_{ext} \qquad (12)$$

where the external force term is substituted by the new calculated GVF field,

$$x(s,t) = \alpha x''(s,t) - \beta x''''(s,t) + v \quad (13)$$

The equation is solved using discretization and the solution is found iteratively.

3.4.3. Implementation

The algorithm used in this work was created by Chenyang Xu e Jerry L. Prince [2], [3]. The main function of the algorithm was replaced by a C version, which allowed an increased speed in terms of processing.

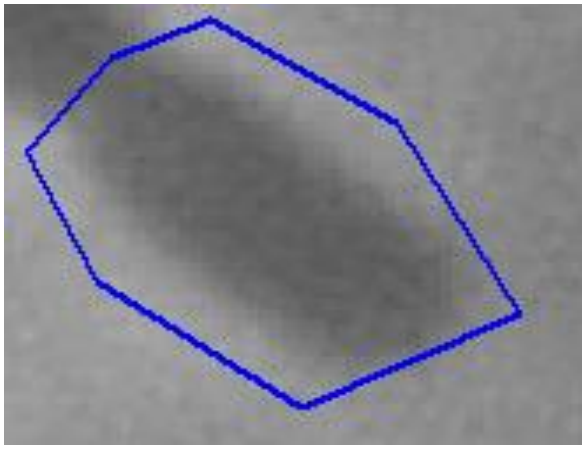

Figure 3.7 - Example of region obtained before applying the snake GVF.

In this work the initial contour is defined manually by the user, through a set of points that form a polygon and is near the cell boundaries (figure 3.7). From the first image, it is possible to obtain the initial contour for the next image, making the process much quicker. However, the images cannot vary too much because the snake will not converge to the right boundary.

From the polygon, the snake GVF algorithm will make the contour evolve iteratively until the user is comfortable with its result (figure 3.8).

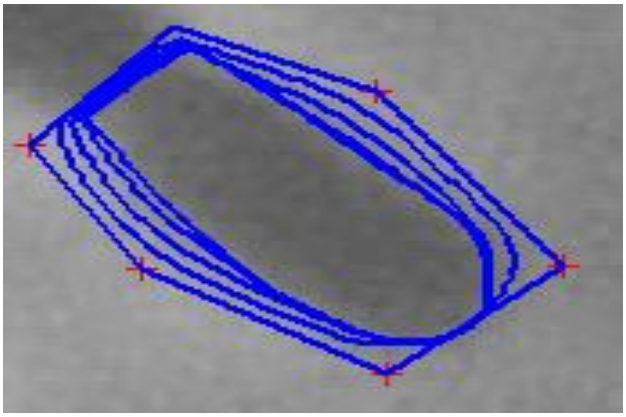

Figure 3.8 - Snake GVF behavior.

The values of attraction and elasticity are critical for this process. The default settings were obtained experimentally and are adjusted to this kind of images. However, it is possible to adapt them to other situations.

In figure 3.9 it is possible to see examples of choosing incorrect settings for the algorithm. In this case, the external force weight was very low, on the left image, making the contour not to hold to the boundaries of the cell. On the right image, it is possible to see the opposite; the external force weight is too high, making the contour unstable and not converging to the cell boundary.

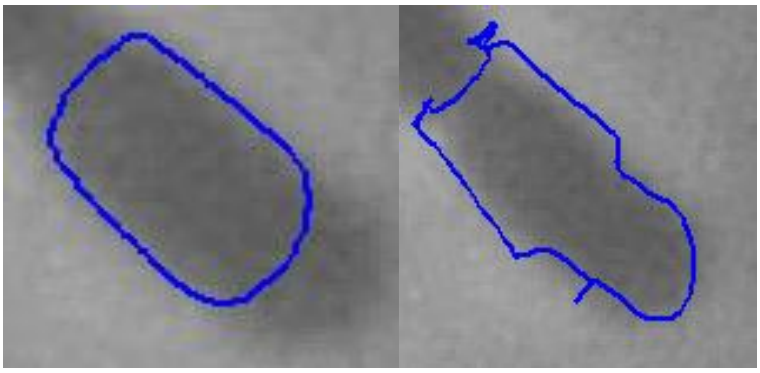

Figure 3.9 - Contours with wrong settings.

3.5. Contour properties

After the detection has occurred, a set of parameters is calculated to give some information about the obtained region.

The parameters are:

- Area (A);
- Perimeter (P);
- Centroid (Cx, Cy);
- Horizontal and vertical radius (r1, r2);
- Angle between the horizontal radius and vertical axis (θ).

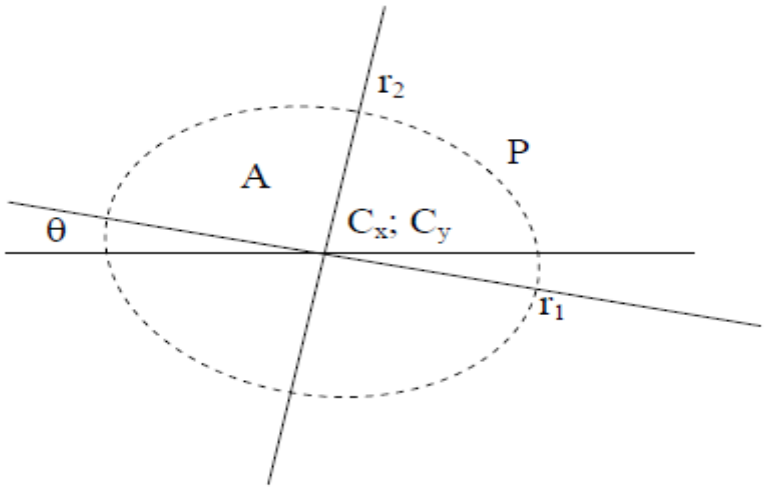

Figure 3.10 - Contour properties schematic.

A Matlab function from H. J. Sommer[21] was used to calculate geometric parameters of polygons through its moment of inertia.

Being the polygon contour of n vertices with $x_i\, y_i$ coordinates,

$$\Delta x_i = x_{i+1} - x_i$$

$$\Delta y_i = y_{i+1} - y_i$$

the area and perimeter can be calculated by these equations,

$$A = \sum_{i=1}^{n} (\frac{(y_i \Delta x_i - x_i \Delta y_i)}{2}$$

$$P = \sum_{i=1}^{n} \sqrt{{\Delta x_i}^2 + {\Delta y_i}^2}$$

To obtain the centroid, it is necessary to calculate the first moment of area in both axis,

$$A_{xc} = \sum_{i=1}^{n} \frac{(6x_i y_i \Delta x_i - 3{x_i}^2 \Delta y_i + 3{y_i}^2 \Delta x_i + \Delta {x_i}^2 \Delta y_i)}{12}$$

$$A_{yc} = \sum_{i=1}^{n} \frac{(3{y_i}^2 \Delta x_i - 6x_i y_i \Delta y_i + 3x_i \Delta {y_i}^2 + \Delta x_i \Delta {y_i}^2)}{12}$$

The moments of centroid are

$$x_c = \frac{A_{xc}}{A}$$

$$y_c = \frac{A_{yc}}{A}$$

If x_m and y_m are the contour vertices media in each axis, it is possible to calculate the centroid for each axis with these equations,

$$C_x = x_c + x_m$$

$$C_y = y_c + y_m$$

Calculating r_1 and r_2

Considering r_{MX} and r_{MY} the maximum set of points of the contour for the X and Y axis, and r_{mX} and r_{mY} the minimum set of points of the contour for the X and Y axis, the radius of each contour axis is given by these equations

$$r_1 = \sqrt{(C_y - r_{MY})^2 + (C_x + r_{MX})^2}$$

$$r_2 = \sqrt{(C_y - r_{mY})^2 + (C_x + r_{mX})^2}$$

Calculating θ

Considering P_{rX} andP_{rY}, as the radius projections obtained above, the angle is obtained through this equation

$$\theta = \tan^{-1}(\frac{P_{rY}}{P_{rX}})$$

3.6. Contour adjustment

This part of the system is available to the user after the automatic detection has occurred. This will allow a more accurate drawing between the cell boundary and the contour obtained by the snake.

This adjustment is done in two stages. The first one is defining the segment where the adjustment has to occur and the second one is the manipulation of that segment until the user is satisfied with the outcome.

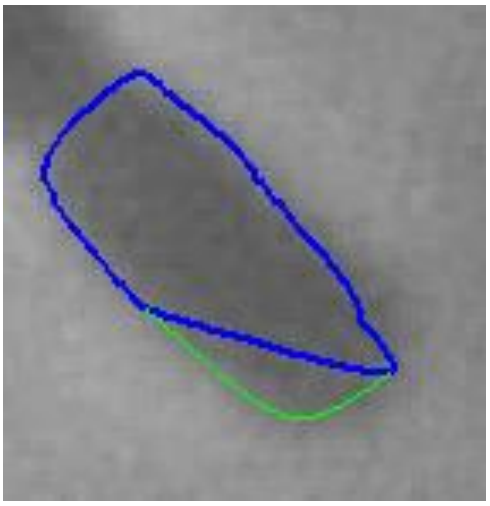

Figure 3.11 - Example of contour adjustment.

The definition of the segment is obtained by calculating the nearest contour point from the mouse cursor. This allows obtaining the initial and final contour point that define the segment that needs adjustment. The nearest contour point from the mouse cursor is determined by comparing the Euclidian distance[22] between the several points of the contour with the location of the mouse cursor.

Considering $P = (p_1, p_2)$ and $Q = (q_1, q_2)$, the distance between these two points, in the Euclidian plane, is

$$D(P, Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$

If this equation is applied to all contour points given by the mouse cursor location, the nearest point will be determined.

After defining the segment, all points within the segment are erased and a new set of points is created using a cubic Bézier curve[23].

The cubic Bézier curve is a parametric curve defined by four points (see figure 3.12). The first (P_0) and last (P_3) control points are always the end points of the curve. However, the intermediate control points(P_1, P_2) , the ones that define the shape of the curve, do not lie on the curve.

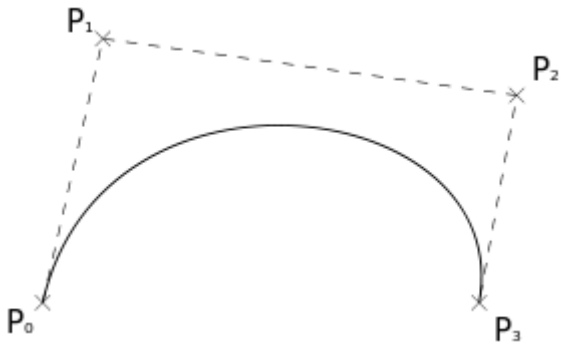

Figure 3.12 - Cubic Bézier curve.

The curve starts at P_0 going towards $P_1$and arrives at P_3 coming from the direction of P_2. Usually, the curve will not pass by P_1 and P_2, since these points are just there to provide directional information for the curve. The distance between P_0 and P_1, determines how long the curve takes to move in direction of P_2 before going to P_3.

The parametric form of the curve is

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t)t^2 P_2 + t^3 P_3 \, , t \in [0,1]$$

To make the adjustment easier, the intermediate points P_1 and P_2 are defined by the mouse cursor position as one.

Chapter 4

4. System interface and functionalities

The application was developed in Matlab 8.0.0. The visual aspect of the interface can be seen in figure 4.1.

