Abstract

The task of Yeast cell tracking is very important for biologists because it allows them to learn the behaviour of the cells and it also helps them to build the cells' pedigree trees by extracting their ancestral information. The latter is very important in medical research because Yeast cells' ancestry is intensively used for the study of aging and also for cancer research.

This project researches and implements a novel Yeast cell detection and tracking system using Viola-jones object recognition, a domain model, and a spatio-temporal Particle Filter. The project is implemented using OpenCV and the C programming language. The project combines successively 3 areas of Computer Vision: Object Recognition, Geometrical Modelling and Object Tracking. The novelty of the project lies in that fact that to our knowledge no one has employed Viola-jones and Particle Filters for cell recognition and tracking. And, additionally we have devised a novel cell model establishment and validation heuristic that has not been proposed before.

We found that the performance of the tracking system improved significantly after each of the stages of the system (i.e. cell detection, model establishment and validation, Particle Filter tracking). We concluded that a combination of Viola-Jones and Particle Filters can be used for an effective detection and tracking of Yeast cells; and the latter could be used as a basis for later pedigree tree extraction.

Table of Contents

1. Introduction

The tracking of biological cells is a prerequisite to understanding their behaviour and to build their ancestry [1]. Therefore, their tracking should be reliable enough not only for tracking separated cells. But, it should also be able to handle overlapping or partially touching cells within a population [2]. However, to this date, the only reliable technique to achieve the latter is manual tracking. That is, biologists have to, using fusion proteins (i.e. green fluorescent proteins), manually track the cells under the microscope [2]. This approach is time consuming, tedious and not sustainable in the long term because recent developments in the field of genomic sequencing for instance has triggered an era of high-throughput biology[2].

This explosion of data has produced thousands of biological images/videos that all need to be processed. Therefore, keeping in mind the significance of cell behaviour analysis for research and discovery in biology and medicine [1], scientists have started to look for alternative approaches namely in the field of Computer Vision (CV).

In fact, feature or object tracking is one the core tasks of CV [3]. Many algorithms are available and the scientific community has already started to investigate their applications to cell tracking.

The project mainly investigates and interlinks two techniques: the Viola-jones methods (VJ) and Particle Filters (PF). The former are used for object detection tasks and they have a fast detection rate; which makes them suitable for real-time tracking but suffer from long classifier training. The latter are sequential Monte Carlo techniques which are used for object localisation estimation.

The novelty of the project lies in the fact that, to our knowledge, VJ and PF have not been used together for biological cell tracking.

1.1 Motivation and Yeast cell studies

Yeast cells are eukaryotic micro-organisms [26]; which mean that they distinguished themselves by the fact that they possess a nucleus. Yeast cells, apart from being of great importance for the food industry (i.e. fermentation of beverages or baking), play a paramount role as a model organism in modern cell biology [4]. Scientists have been using them to learn more about eukaryotes and ultimately human biology.

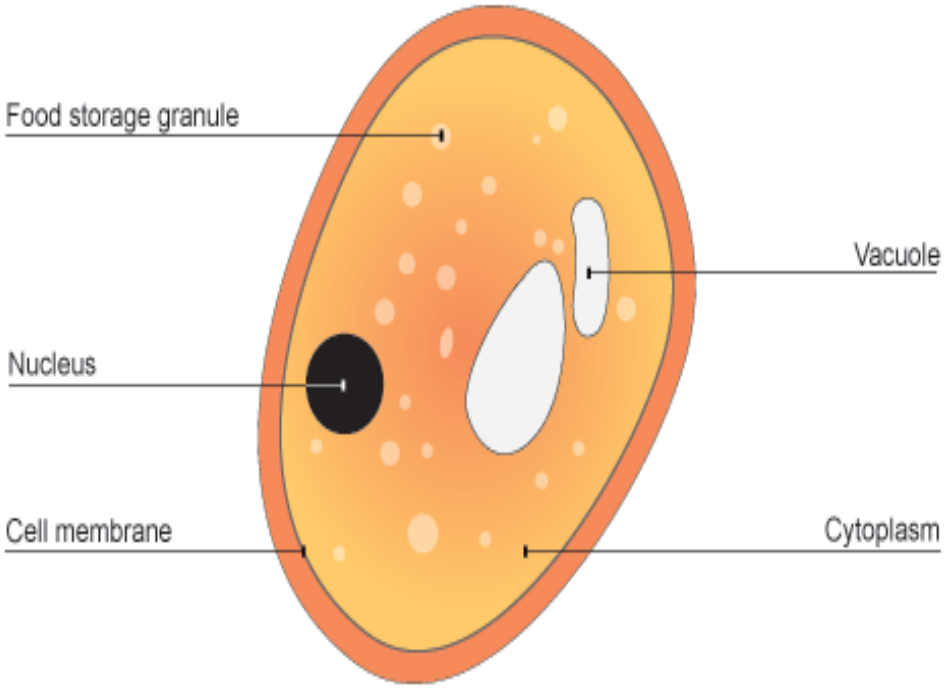

Figure 1. Diagram of a Yeast cell [43].

In addition, the reason why biologists are interested in automating the detection and the tracking of the Yeast cells is because this could help them to build the Yeast cells' pedigree trees more quickly. The pedigree tree is used to construct the ancestry of the Yeast cells. The latter can be used to understand their genetic history by, for instance, looking at the occurrence or appearance of a particular gene from one generation to other. Thus, the pedigree tree can provide information about the genotype relatedness of the Yeast cells [38],[39] - this is very useful for biologists. Furthermore, by tracking the Yeast cells, the biologists will be able to know at what point in time the different mitotic events of the pedigree tree have happened.