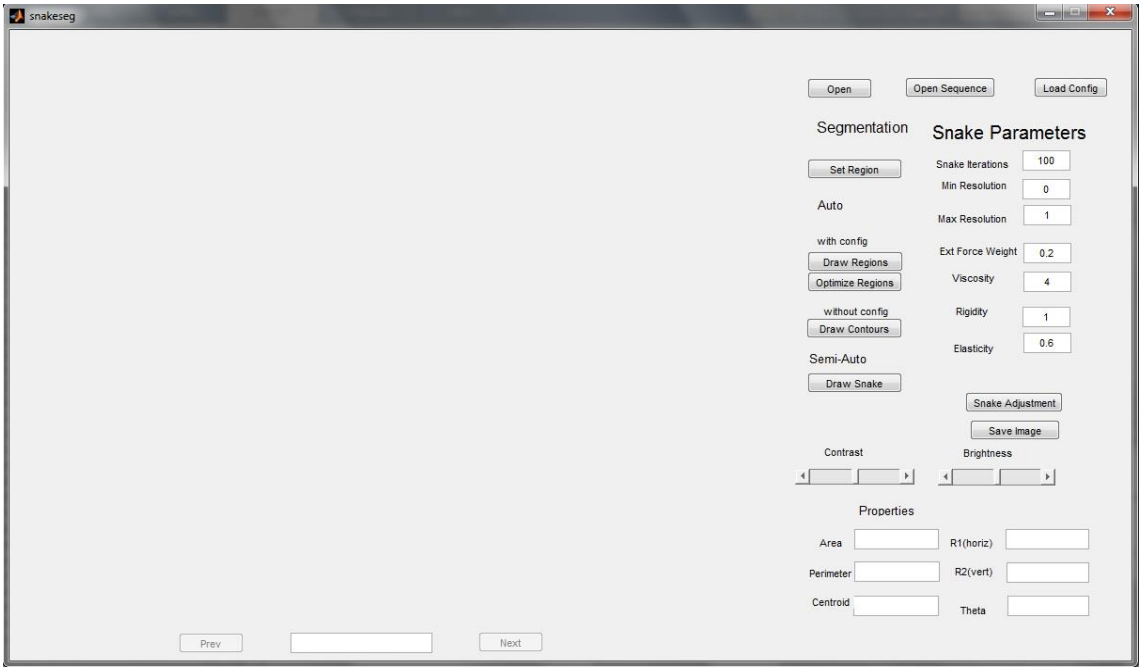

Figure 4.1 - Graphic interface.

This application is divided in 6 sections:

- Image opening;
- Snake parameters;
- Segmentation;
- Region properties;
- Contrast and brightness adjustment;
- Contour adjustment and image saving.

4.1. Image opening

In this section, the user has two options. The first one is to open a single image and the other is to open a sequence of images (see figure 4.2).

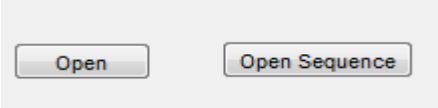

Figure 4.2 - Image opening options.

The option Open Sequence is used when the user wants to analyze a sequence of images that represent the evolution of cells. This mode allows the user to make the initial contour on the first image and in the next images the contour is obtained automatically, through the result of the previous one.

Below the image, there are two buttons that allow the user to change between every image within the sequence, showing the contour and the shape factors.

4.2. Snake parameters

To make the contour, a set of parameters have to be taken in consideration in order to obtain the best outcome possible. In this case, 7 parameters can be manipulated (see figure 4.3).

Figure 4.3 - Snake Parameters configuration.

- Snake iterations: defines the number of iterations the snake algorithm is executed until the contour is obtained;
- Min Resolution: defines the minimum distance between two points from the snake. If this distance is lower than the value of this parameter, one of those points is removed;
- Max Resolution: defines the maximum distance between two points from the snake. If this distance is higher than the value of this parameter, one of those points is removed;
- External Force Weight: defines the weight that the external forces have on the snake contour;
- Viscosity: defines the viscosity degree of the snake contour;
- Rigidity: defines the rigidity degree of the snake contour;
- Elasticity: defines the elasticity degree of the snake contour.

These values can be changed to accommodate any kind of situation.

4.3. Segmentation

This section can be viewed in figure 4.4.

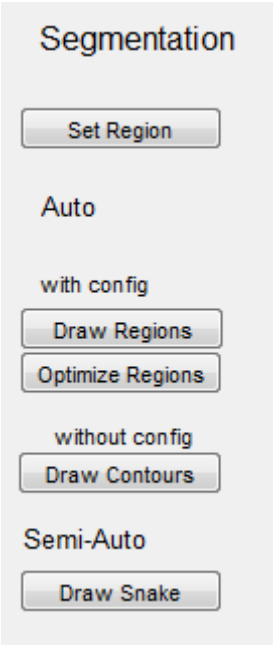

Figure 4.4 - Segmentation section.

The segmentation can be done in two ways, auto and semi-auto. In the auto section, the user can choose between the options with config or without config. The buttons and their functionalities are as follows:

- Set Region: this allows the user to define the initial set of points from where the snake algorithm is going to start. The points that create the region should be as close as possible from the cell boundary to obtain better results. When the user is finished selecting points, use the button again (known as Finish Region) to create the region.
- Draw Regions: this button is if the sequence has a pre-determined set of initial points; the user can load a config and the snake algorithm will be applied to all cells within the image.
- Optimize Regions: this allows the user, after running the Draw Regions function, to run the snake algorithm in a single cell, in order to obtain better results.
- Draw Contours: this button will, after obtaining the initial region through the Set Region function, execute the snake algorithm in a sequence of images, using the previous result to start the algorithm in the next image.
- Draw Snake: this button allows the user to run the snake algorithm manually. This functionality is normally used in a single image or to make some adjustments to a determined region.

4.4. Region properties

The region properties (figure 4.5) are calculated automatically every time a new contour is obtained.

Figure 4.5 - Region properties.

These, explained in 3.6, are:

- Area: gives the value of the area of the snake contour;
- Perimeter: gives the value of the perimeter of the snake contour;
- Centroid: gives the coordinates X and Y of the centroid of the snake contour;
- R1: gives the length of the horizontal segment of the snake contour;
- R2: gives the length of the vertical segment of the snake contour;
- Theta: gives the angle between R1 and the horizontal axis.

4.5. Contrast and brightness adjustment

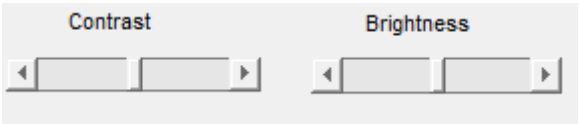

Figure 4.6 - Contrast and Brightness adjustment.

In this section it is possible, as the name indicates, to control the contrast and brightness of the current image in order to obtain better results than the original one.

4.6. Contour adjustment and image saving

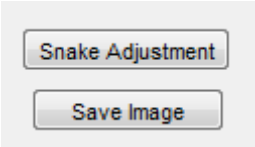

Figure 4.7 - Contour adjustment and image saving.

There are two possibilities in this section:

- Snake adjustment: this button allows the user to adjust the contour through the variation of a cubic Bézier curve. This procedure begins by selecting two points, from the boundary of the contour, in order to obtain the segment to be adjusted. After that, the curve is drawn onto the image and can be manipulated by the mouse cursor. Once the

user is satisfied with the result, just click on the mouse in order to substitute the old contour with the new curve.

- Image saving: this button allows the user to save the current image and its contour to a JPEG file.

Chapter 5

5. Results analysis

The testing of this application occurred in two different ways. In the first test, 5 sets of 12 images were analyzed, where the evolution of a cell can be seen along the set. The first one is the original image sequence, without any modifications (figure 5.1).

Figure 5.1 - Cell image without modifications.

The next two sequences are a result of contrast modification.

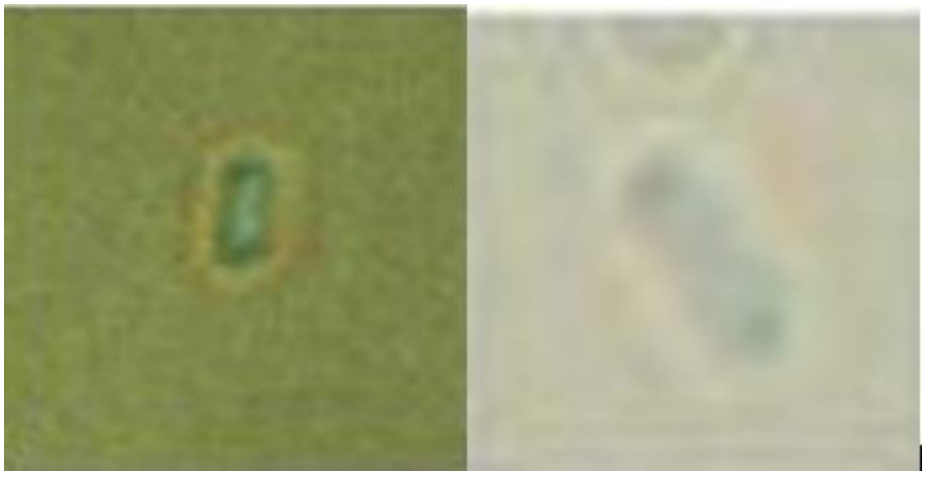

Figure 5.2 - Cell images with contrast modification.

The final two sequences are a result of brightness modification.

Figure 5.3 - Cell images with brightness modification.

5.1. First set – original image sequence

In figure 5.4 it is possible to see the evolution of the cell in each frame.

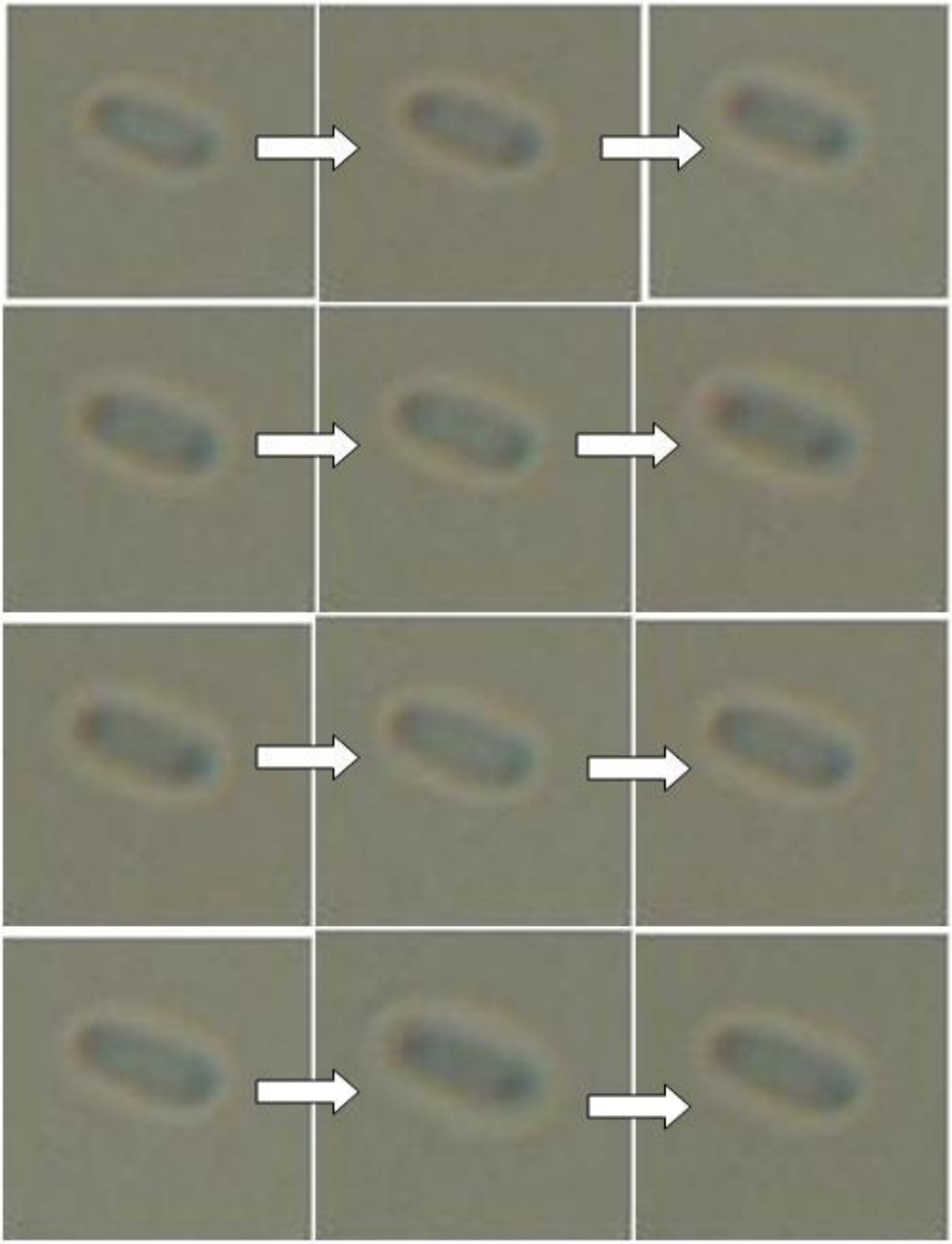

Figure 5.4 - First set of images.

In figure 5.5 it is possible to see the result of the snake algorithm.