Moreover, the pedigree tree of the Yeast cells is used to study the genomic instability associated with aging and cancer. But building a pedigree tree is very labour intensive because biologists have to manually manipulate the Yeast cells every 90 minutes for as long as 150 hours (i.e. more than 6 days). Therefore devising a system that automatically do this could speed up the experiments that would have taken a biologist a life time to do[40].

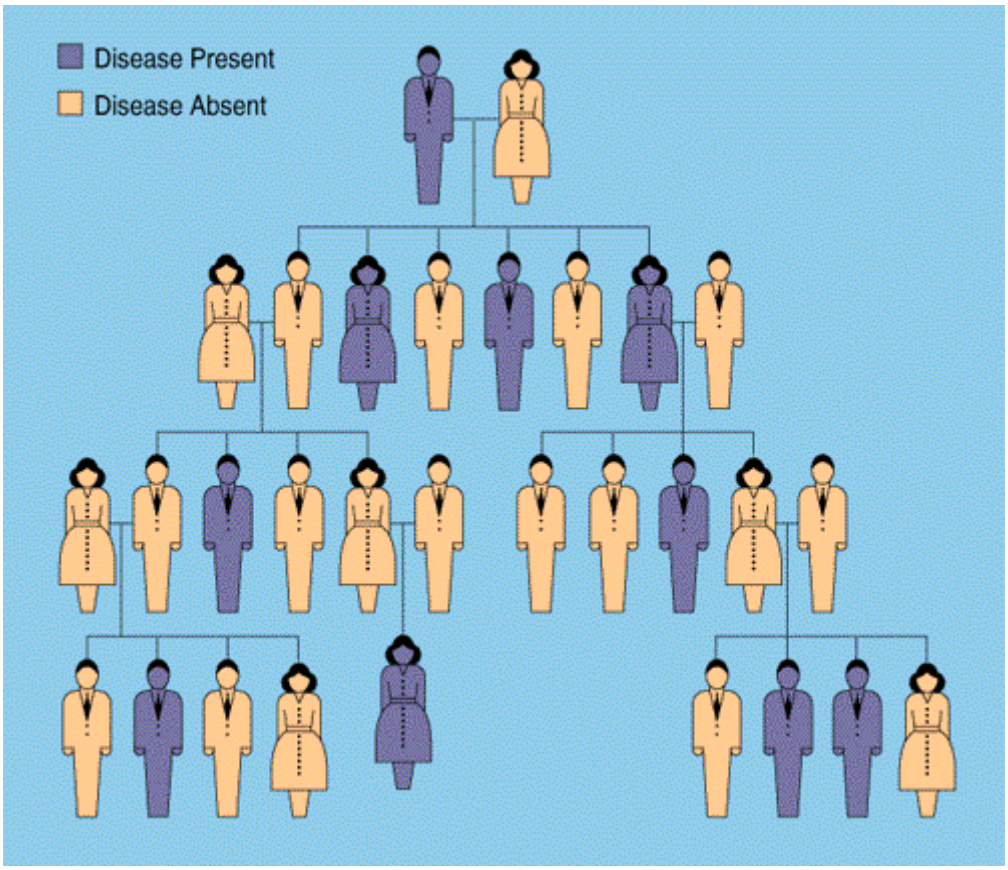

Figure 2. Example pedigree analysis on Human- Scientists look for a disease gene by studying over several generations DNA samples from members of the family who have developed the illness [44].

For instance, McMurray and Gottschling [41] during their Yeast pedigree analysis were only able to analyse 2 genes of the pedigree of 40 mother cells in 150 hours. And if, we extend this study by analysing the remaining loci (i.e. gene location) on every chromosome using multiple mutations and in different environmental conditions this could take several researchers' life time [40]. But with automation this could take only few years to complete. Hence, we realise that developing a system that could automatically track Yeast cells and build their ancestry would have huge benefits for the scientific community.

1.2 Aims and Objectives

The principal aim of my project is to investigate how state-of–the-art computer vision techniques could be applied to the tracking of Yeast cells. This will help to increase the throughput of biological experiments and to remove the complexity in tracking. Thus, we aim to research and build during our project a software tool that could be used to successfully track Yeast cells both in static images and in real-time conditions. The latter

could help in automating the constructing of the Yeast cells' pedigree trees and also in learning their behaviour.

The project is more of type II. Thus it includes lot of programming but, it also involves substantial amount of theoretical discussion and research work. During the implementation phase we will be using OpenCV and C/C++.

In addition, we plan to tackle the project by combining the Viola-Jones object recognition framework with Particle Filters. The latter would be used to track the detected cells and it could help in building and storing the cell trajectories. Also, we plan to devise a method for validating the models built.

1.3 Data

The data we will use during the project was provided to us by ***Dr Thilo Gross*** from the Department of Engineering Mathematics at the University of Bristol. The data is in the form of video recordings and we will use it to train the Viola-jones classifier and to artificially generate samples which will be used for testing – the figures 3 & 4 are samples taken from our data.

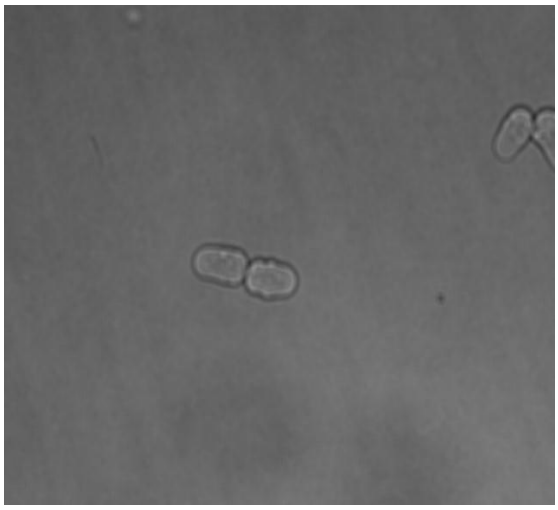

Figure 3. Yeast cells recording at the beginning of the experiment.

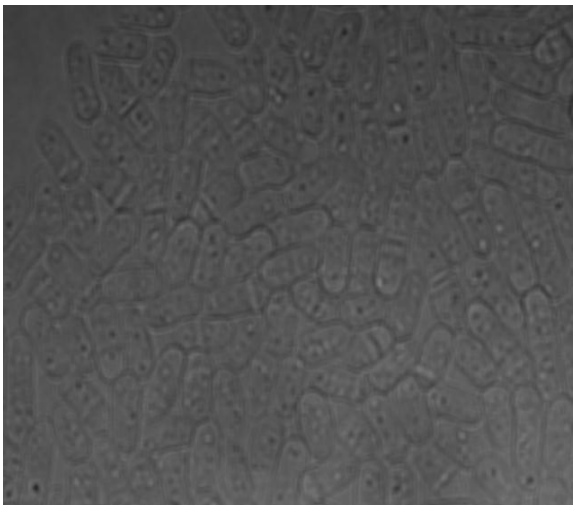

Figure 4. Yeast cells recording after few frames.

1.4 Thesis outline

In chapter 2, we present the theoretical background of the main techniques used during the project; these are Viola-Jones methods, Particle Filters and Multi-target tracking. Also, we present a literature review of related work in the fields of biological cell recognition, tracking and behaviour analysis in the context of Machine Learning and Computer Vision.

Chapter 3 discusses the implementation of the proposed methods as well as the techniques, such as Good features to track with Optical flow; we adopted in order to increase the accuracy of the system. Also, we present the parameters selected during the training of the Viola-jones and the methods for extracting positive and negative samples from our dataset. In addition, we present the intuition behind the novel Yeast cell model establishment and validation heuristic which is used to combine the detected cell ends together. Also, we explain how we separated the cell detection, the model establishment and the model tracking into different steps.

In chapter 4, we present and discuss the results obtained from the Viola-Jones training, the model establishment and validation heuristics as well as the Particle Filter tracking. Also, we compare the trajectories of the built paths by the Particle Filter to the ground truth; as well as a demonstration of how the model detection improves at each stage of the development process. Moreover, we computed the statistical significance of the Particle filter model tracking against the greedy model selection and the greedy model selection with validation heuristic. The chapter also points out the techniques we retained and the ones we discarded; because they were not suitable for our dataset.

In chapter 5, an evaluation of the project is presented. Here also, we suggest some possible further work.

2. Theoretical background

This chapter presents in its first two sections the theory behind the technique we used in our project. Also, in the last section it discusses the related work to the project in the form of a literature review.

2.1 Cell detections

2.1.1 Viola-Jones methods

VJ is an object detection framework that possesses high object detection rate –so this makes it very suitable for real-time applications. The VJ were introduced in 2001 [11] and the framework is composed of 3 stages: the initial stage is the feature detection; which is followed by the learning stage that used an algorithm based on Adaboost and finally the classification step; which is composed of a cascade of classifiers.

-Feature detection

The VJ uses Haar-like features to classify objects. These are rectangular digital image features; which get their name because of their similarity to Haar-wavelets [32]. Harr-like features are computationally cheaper to compute when compared to methods that extract individual pixel intensities. The reason behind this is the fact that the Haar-like features provide a method for encoding image properties in a form which can be computed more quickly.

The simple Harr-like features are composed of two adjacent rectangles; which are located at any scale and position within the image; and are referred to as ***"2-rect"*** features. And, the features are defined as the difference between the sums of image intensities within each rectangle [32].

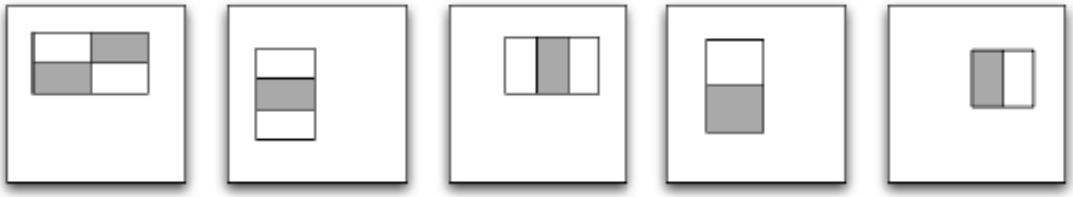

Figure 5. Examples of Haar-like features [32].

-Integral image

The Haar-features can be calculated quickly by using an image representation called integral of image. This concept was introduced by Crow 1984[31],[32] and it is mostly used in computer graphics.