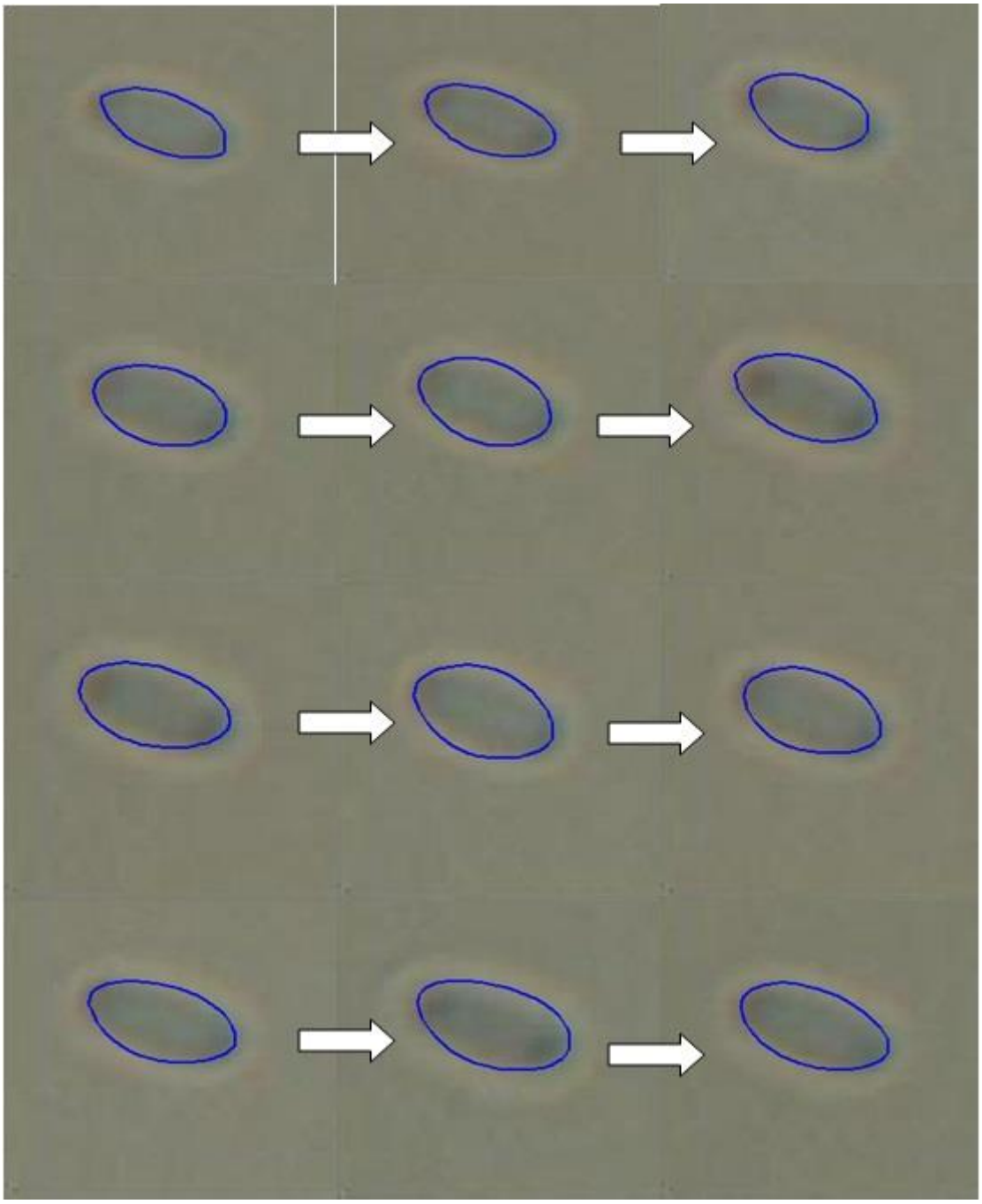

Figure 5.5 - First set of images with contours.

In the first three images the contour is done perfectly; however, in the next images the contour does not converge to the cell boundary, near the edges. This happens because the edges of the cells are darker, introducing more noise to that zone.

After adjusting the contours manually the results in figure 5.6 are obtained.

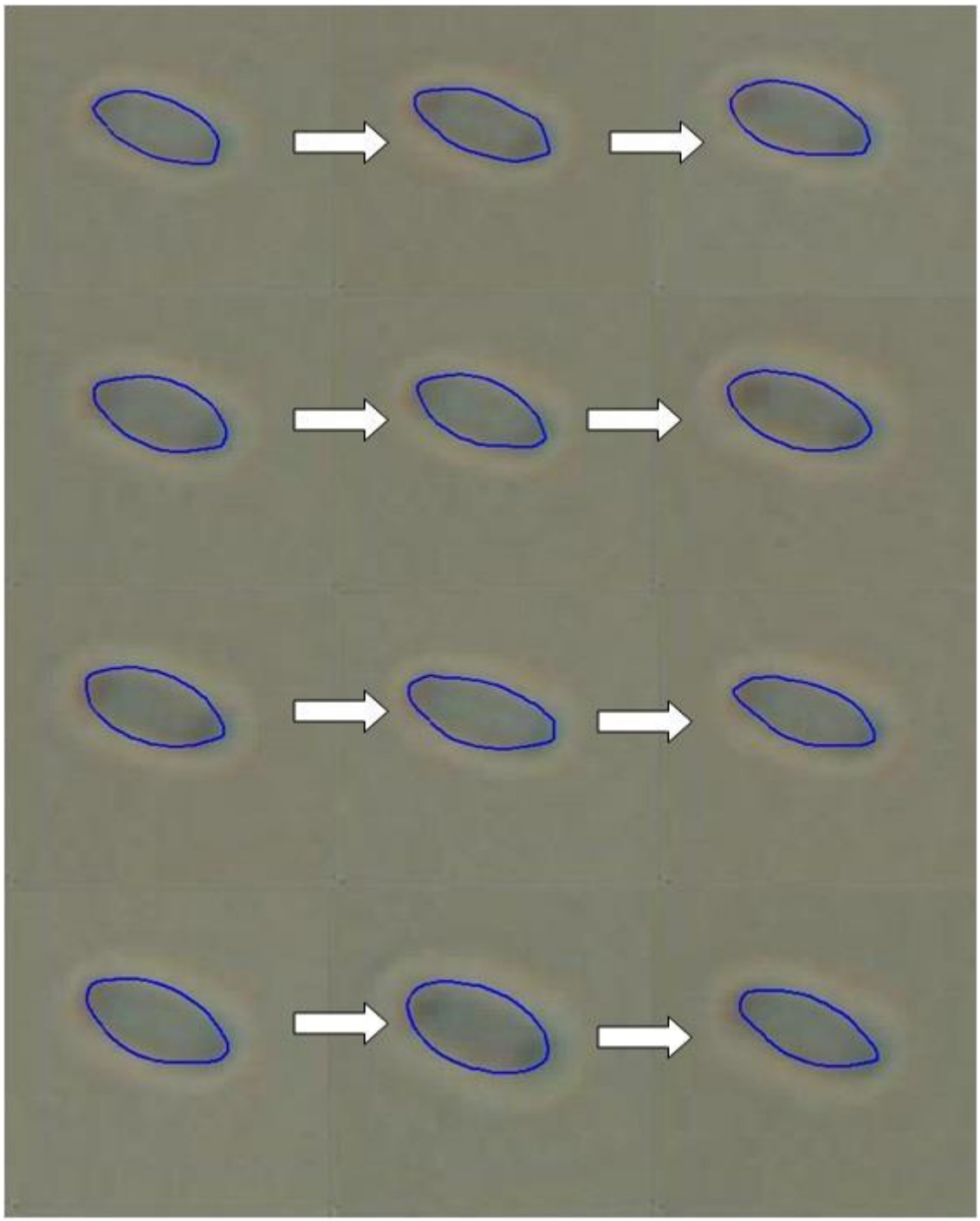

Figure 5.6 - First set of images with adjusted contours.

5.2. Second set – Cell image sequence with higher contrast

In figure 5.7 it is possible to see another cell image sequence; however now the image has a higher level of contrast in comparison with the original sequence.

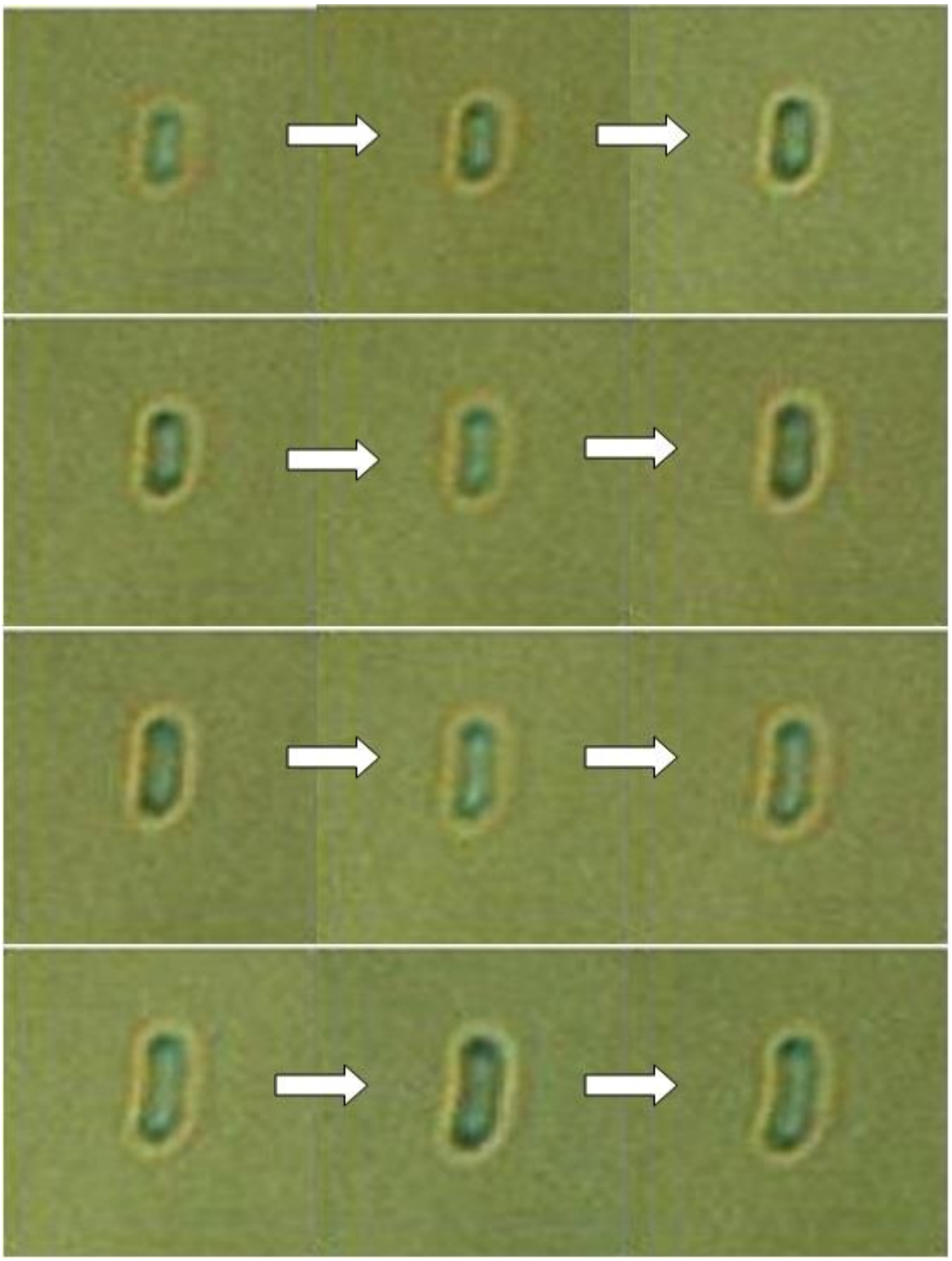

Figure 5.7 - Second set of images.

In figure 5.8 it is possible to see the result from applying the snake algorithm.