$$ii(x,y) = \sum_{x'=<x,y'=<y} i(x',y') \qquad (1)$$

Where *ii(x,y)* is the integral image and *i(x,y)* is the original image intensity. However, the image integral can be easily calculated using the following recurrence:

$$s(x,y) = s(x,y-1) + i(x,y) \qquad (2)$$

$$ii(x,y) = ii(x-1,y) + s(x,y) \qquad (3)$$

Where *s(s,y)* is the cumulative row sum and we have the following base cases:

s(x,-1)=0 and ii(-1,y)=0

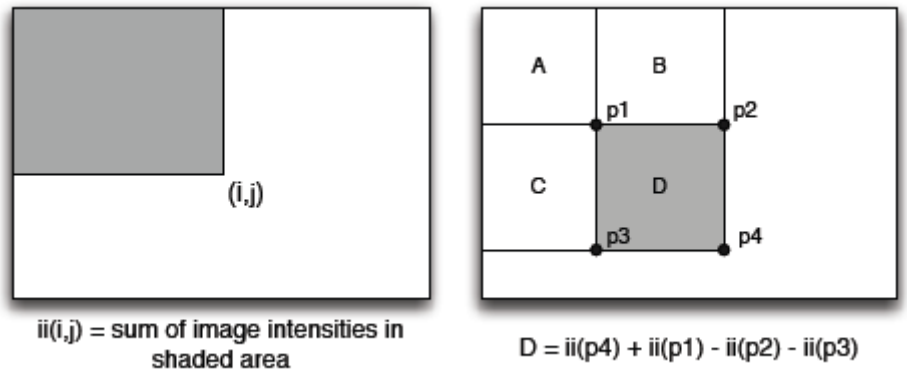

Figure 6. Integral image and rectangular feature calculation [32].

-Learning stage

We realise that the number of features generated in a sub-window can be extremely large. For instance, there are over 160,000 features in any given window (i.e. widow size of 24 x 24). To tackle this, the VJ uses a variation of the original Adaboost algorithm to select a small number of critical features which then can be used to form an effective classifier. However, for best results the training should contain images taken at different lighting conditions [31],[32].

The original Adaboost algorithm is used to boost the performance of weak learning algorithms. But with VJ, in order to reduce the number of features associated with any sub-window, the weak classifiers are designed to select the feature rectangles that best separate the positive from the negative examples. That is, for each of the feature the weak classifier learns the optimal classification threshold such that the minimal number of examples are misclassified [31],[32].

The Adaboost algorithm

Adaboost stands for Adaptive Boosting. It is a Machine Learning algorithm which can be used in conjunction with many other algorithms in order to improve their performance (i.e. it is a Meta-Algorithm). The algorithm works by tweaking subsequent classifiers on the instances/samples misclassified by previous classifiers. Adaboost is less susceptible to over-fitting than many other Machine Learning algorithms. However, it is sensitive to noise and outliers [36],[31],[32].

The algorithm assign to each sample a weight which is updated when it is either classified correctly or misclassified .When, the sample is misclassified its weight is increased and its weight is decreased if the instance is correctly classified- this allows the algorithm to focus on the misclassified samples (i.e. the one with the biggest weights)[36]. The Adaboost algorithm can be implemented as follow:

1. **Input:** N labelled samples $\{(x_1, y_1), \dots, (x_n, y_n)\}$
   Distribution D over N samples
   Weak learning algorithm WeakLearn
   Integer T of number of iterations
2. **Initialise the weight vector**: $w_i^1 = D(i) \quad for\ i = 1\ to\ N$
3. **Do for t= 1,…, T**

   - Set $p^t = \frac{W^t}{\sum_{i=1}^{N} W_i^t}$
   - Call WeakLearn providing it with the probability distribution p^t and getting back an hypothesis $h_t: X \rightarrow [0,1]$.
   - Calculate the error of $h_t: \varepsilon_t = \sum_{i=1}^{N} P_i^t \, |h(x_i) - y_i|$.
   - Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$.
   - Set the new weight vector to be : $W_i^{t+1} = W_i^t \beta_t^{1t^{|h(x_i)-y_i|}}$.

4. **Output the hypothesis:**

$$h_f(t) = \begin{cases} 1 & if \sum_{i=1}^{T} \log(1/\beta_t) h_t(x) \geq \frac{1}{2} \sum_{i=1}^{T} \log(1/\beta_t) \\ 0 & Otherwise \end{cases}$$

-classification/ the attentional cascade

The VJ also used a cascade of classifiers in order to achieve high detection rate while reducing computational times. This is achieved by using simpler classifiers to reject the majority of sub-windows before more complex classifiers are used to achieve low false positive rates. This process transforms the detection phase into a *generative decision tree* where a positive result from the first classifier triggers the second classifier and a positive from the second triggers the third and so on. However, this process is stopped (i.e. the sub-window is rejected) as soon as a negative classification appears at any point [11],[31].

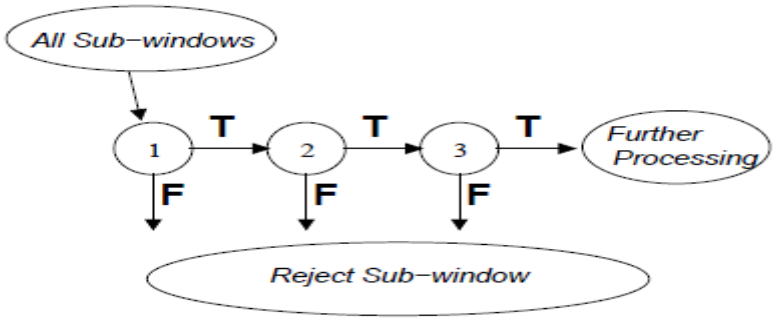

Figure 7. Object detection process (with cascade) [31].

Viola and Jones [11], using this method achieved results that were comparable to other systems such as Neural based or Example based approaches.

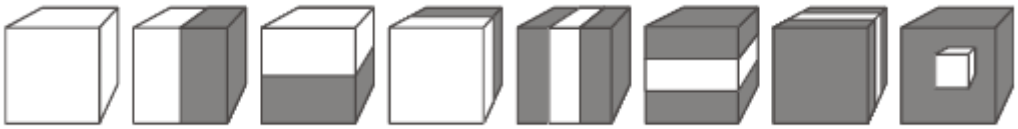

Figure 8. Haar-like (volumetric) filters used for mistosis feature extraction [12].