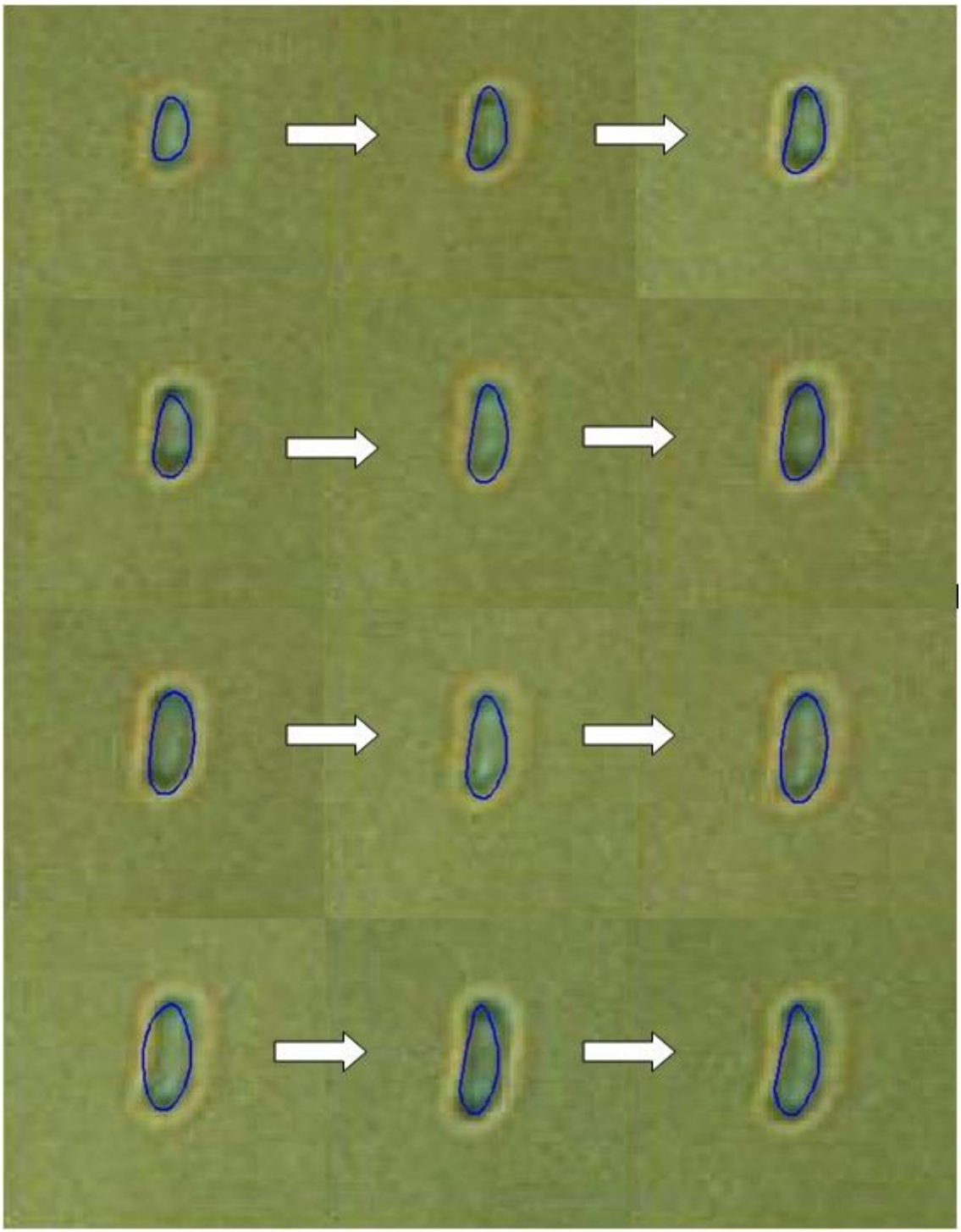

Figure 5.8 - Second set of images with contours.

As it is possible to see in figure 5.8, the snake algorithm gives good results in the beginning, but as the frames advance, the contour starts to collapse, primarily in the edges of the cell. This happens because of the noise introduced by the high level of contrast of these images.

After adjusting manually the contour, it is possible to obtain the results in figure 5.9.

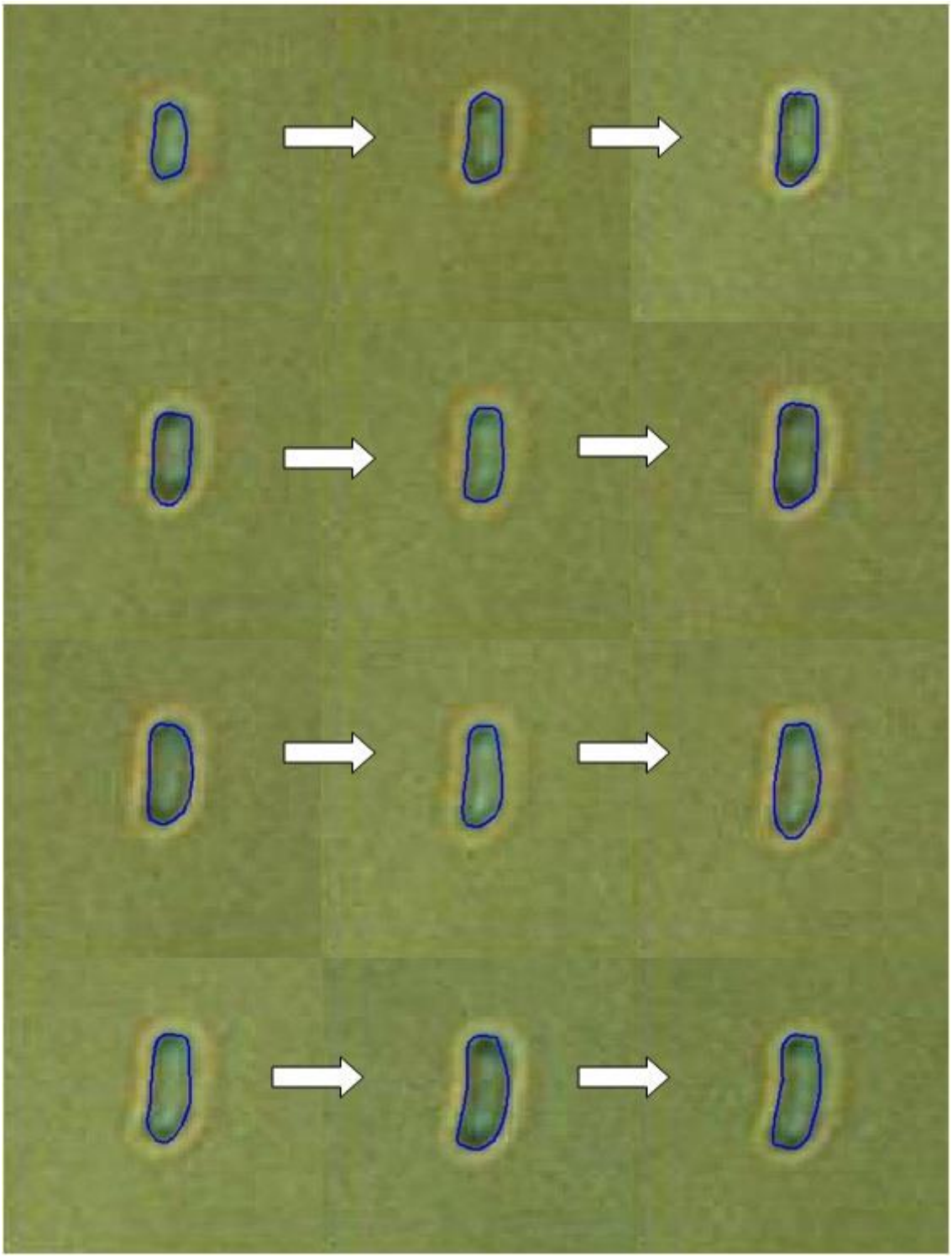

Figure 5.9 - Second set of images with adjusted contours.

5.3. Third set – cell image with lower contrast

In figure 5.10 it is possible to see another cell image sequence; however now the image has a lower level of contrast in comparison with the original sequence.

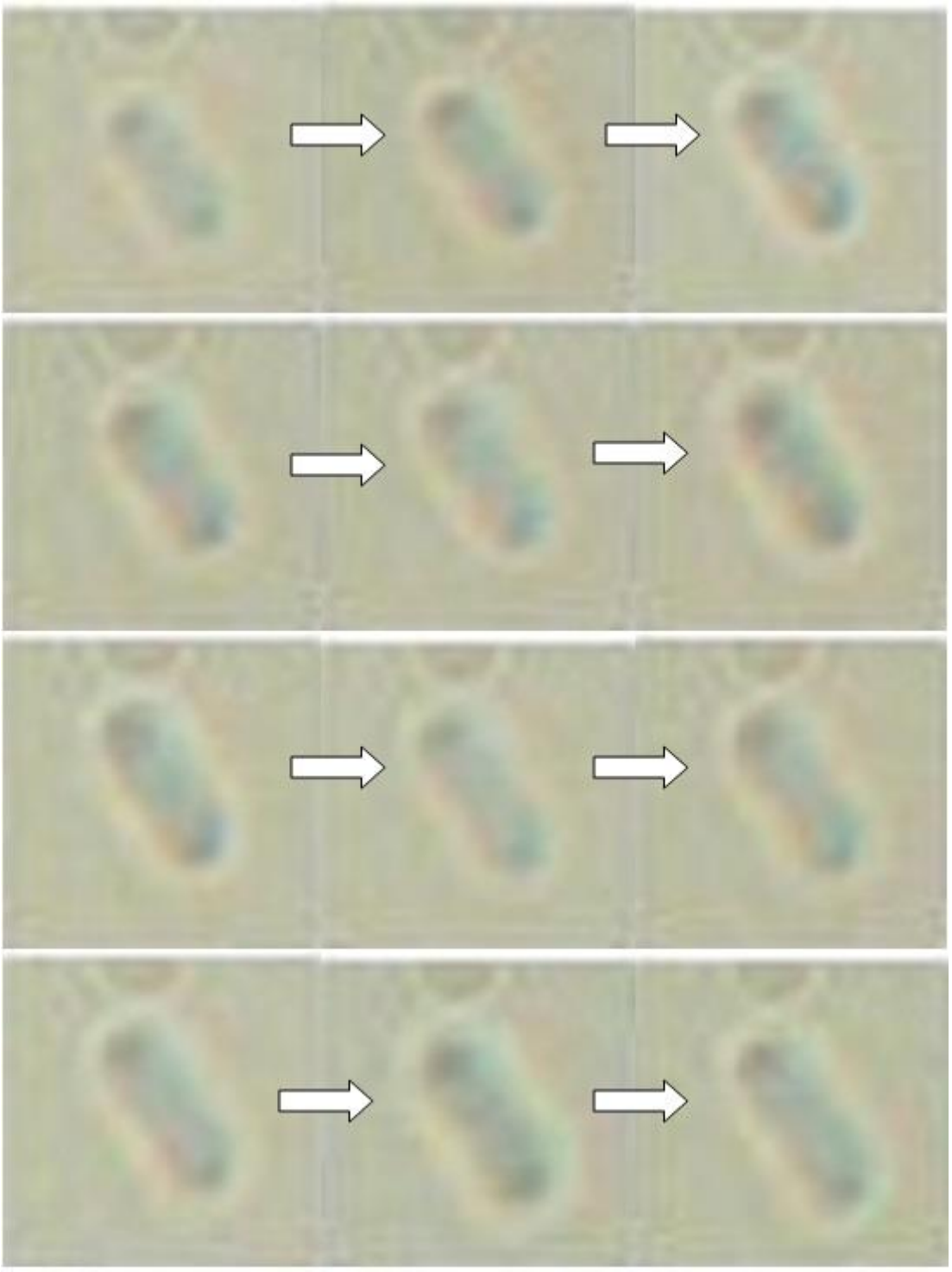

Figure 5.10 - Third set of images.

In figure 5.11 it is possible to see the result from applying the snake algorithm.

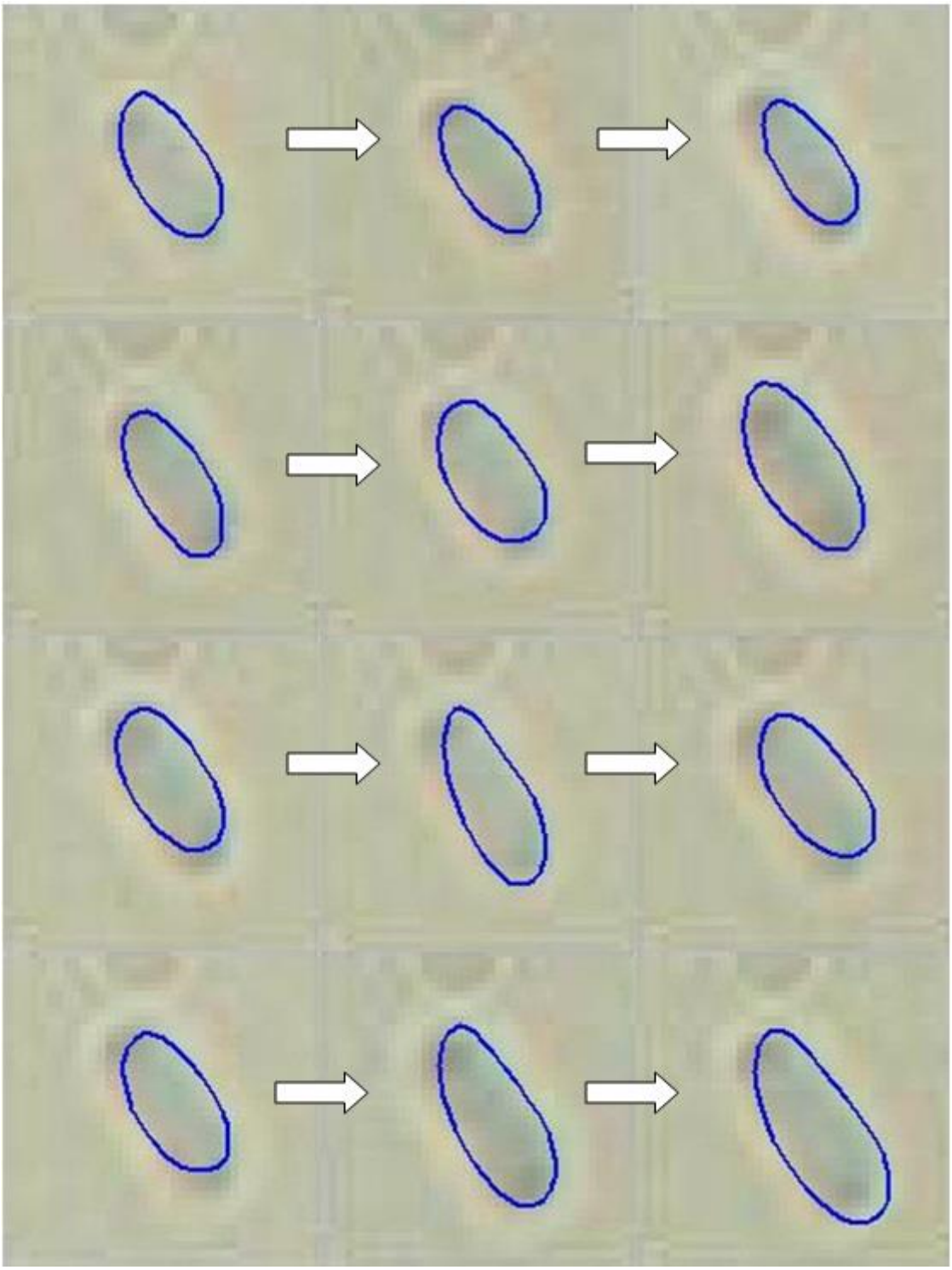

Figure 5.11 - Third set of images with contous.

In this sequence, the snake algorithm has some problems regarding the position of the cell. Since the level of contrast is minimal, the boundary of the cell is harder to detect, making the snake algorithm to present results in different sizes.

After adjusting manually the contour, it is possible to obtain the results in figure 5.12.

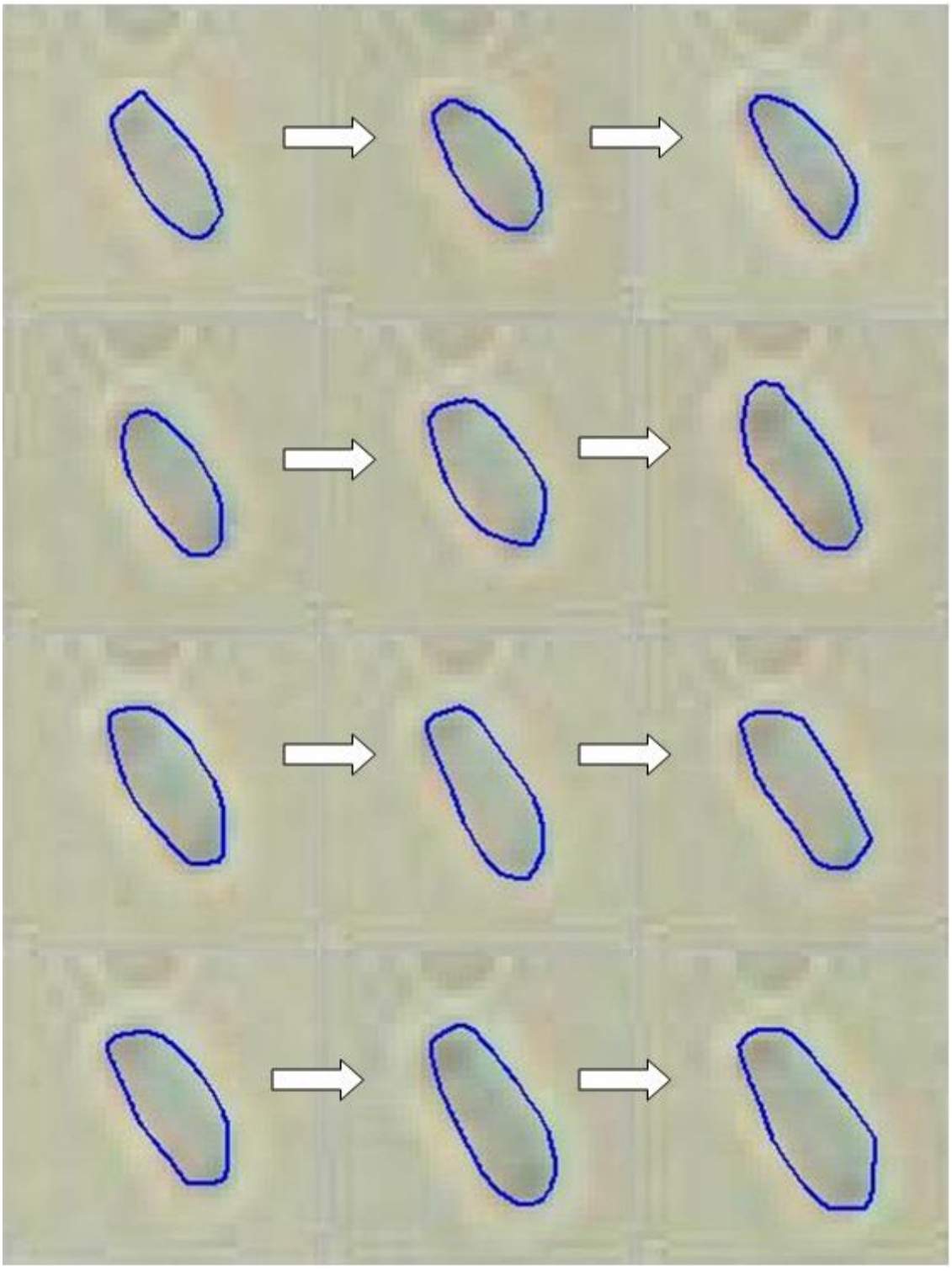

Figure 5.12 - Third set of images with adjusted contours.

5.4. Fourth set – cell image sequence with lower brightness

In figure 5.12 it is possible to see another cell image sequence; however now the image has a lower level of brightness in comparison with the original sequence.

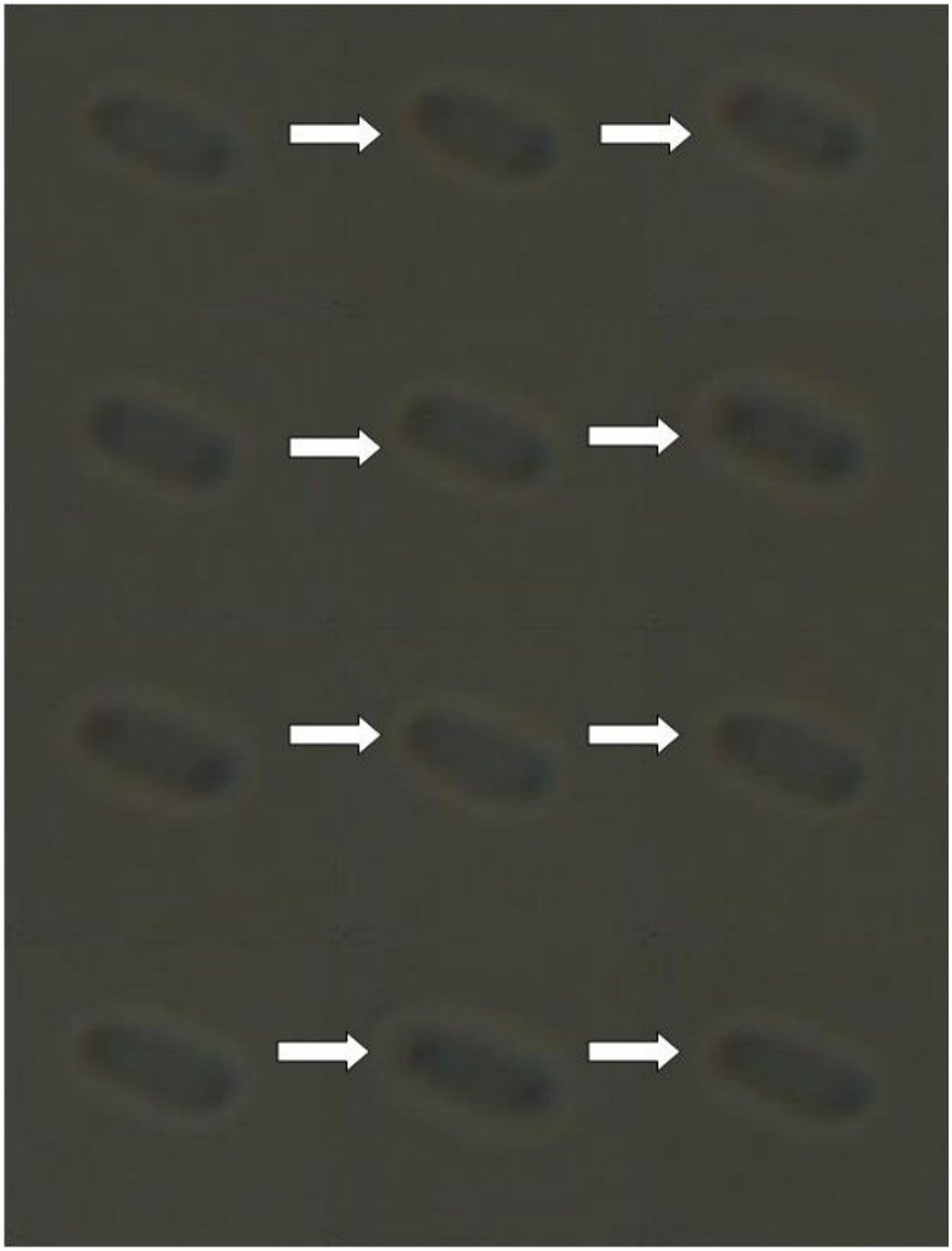

Figure 5.13 - Fourth set of images.

In figure 5.13 it is possible to see the result from applying the snake algorithm.