Li et al. [12] used a variant of the Viola-Jones method to detect biological cells. They extracted both positive and negative examples from the image frames and adopted a feature detection called ***Integral Volume;*** which is based on the work of Ke et al [27]- see Fig 8.

$$V(x,y,t) = \sum_{x' \le x, y' \le y, t' \le t} I(x',y',t') \quad (4)$$

Where V is the integral volume at pixel location *(x, y)* and at time t; *I (.)* represent the pixel at the original image at time t.

In addition, the features were calculated using the local variance of the image because using the pixel intensities of the original image would give a poor performance- this was pointed out by Ke et al [27]. The training classifier adopted was proposed by Wu et al. [38] and it has the advantage to be less computationally expensive that the original Adaboost; which requires substantial amount of time for training.

Furthermore, the cell detection was improved using a bilateral filter. The latter is a non-linear filter that smoothes out a signal but preserves its strong edges.

$$Var(x,y) = \frac{1}{|N|}\sum_{(x,y)\epsilon N}[I(x,y)-\mu]^2 \ , \qquad \mu = \frac{1}{|N|}\sum_{(x,y)\in N} I(x,y) \quad (5)$$

Where N is the local neighbourhood of size |N|.

Finally, a roll back filter was applied to extract non mitotic cells (i.e dividing cells). These cells appear as dark regions surrounded by halo pixels. And, the mitotic cells near interphase or prophase were detected by first applying a canny edge detector to extract their edges. Then the regions that are encircled by closed edges are filled. The regions with solidity greater than 0.9 and eccentricity less than 0.9 with mean intensity greater than $\mu+2\sigma$ and size within a certain range were selected.

In this work Li et al. [12] demonstrated how pure Machine Learning could be applied to cell detection

2.2 Cell tracking

2.2.1 Particles Filters

PF also know in the literature as Condensation, are a type of sequential Monte Carlo techniques that are used to predict the position of an object from it posterior distribution.

The PF use factored sampling to approximate arbitrary probability density functions. Factor sampling is a method for approximating probability densities. The problem here is to solve p(x|Z). That is to predict the state x of the object given the observation Z. In principle we can use the Bayes theorem to solve this. However, in practice this cannot be solved. Therefore, the factor sampling algorithm generate a random variate that approximate p(x|Z) [45].

$$p(x|z) = kp(x)p(X|x) \quad (6)$$

Where ***k*** is a normalisation term.

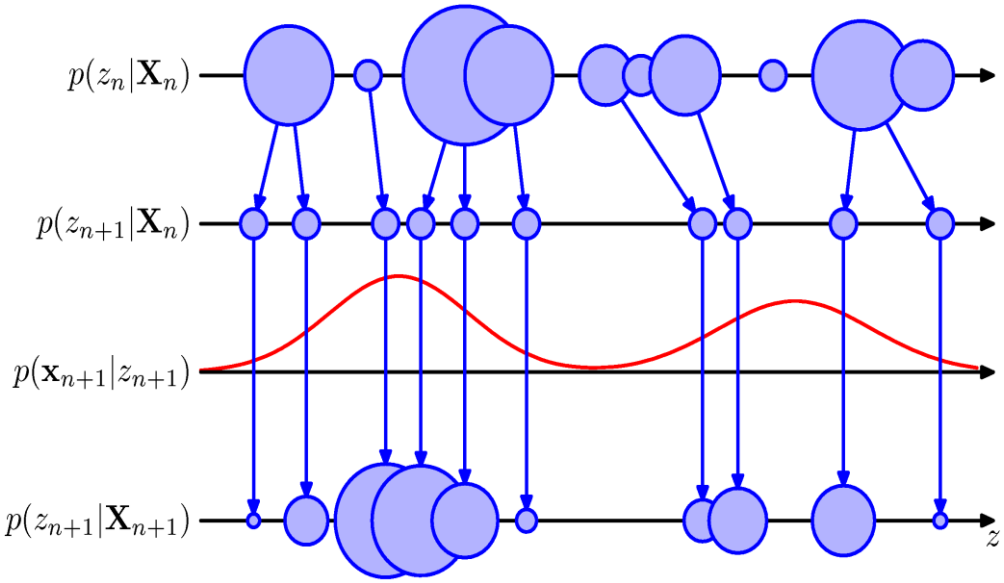

Figure 9. PF illustration [30]- state $p(z_{n+1}|X_n)$ demonstrates the resample process (i.e. the bigger the weight of the particle the more new particles are generated around it), at state $p(x_{n+1}|z_{n+1})$ a new likelihood is computed given the new observation and the bigger the likelihood the bigger the update weight of the particles at state $p(z_{n+1}|X_{n+1})$.

With the PF, the posterior probability distribution $\boldsymbol{p(x_k|X_k)}$ has to be evaluated at each step. However, the latter can be calculated, in the case of sequential Bayesian filtering by using the following two steps [34]:

-Prediction step

$$p(x_k|X_{k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|X_{k-1})\,dx_{k-1} \quad (7)$$

-Filtering step

$$p(x_k|X_k) \propto p(z_k|x_k)p(x_k|X_{k-1}) \quad (8)$$

The prediction step (Eq. 7) is followed by a marginalisation and a new distribution is obtained using the Bayes rule. In addition, the PF requires a model that describes the dynamic evolution/movement of the particles $\boldsymbol{p(x_k|x_{k-1})}$ as well as a model to compute the likelihood of each particle against the current observation $\boldsymbol{p(X_k|x_k)}$. Therefore we can see that the idea behind the particle filter is quite simple and the algorithm can be summarised as follow [37]:

- **Initialisation:**
  At state t=0, generate N particles with an initial state $S_0^n = \{x_0^n, w_0^n\}$ with n= 1, …, N and where x_0^n, w_0^n are the position and weight vectors respectively.