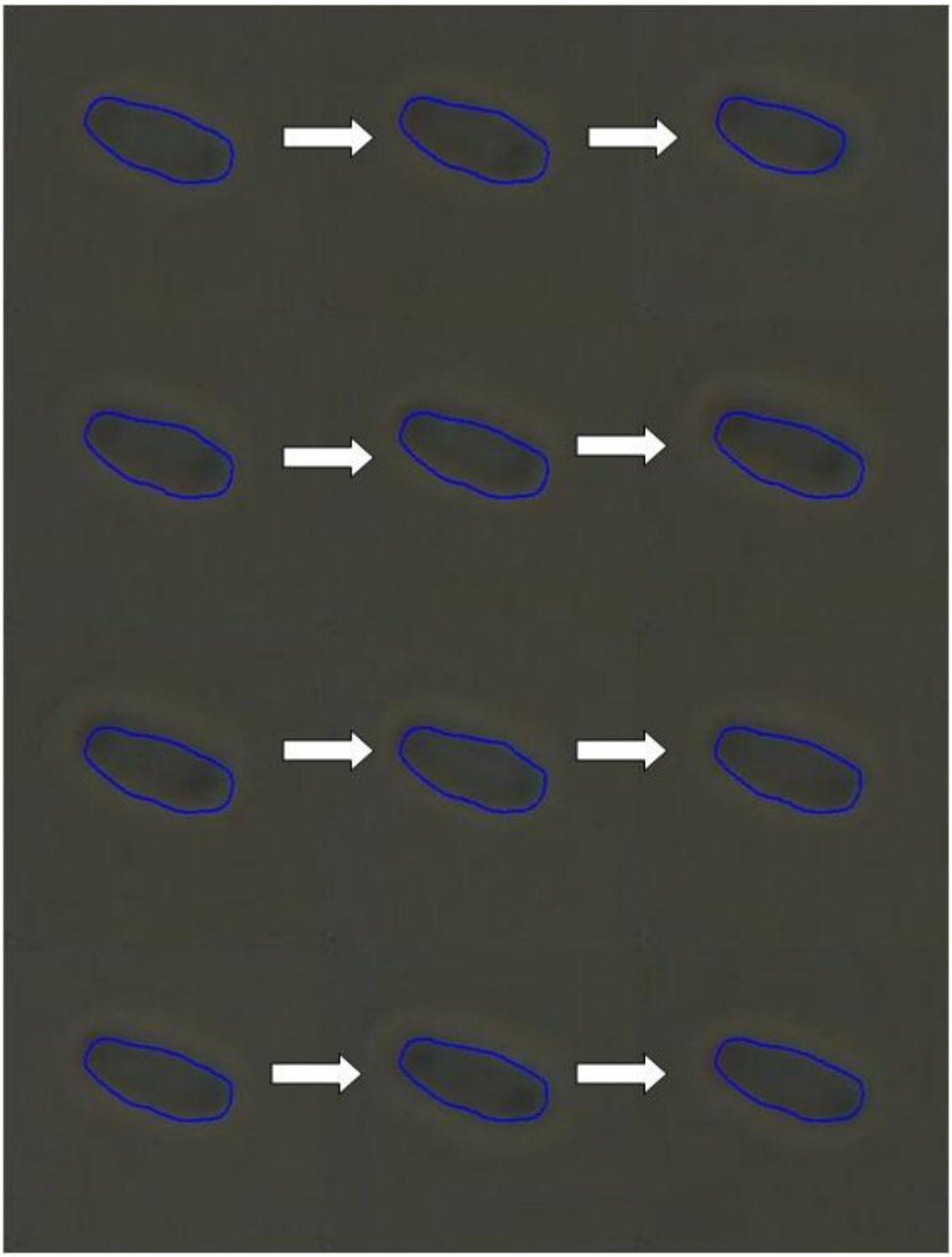

Figure 5.14 - Fourth set of images with contours.

This set gives better results than the previous ones, even though there are some deviations from the cell boundary. This happens because, since the brightness level is too low,

the cell membrane acts as a differentiator between the actual cell and the surrounding environment, making the contour to be easier to find.

After adjusting manually the contour of some images, it is possible to obtain the results in figure 5.14.

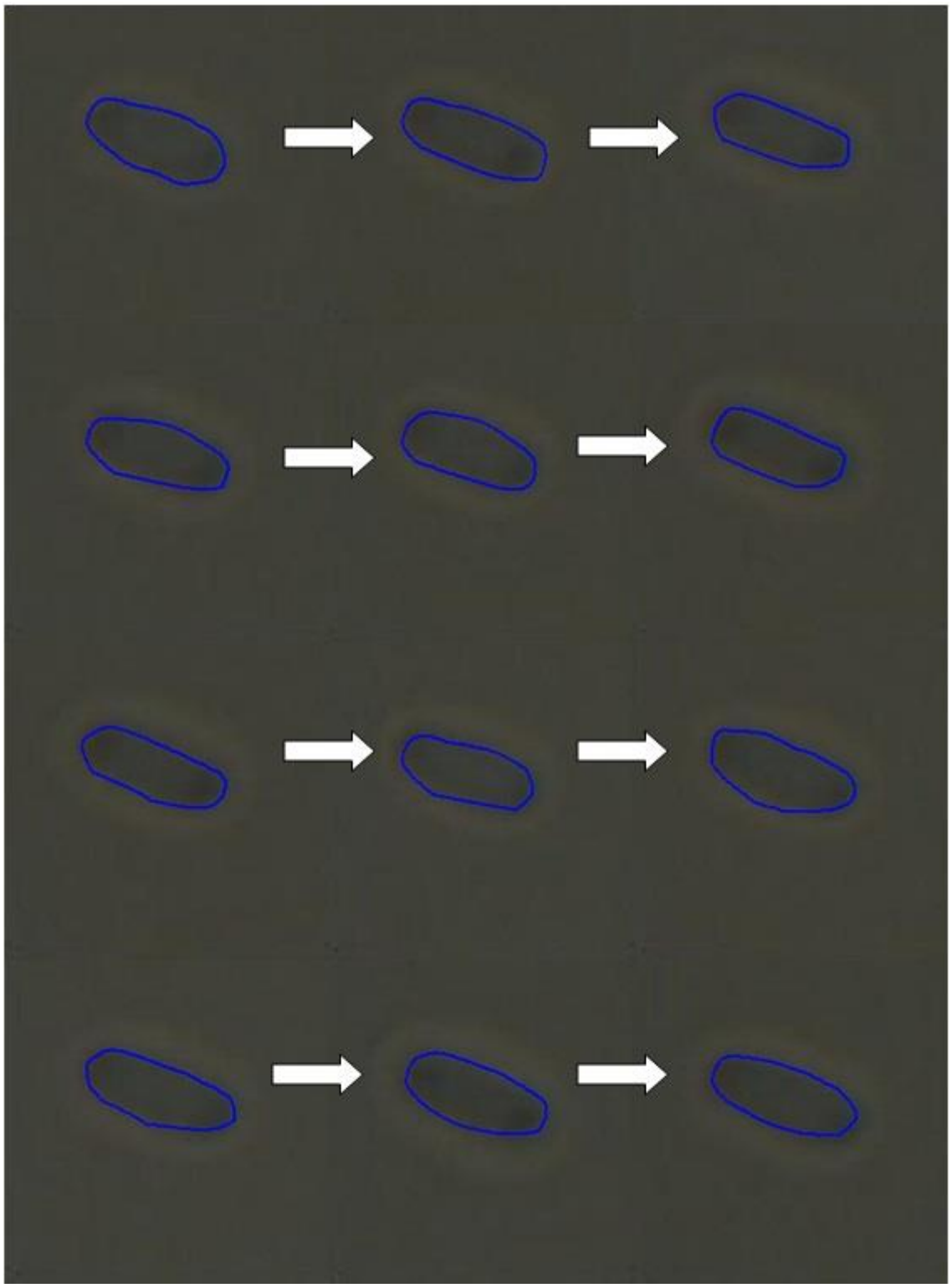

Figure 5.15 - Fourth set of images with adjusted contours.

5.5. Fifth set – cell image sequence with higher brightness

In figure 5.15 it is possible to see another cell image sequence; however now the image has a higher level of brightness in comparison with the original sequence.

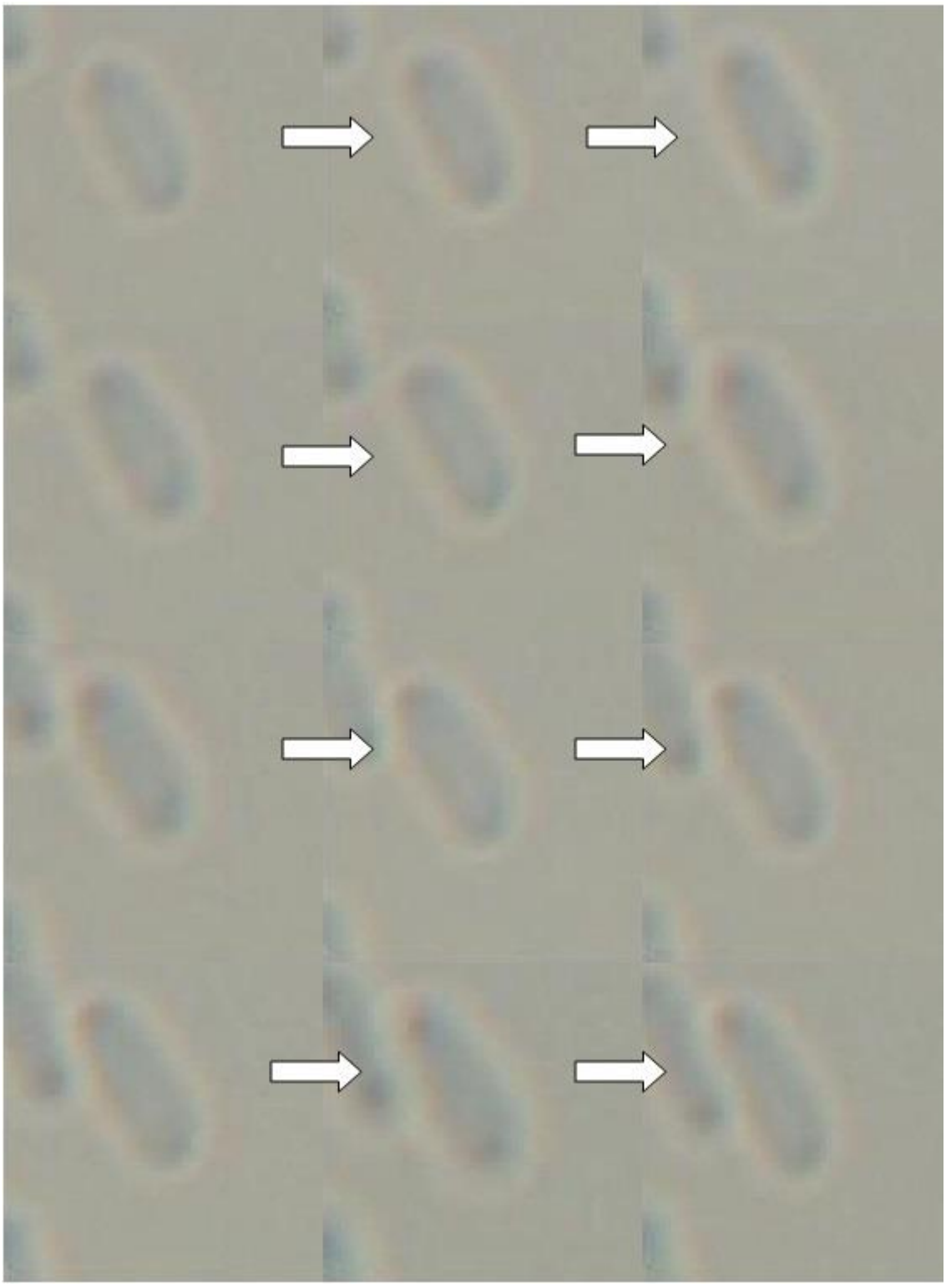

Figure 5.16 - Fifth set of images.

In figure 5.16 it is possible to see the result from applying the snake algorithm.