- **State transition**
  For n=1,…,N update the particles using the state transition model $p(x_k | x_{k-1})$ of your choice.

- **Weighting and normalisation**
  Compute the new weights of the particles and normalise them

$$w_t^{(n)} \propto p\left(y_t^{(n)} \middle| x_t^{(n)}\right) \quad (9)$$

  Where $\mathbf{p}\left(\mathbf{y_t^{(n)}} \middle| \mathbf{x_t^{(n)}}\right)$ is the likelihood based on the observation model.

- **Resampling**
  Resample and replace the particle according to $\boldsymbol{w^{(n)}/w_t^{(all)}}$. The number of particles should remain the same. This is called importance sampling but there are also some other resampling methods.

- **Estimate the position of the particles**
  This can be achieved by either assigning the prediction to the one with the biggest weight or by computing a weighted sum of the all the particles positions.
  Set t=t+1 and go to step 2.

Shen et al. used a PF [1] along with Active Contours to track cells- they were the first to use PF for cell tracking. Their study found that results obtained by PF tracking are comparable to manual tracking. However, they failed to test their approach against a benchmark. Their solution, involved the modelling of the state of the cell using a defined vector ***s***. The vector ***s*** was composed of the position of the contour, the orientation of the contour and the contour measurement. The variables of *s* were extracted on each image frame. The aim was to estimate the posterior density $\boldsymbol{P(s_k|z_{1:k})}$ as well as the associate expectation of the integral function $\boldsymbol{g_k(s_k)}$.

$$E\left(g_k(s_k)\right) = \int g_k(s_k) p(s_k | Z_{1:k}) ds_k \quad (10)$$

However, in the PF the above equation is approximated using the following equation:

$$\frac{E(g(s))1}{N}\sum_{n=1}^{N} g(s^{(n)})\frac{p(s^{(n)})}{\pi(s^{(n)})} \qquad (11)$$

With

$$\pi(s) = \begin{cases} 0, s \in \Gamma \\ -\varepsilon, s \in \Gamma_{in} \\ \varepsilon, s \in \Gamma_{out} \end{cases} \qquad (12)$$

Where ε is a positive constant and Γ_{in} is the area inside the curve and Γ_{out} represents the area outside the curve.

2.2.2 Multi-target tracking and data association problem

The data association problem is one the biggest challenges of multi-target tracking. This is due to the difficulty that arises when we want to relate each new observation to the correct object/Yeast cell. Usually, the data association problem deals with the state estimation of an unknown number of targets.

In the literature, several approached have been adopted for instance probabilistic approaches were used in [34]. However, in our project we adopted a heuristic approach that is based on the Nearest Neighbour standard filter. That is to assign each new observation to its closest state in order to compute the likelihood and update the state of the particles. We selected to use this approach because it is simple to implement and we believe that it is suitable for our data.

The technique has been used on its own for cell tracking. For instance, Zhou et al. [19] presented a NN-based cell tracking methods. They define a distance and a size based matching similarity measure which they then stored inside a distance matrix. The latter contained all nuclei positions at time t and time t+1. And the distance between adjacent x_i (t) and x_j(t+1) was denoted d_{ij}(t). Then, the authors matched the cell by scanning all adjacent matrices to find the successive candidates at frame t+1; these candidates were added up in ascending order of their distance size.

Furthermore, they used a 10% threshold; that is, when the sum of the distances exceed by 10% the size of the cell at time x_i (t), the algorithm is stopped and the last added distance is removed. The rational of the 10% threshold was based on experiments they did; they noted that the size of the nuclei would not exceed 10% of its

size at time t. Therefore, using this tracking method Wong et al. managed to detect cell migration, division and death points.

However, this approach cannot be used on its own because the error rate of the NN classifiers is proportional of the sample size. In addition, this technique in the context of Computer Vision is known to perform poorly in the case of multiple targets tracking with crossing paths [17]. This is reason we have used here in conjunction with PF so that the performance of our system will not be affected.

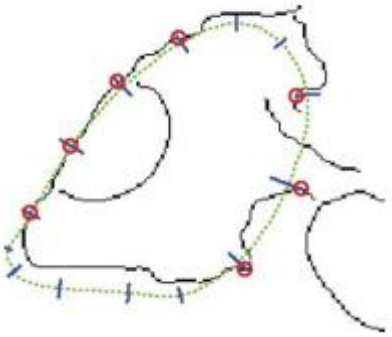

Figure 10. Observation model constructed with a canny edge detector [1].

2.3 Related work/Literature review

2.3.1 Cell detection

This section presents the different approaches that have been used for cell detection. It is divided into 3 main parts. The first part covers the contour-based methods for cell detection; the second part introduces the threshold based approaches. The last part brings together other types of techniques such as Support Vector Machines (SVM) and the Hough transform methods.

Contour-based models

Active Contour (AC) or snakes were introduced by Kass et al. they are used to represent deformable models where contours are deformed in order to minimize an energy function (Eq. 13) [1].

$$E^*_{snake} = \int_0^1 E_{sanke}\left(V(s)\right)ds = \int_0^1 \{[E_{int}\left(V(s)\right)] + \left[E_{image}\left(V(s)\right)\right] + [E_{con}\left(V(s)\right)]\}ds \quad (13)$$

Where E_{int} is the internal energy of the spline; E_{image} represents the image forces and E_{con} is the external forces.

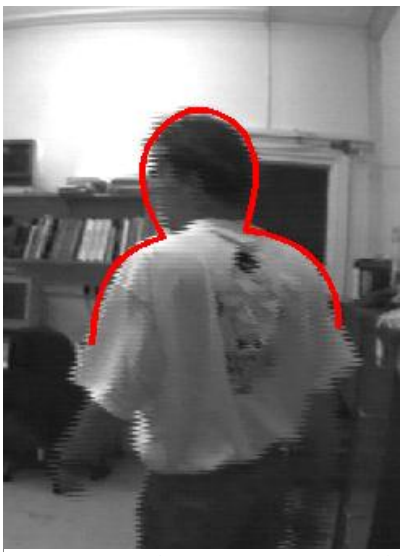

Figure 11. Body contours extracted using AC [42].

Although, AC are popular they have some drawbacks for instance they are computationally expensive; they can get stuck in local minima.

Standard AC approach

In 2005, Shen at al. [1] demonstrated that AC contours could be used for biological cell detection. They accomplished it by having 3 models of the biological cell. One that represents its shape, another that mimics its dynamics and a last model they called observation which is used to calculate the fitness/likelihood of the detected cell. The latter was built using a simple canny edge detector. And the probability of correctness of an edge was given by:

$$P \backsimeq 1 + \frac{1}{\sqrt{2\pi\sigma\lambda}} \sum_{m=1}^{M} e^{-\frac{f(dm,\mu)}{2\sigma^2}} \quad (14)$$

Where $f(dm, \mu) = \min(dm^2, \mu^2)$; dm is the distance between the contour and the edge and μ is the maximum distance between the contour and the edge point under consideration.

Also, in the following year they produced a software tool called CellTraker which is an open source implementation of their approach.

Thus, they built a system that could detect biological cells in real-time. The system could be applied to our project and it would certainly produce good results as the Yeast cells have a more rigid cytoplasm than the ***HeLa cells*** used by the authors. However, the authors have failed to test the system against any benchmark- we do have its accuracy rate and there is no evidence which suggests that the system have been used on other type of cells. Hence, we could use this as a starting point in our project but ultimately our aim is to build a system that is versatile enough to handle different type of cells. Furthermore, this approach would not deal well with the changes in the curve topology overtime. The latter is one of the disadvantages of AC which means that it would be very difficult to detect mitotic cells using this technique.

Level sets

Level sets are an improvement of the original snake contours. The main idea here is to represent the evolving contours (Γ) using a signed function where its actual contour corresponds to its zero levels.

$$\emptyset(s) = \begin{cases} 0, s \in \Gamma \\ -\varepsilon, s \in \Gamma_{in} \\ \varepsilon, s \in \Gamma_{out} \end{cases} \quad (15)$$

Where ε is a positive constant and Γ_{in} is the area inside the curve and Γ_{out} represents the area outside the curve.

Then according to the motion/ evolution equation (usually the Geodesic AC Eq. 16) of the AC one can then easily derive a similar flow for the implicit surface that when applied to the zero level will reflect the propagation of the AC [6].

Degerman et al. [5] investigated the performance of a level sets approach to biological cells detection. They introduced a pre-processing step that uses Laplacian of Gaussian (LoG) filtering. The latter, significantly increased the cells detection rate in low contrast images. These results were obtained by combining a Geodesic AC energy formulation with the Chan and Vese method [36].

$$E_{CV}=\mu.\text{Length}(C)+\lambda_1+\int_{insideC}|u(x,y)-c1|^2\text{dxdy}+\lambda_2+\int_{insideC}|u(x,y)-c2|^2\text{dxdy} \quad (16)$$

Where μ, λ_1 and λ_2 are just constants and c1 and c2 represent the average u inside and outside the zero level set iso-contour respectively.

The background pixels were ignored and the contrast was enhanced using eq5.

$$g\big(I(x,y)\big)=0.5\left(1+\tanh\left(\frac{G\sigma * I(x,y)-\gamma 1}{\frac{\gamma 2}{\pi}}\right)\right) \quad (17)$$

Where Gσ is a Gaussian and γ_1 and γ_2 are constants which are set so that the dark border of the cell are amplified, and so that the brightness differences inside the image are reduced. (Here γ_1 and γ_2 are all set to 100)

Furthermore, the team managed to speed up the level set function by only considering pixels that belong to the background. The approach managed to achieve an accuracy of 76% on a very difficult data set. This approach is relevant to our project because of the fact that Level sets are parameter free (i.e. they do not need to have a parameter representation of the object /cell); they are intrinsic and can easily follow shapes that changes in topology. The latter means that this technique would be able to handle mitotic (i.e. dividing) cells. However, they are computationally expensive because they involve solving partial differential equations that describe the surface evolution and they are still susceptible to local minima since they are based on local measurements such as the image gradient [6]. In addition we are not sure how this technique would perform with overlapping or touching cells which is one of the challenges we are facing with our dataset. Also, we think that 76% accuracy is quite low and it would not really justify the replacement of the human expertise by a real-time tracking system.

Threshold based segmentation

Thresholding works by separating images into dark and light regions. The technique exploits the regions' pixel intensities. For instance if g(x,y) is the thresholded version of f(x,y) at some global threshold T; this can be translated using the following equation:

$$g(x,y) = \begin{cases} 1 \; if \; f(x,y) \geq T \\ 0 \qquad otherwise \end{cases} \quad (18)$$

Chen et al. [7] investigated the use of threshold based segmentation in biological cell images. They used the Otsu segmentation algorithm which applied a global threshold to the image. Also, they managed to improve the technique by passing a Watershed algorithm which reduced the number of cells touching or overlapping. Moreover, they included another step to correct and merge the over-segmented fragments. The latter was achieved by looking at the compactness of the cell nuclei-the cells were merged based on that observation.