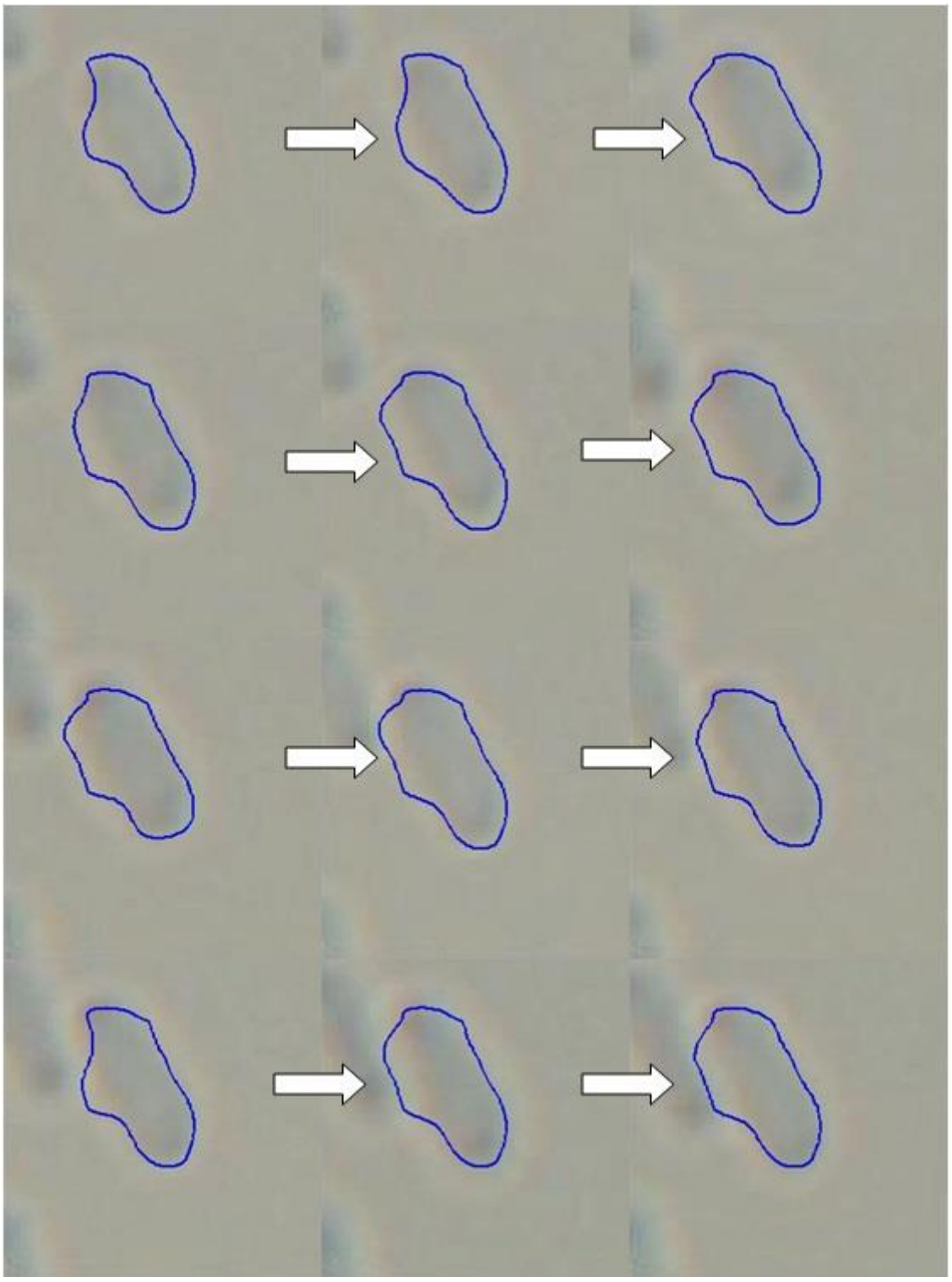

Figure 5.17 - Fifth set of images with contours.

In this set, it is possible to see a limitation of this algorithm. The contour on the left part of the cell is always out of position. The reason this happens is that, as the frames pass, another

cell in the surrounding environment starts to get closer to the central one, which makes the snake algorithm not knowing whether it goes to one side or the other. Apart from that, the rest of the contour presents excellent results, thanks to the lighter background.

After adjusting manually the contours, it is possible to obtain the results in figure 5.17.

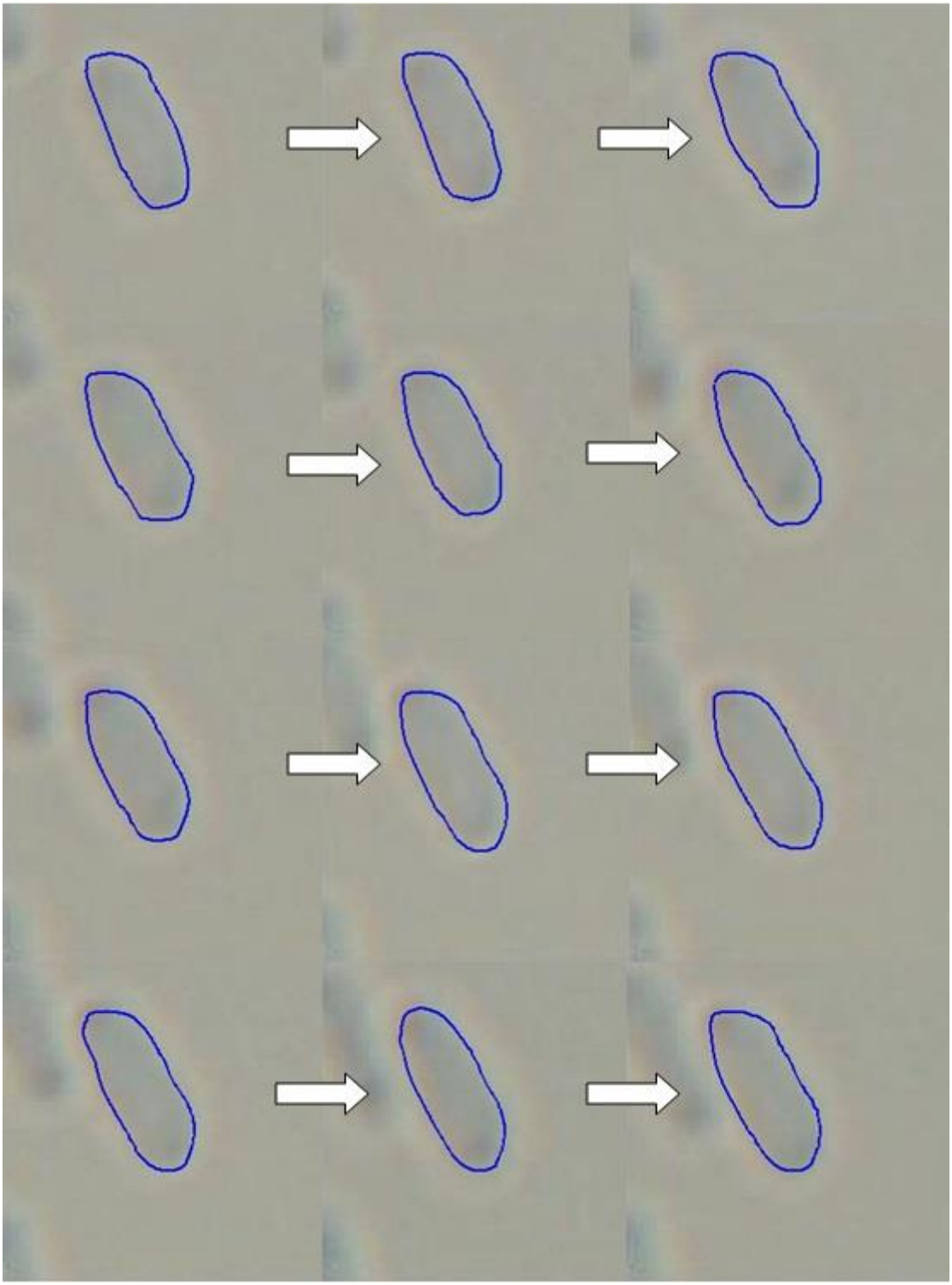

Figure 5.18 - Fifth set of images with adjusted contours.

5.6. Test results

The second test consisted in analyzing 50 images, each one with one cell. The first stage of the test was made manually, and the results were considered has the perfect segmentation in order to test the algorithm. The second stage was running the algorithm through all the images. Finally, the results were compared and the summary of the test can be seen in table 5.1.

Table 5.1 – Average error values.

Region properties	**Difference (Pixels)**	**Error Percentage (%)**
Area	6,7766	1,9845
Perimeter	8,6368	10,6864
Centroid X	-0,3960	-1,9394
Centroid Y	-0,0212	-0,1497

The 3 region properties taken in consideration were area, perimeter and the coordinates of the centroid.

The difference column represents the difference between the contour made manually and the contour made by the algorithm. The percentage column gives us an idea what the average percentage error between those two measurements.

In the appendix, all test values can be seen in tables 1 and 2.

Chapter 6

6. Conclusions and future work

The active contours have been one of the most used techniques to solve segmentation problems. The difficulty, in this kind of problems, resides in the size of the sample, in the complexity and various forms that the regions have, as well as the quality of the image where the regions are inserted. In these last years, various techniques have been proposed in order to surpass some of the limitations presented in older versions, such as autonomy, user interface and noise tolerance.

In this work, an algorithm was presented in order to obtain better results in cell image segmentation, with the assistance of a human user. Even though a semi-automatic is not the perfect situation, it makes the application to be more tolerant to any kind of situation. On one hand, it allows to obtain results at a faster rate than a totally manual segmentation. On the other hand, it has the potential to give better results than purely automatic applications, since the user can alter the final result, adjusting the contour to the correct position with a quick and simple procedure.

The results obtained shows that the snake GVF used, even though it is efficient, it presents some limitations, such as the manual adjustment of the contour in some cases. This happens because the boundary of the cell is not defined properly or the image has a lot of noise. However, the adjustment is done rapidly, and the results improved.

Even though the active contours have evolved, it will be hard to develop a technique that will be completely tolerant to a great variety of images and regions, since these kinds of applications are developed to focus on a certain set of images. If the images introduced are significantly different from the default set, the technique will basically start behaving strangely. To address this problem, the use of different techniques is the best solution. It will make the application more “open” to other images, even though there will always exist certain kind of situations where it is impossible to obtain good results.

As future work, it is recommended the study of this algorithm when applied in other circumstances, in order to grasp a greater variety of situations, as well as the integration with other techniques, reducing the need to do manual adjustments. Another particularity to be developed is the segmentation of two or more regions within one image. The algorithm used in this work presents a lot of limitations when these kinds of situations arise. Finally, the migration

of this application to another language, such as C++ or C#, would be a plus, in order to obtain a more powerful and quick application.

References

[1] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *Int. J. Comput. Vis.*, vol. 1, no. 4, pp. 321–331, 1988.

[2] C. Xu and J. L. Prince, “Gradient vector flow: a new external force for snakes,” *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. pp. 66–71, 1997.

[3] C. Xu and J. L. Prince, “Snakes, shapes, and gradient vector flow,” *Image Processing, IEEE Transactions on*, vol. 7, no. 3. pp. 359–369, 1998.

[4] J. S. Silva, B. S. Santos, A. Silva, and J. Madeira, “Modelos Deformáveis na Segmentação de Imagens Médicas : uma introdução,” 2004.

[5] G. Ongun, U. Halici, K. Leblebicioglu, V. Atalay, M. Beksac, and S. Beksac, “An automated differential blood count system,” *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, vol. 3. pp. 2583–2586 vol.3, 2001.

[6] T. W. Nattkemper, W. Schubert, T. Hermann, and H. Ritter, “A Hybrid System for Cell Detection in Digital Micrographs,” 2004, vol. 417.

[7] T. W. Nattkemper, H. Wersing, W. Schubert, and H. Ritter, “A neural network architecture for automatic segmentation of fluorescence micrographs,” *Neurocomputing*, vol. 48, no. 1–4, pp. 357–367, Oct. 2002.

[8] H. Wersing, J. J. Steil, and H. Ritter, “A Competitive-Layer Model for Feature Binding and Sensory Segmentation,” *Neural Comput.*, vol. 13, no. 2, pp. 357–387, Feb. 2001.

[9] M. Martín-Fernández and C. Alberola-López, “An approach for contour detection of human kidneys from ultrasound images using Markov random fields and active contours,” *Med. Image Anal.*, vol. 9, no. 1, pp. 1–23, Feb. 2005.

[10] B. Fang, W. Hsu, and M. L. Lee, “Tumor Cell Identification Using Features Rules,” in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002, pp. 495–500.

[11] O. Sliusarenko, J. Heinritz, T. Emonet, and C. Jacobs-Wagner, “High-throughput, subpixel precision analysis of bacterial morphogenesis and intracellular spatio-temporal dynamics,” *Mol. Microbiol.*, vol. 80, no. 3, pp. 612–627, 2011.

[12] J. W. Young, J. C. W. Locke, A. Altinok, N. Rosenfeld, T. Bacarian, P. S. Swain, E. Mjolsness, and M. B. Elowitz, “Measuring single-cell gene expression dynamics in bacteria using fluorescence time-lapse microscopy.,” *Nat. Protoc.*, vol. 7, no. 1, pp. 80–8, 2012.

[13] M. Lamprecht, D. M. Sabatini, and A. E. Carpenter, “CellProfiler: free, versatile software for automated biological image analysis.,” *Biotechniques*.

[14] Wikipedia, “Gaussian Blur.” [Online]. Available: http://en.wikipedia.org/wiki/Gaussian_blur. [Accessed: 16-Oct-2013].

[15] P. Mlsna and J. Rodríguez, “Gradient and Laplacian Edge Detection,” in in *Handbook of Image and Video processing*, Second., Academic Press, 2005, pp. 535–553.

[16] B. M. Dawant and A. P. Zijdenbos, “Image segmentation,” in in *Handbook of Medical Imaging, vol. 2*, Bellingham, WA, 2000.

[17] A. Singh, D. Terzopoulos, and D. B. Goldgof, *Deformable Models in Medical Image Analysis*, 1st ed. Los Alamitos, CA, USA: IEEE Computer Society Press, 1998.

[18] R. Ďurikovič, K. Kaneda, and H. Yamashita, “Dynamic contour: A texture approach and contour operations,” *Vis. Comput.*, vol. 11, no. 6, pp. 277–289, 1995.

[19] T. Drummond and R. Cipolla, “Real-time visual tracking of complex structures,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7. pp. 932–946, 2002.

[20] R. Courant, “Methods of Mathematical Physics,” *American Journal of Physics*, vol. 22, no. 4. p. 241, 1954.

[21] H. J. Sommer, “POLYGEOM Geometry of a planar polygon.” 1998.

[22] Wikipedia, “Euclidian distance.” [Online]. Available: http://en.wikipedia.org/wiki/Euclidean_distance. [Accessed: 10-Dec-2013].

[23] Wikipedia, “Bézier curve.” [Online]. Available: http://en.wikipedia.org/wiki/B%C3%A9zier_curve. [Accessed: 15-Dec-2013].

Appendix

1. Test Results

Image	Area (manual)	Perimeter (manual)	Centroid X (manual)	Centroid Y (manual)	Area (snake)	Perimeter (snake)	Centroid X (snake)	Centroid Y (snake)
1	496,8938	109,0007	39,0371	26,3417	471,2897	97,8288	38,4634	26,6431
2	332,446	81,6072	28,4094	25,1295	308,4493	70,7096	28,5224	24,8456
3	334,2061	78,7897	28,056	21,6566	300,6625	71,0662	27,4648	20,8288
4	379,2811	85,1232	18,038	28,5512	369,8856	77,8462	18,1582	28,2262
5	346,6421	86,8131	21,723	30,456	321,0648	76,2087	21,9454	30,1662
6	275,9238	74,1781	24,3367	20,8635	273,2487	66,3738	24,808	20,9273
7	331,4858	82,5409	21,8702	22,6153	295,9779	71,7486	22,7723	22,1907
8	424,1809	101,3395	32,2522	24,2601	424,8547	91,1638	32,7643	23,8822
9	429,52	97,3895	22,7379	29,8923	429,6476	87,2046	22,9359	29,9675
10	406,1666	96,8132	16,2697	30,5782	377,6461	82,8446	16,4578	30,2099
11	362,9123	93,4716	23,0118	26,615	363,9411	84,2559	23,3593	26,3808
12	337,4708	83,6575	22,8589	21,9173	336,3815	72,2698	23,6321	22,8094
13	471,3944	101,689	28,892	28,8767	444,4065	90,1565	30,6114	28,8956
14	202,0967	64,4997	19,7622	21,499	215,2277	59,3295	20,55	22,3621
15	311,1371	83,4091	24,4773	22,2012	310,1116	75,0913	24,709	22,1218
16	409,5046	102,4289	17,5455	31,3785	414,1815	93,5217	17,8488	31,8855
17	170,7281	54,1497	16,335	21,8121	171,3667	50,2564	16,6138	21,1114
18	205,8923	58,8387	14,6367	21,3413	200,0468	54,1146	15,29	21,571
19	213,8904	58,9645	19,8904	20,3621	200,7646	52,9407	20,0297	19,8211
20	266,2729	74,1075	24,9686	20,4283	243,7814	64,924	25,1388	20,6601
21	189,6172	59,9395	18,2136	19,4662	187,2903	53,1386	18,3739	19,5223
22	235,9074	74,4874	20,7505	21,5237	218,2373	64,3506	21,4782	21,189
23	202,4298	63,9747	19,691	19,7681	193,8451	56,9251	20,4396	19,2247

24	292,0064	72,0584	17,6173	24,7143	278,063	65,1032	17,989	24,0595
25	227,8184	65,438	15,7794	20,6922	221,6687	59,3727	16,2647	20,6334
26	205,7565	57,9402	19,5831	17,0205	188,4174	53,4561	20,2868	17,0658
27	149,2013	49,0347	17,3854	16,6007	131,6998	42,5307	17,7153	16,9993
28	160,2525	50,9024	16,3882	19,224	152,9034	46,5852	17,0214	19,3154
29	221,8877	68,7284	21,1166	21,7973	245,9952	63,4685	22,7237	21,7161
30	371,46	97,1342	16,1303	27,6755	371,9084	85,7432	16,3851	27,4045
31	379,5834	94,9165	24,2166	32,5202	380,6091	85,1449	24,6292	32,2482
32	293,0804	86,5469	23,0532	26,3325	315,5564	79,0432	23,4604	26,5434
33	202,478	62,564	19,5274	21,3909	204,0012	57,4137	20,1617	20,9964
34	206,8015	67,188	20,0359	19,181	203,6904	58,3853	20,7559	19,6441
35	304,3648	80,5292	24,0089	23,9275	304,8765	71,4798	24,4136	24,2448
36	376,9027	97,5217	21,0391	31,0375	375,9198	88,8959	21,3471	31,7525
37	346,272	96,0759	15,5057	31,0675	338,9723	86,3934	15,6136	32,5818
38	141,0773	55,5086	23,1688	15,9678	172,0032	51,2782	22,8373	16,5829
39	285,1444	85,3037	26,9131	21,5804	289,0085	76,0465	26,6601	22,1835
40	243,0912	72,6939	24,6303	18,7565	223,3412	62,4511	25,1092	18,9445
41	314,376	86,774	20,6849	26,6739	318,6947	77,606	20,3995	27,1431
42	281,3865	77,814	22,1227	20,3032	274,3154	66,2632	23,0924	20,5537
43	364,7989	91,9665	17,3448	28,3368	351,9175	80,1595	17,8364	28,8485
44	311,7258	81,1433	19,4219	27,9577	307,3991	75,2279	20,3725	26,1284
45	347,9205	93,0366	20,0559	28,2855	333,1445	78,923	20,6087	27,8595
46	257,5188	87,8241	19,7435	27,1368	257,5631	77,0977	19,3786	27,3879
47	353,6412	96,8736	33,8895	25,9652	347,7866	86,8802	34,1167	26,16
48	301,2333	88,4465	24,4649	27,4059	301,3416	79,727	24,6739	27,1241
49	446,085	106,1064	33,6364	17,1204	425,9588	94,2842	34,0305	18,192
50	243,9358	69,0111	20,7302	18,8108	237,9051	61,2254	21,5055	18,3232

2. Differences between manual and semi-automatic

Image	Diff Area	Area %	Diff Perimeter	Perimeter %	Diff Centroid X	Centroid X %	Diff Centroid Y	Centroid Y %
1	25,6041	5,1528	11,1719	10,2494	0,5737	1,4696	-0,3014	-1,1442
2	23,9967	7,2182	10,8976	13,3537	-0,1130	-0,3978	0,2839	1,1297
3	33,5436	10,0368	7,7235	9,8027	0,5912	2,1072	0,8278	3,8224
4	9,3955	2,4772	7,2770	8,5488	-0,1202	-0,6664	0,3250	1,1383
5	25,5773	7,3786	10,6044	12,2152	-0,2224	-1,0238	0,2898	0,9515
6	2,6751	0,9695	7,8043	10,5210	-0,4713	-1,9366	-0,0638	-0,3058
7	35,5079	10,7117	10,7923	13,0751	-0,9021	-4,1248	0,4246	1,8775
8	-0,6738	-0,1588	10,1757	10,0412	-0,5121	-1,5878	0,3779	1,5577
9	-0,1276	-0,0297	10,1849	10,4579	-0,1980	-0,8708	-0,0752	-0,2516
10	28,5205	7,0219	13,9686	14,4284	-0,1881	-1,1561	0,3683	1,2045
11	-1,0288	-0,2835	9,2157	9,8594	-0,3475	-1,5101	0,2342	0,8800
12	1,0893	0,3228	11,3877	13,6123	-0,7732	-3,3825	-0,8921	-4,0703
13	26,9879	5,7251	11,5325	11,3410	-1,7194	-5,9511	-0,0189	-0,0655
14	-13,1310	-6,4974	5,1702	8,0159	-0,7878	-3,9864	-0,8631	-4,0146
15	1,0255	0,3296	8,3178	9,9723	-0,2317	-0,9466	0,0794	0,3576
16	-4,6769	-1,1421	8,9072	8,6960	-0,3033	-1,7286	-0,5070	-1,6158
17	-0,6386	-0,3740	3,8933	7,1899	-0,2788	-1,7068	0,7007	3,2124
18	5,8455	2,8391	4,7241	8,0289	-0,6533	-4,4634	-0,2297	-1,0763
19	13,1258	6,1367	6,0238	10,2160	-0,1393	-0,7003	0,5410	2,6569
20	22,4915	8,4468	9,1835	12,3921	-0,1702	-0,6817	-0,2318	-1,1347
21	2,3269	1,2272	6,8009	11,3463	-0,1603	-0,8801	-0,0561	-0,2882
22	17,6701	7,4903	10,1368	13,6087	-0,7277	-3,5069	0,3347	1,5550
23	8,5847	4,2408	7,0496	11,0194	-0,7486	-3,8017	0,5434	2,7489
24	13,9434	4,7750	6,9552	9,6522	-0,3717	-2,1099	0,6548	2,6495
25	6,1497	2,6994	6,0653	9,2688	-0,4853	-3,0755	0,0588	0,2842
26	17,3391	8,4270	4,4841	7,7392	-0,7037	-3,5934	-0,0453	-0,2661
27	17,5015	11,7301	6,5040	13,2641	-0,3299	-1,8976	-0,3986	-2,4011
28	7,3491	4,5860	4,3172	8,4813	-0,6332	-3,8638	-0,0914	-0,4754
29	-24,1075	-10,8647	5,2599	7,6532	-1,6071	-7,6106	0,0812	0,3725
30	-0,4484	-0,1207	11,3910	11,7271	-0,2548	-1,5796	0,2710	0,9792
31	-1,0257	-0,2702	9,7716	10,2949	-0,4126	-1,7038	0,2720	0,8364
32	-22,4760	-7,6689	7,5037	8,6701	-0,4072	-1,7663	-0,2109	-0,8009
33	-1,5232	-0,7523	5,1503	8,2321	-0,6343	-3,2483	0,3945	1,8442
34	3,1111	1,5044	8,8027	13,1016	-0,7200	-3,5935	-0,4631	-2,4144
35	-0,5117	-0,1681	9,0494	11,2374	-0,4047	-1,6856	-0,3173	-1,3261
36	0,9829	0,2608	8,6258	8,8450	-0,3080	-1,4639	-0,7150	-2,3037
37	7,2997	2,1081	9,6825	10,0780	-0,1079	-0,6959	-1,5143	-4,8742
38	-30,9259	-21,9212	4,2304	7,6212	0,3315	1,4308	-0,6151	-3,8521
39	-3,8641	-1,3551	9,2572	10,8520	0,2530	0,9401	-0,6031	-2,7947
40	19,7500	8,1245	10,2428	14,0903	-0,4789	-1,9444	-0,1880	-1,0023
41	-4,3187	-1,3737	9,1680	10,5654	0,2854	1,3798	-0,4692	-1,7590
42	7,0711	2,5129	11,5508	14,8441	-0,9697	-4,3833	-0,2505	-1,2338
43	12,8814	3,5311	11,8070	12,8384	-0,4916	-2,8343	-0,5117	-1,8058

44	4,3267	1,3880	5,9154	7,2901	-0,9506	-4,8945	1,8293	6,5431
45	14,7760	4,2469	14,1136	15,1699	-0,5528	-2,7563	0,4260	1,5061
46	-0,0443	-0,0172	10,7264	12,2135	0,3649	1,8482	-0,2511	-0,9253
47	5,8546	1,6555	9,9934	10,3159	-0,2272	-0,6704	-0,1948	-0,7502
48	-0,1083	-0,0360	8,7195	9,8585	-0,2090	-0,8543	0,2818	1,0282
49	20,1262	4,5117	11,8222	11,1418	-0,3941	-1,1716	-1,0716	-6,2592
50	6,0307	2,4722	7,7857	11,2818	-0,7753	-3,7400	0,4876	2,5921