Furthermore, Rapoport et al. [8] adopted also a similar approach but they solved the over-segmentation problem of the images by using morphological filters. These filters helped to remove small particles and cell debris. Also, they measure the performance of the cell detection algorithm by computing the false acceptance rate (FAR) and false rejection rate (FRR). The former is the probability that the algorithm would detect a cell when in fact it is not a cell and the latter is the probability that the system would fails to detect a genuine cell; these techniques are usually performance metrics used in biometrics [24].

Cohen et al. [9] also used this approach; they exploited the intensity of the pixels within the image. They labelled all the pixels with a brightness of 5 s.d. above the average pixel value as belonging to the interior region of the cell (i.e. halo pixels). Then, the halo pixels were separated into connected regions using the watershed transform. And, K-means clustering was subsequently used to split the pixels into 3 distinct regions with the brightest and the darkest belonging to the cell while the intermediate to the halo region. The halo region was discarded and another watershed transform was applied to separate touching cells.

We realise that this methodology could be of great significance for our project because the technique is easy to implement [25] and, the approach deals with inconsistencies that occur during the process. The only caution is the fact that debris and cell fragments are discarded. We do not know whether such discarded information would not have any impact in the cell behaviour analysis. Another issue to point out is the fact that thresholding does not consider relationship between pixels and it does not

guarantee that the identified pixels are continuous. In addition, this technique would not perform well near the region boundaries and the threshold is heuristically set. However, we think that the evaluation method pointed out by Rapoport et al. [8] could be applied to our project.

Other approaches

Hough Transform (HT)

The HT is a feature extraction technique which is used to identify objects with regular shape such as circles or rectangles. The technique since its introduction in the 1970s has being applied to many real-world object detection problems. The technique works by letting each edge "vote" for a plausible line location. That is each edge votes for all the lines passing through it and the line that corresponds to a high bin value is selected [6].

Mouroutis et al. [10] devised a method for robust cell nuclei detection using a variant of the HT called Compact Hough Transform (CHT). They used the CHT to determine the possible locations of the nuclei then they used a likelihood function to optimize it by choosing the location that maximises it.

Mouroutis and his team [10] based their study on the fact that there is no extreme variation in the shape of the nuclei. Thus, they defined *Dmax* and *Dmin* the bounds at which each pixel could lie. And to overcome the irregularity of the shape of the nuclei they assumed that the nuclei have some large convex structure- the edged of the cell were detected using a Sobel mask.

$$CHT(x) = \int_{r=dmin}^{r=dmax} HT(x,r).dr \quad (19)$$

Where r is the radial parameter in the circular HT.

The team managed with some optimization to successfully approximate as much of 95% of their image database. Also they algorithm perform well in dividing nuclei in to likely substructure.

Although this approach gave encouraging results we do not think that the HT is appropriate for our project. That is not to say the HT will not do the job as Ulrich et al. [23] have used a variant of the Generalized HT for real-time object recognition. The HT can perform relatively well in noisy conditions as well as when there are some additional structures in the image. Another advantage of the HT is the fact that is independent of the segmentation result [6]. But it is also known to be computationally expressive and it does require lot of store space because the algorithm uses an accumulator to detect the existence of a line. In addition, the complexity of the algorithm depends both on the image space and the number of parameters within it [6] [23].Therefore we do not think that this approach is the best for real-time tracking in this case.

Support Vector Machines (SVM) approach

SVM are patterns classification algorithms that work by separating the hyperplane at a maximum closest to the training data (see figure 12). This is achieved by maximizing the margin that separates the planes in the features space between different classes (i.e. cell and non-cell regions).

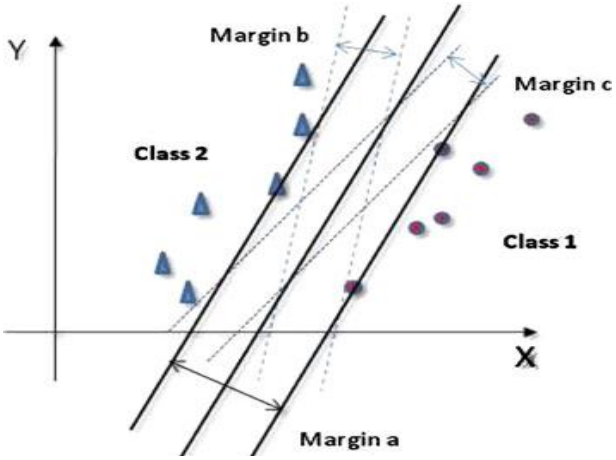

Figure 12. SVM hyperplane separating example [13].

$$f(x) = sign\left[\sum_{i=1}^{N} \alpha iyiK(x, xi) + b\right] \quad (20)$$

f(x) is the SVM decision function, where x is the data point to be classified, *xi* is the support vector, *N* is the number of support vectors, *b* is a constant and *yi* is the class label.

Han et al. [13] investigated the application of SVM to cell nuclei detection. During training of the SVM, they used a grid based search to perform a cross-validation over ball possible parameter configurations that are within a certain range. After that, they noted that the SVM approach on average correctly classified 90% of the cells- the edges of the cell were detected using a Laplace detector. They also pointed out that the success of the technique is highly reliant to the feature vector-so they combine the gradient edge and the row pixel value to build a strong feature vector.

Furthermore, in another publication by Triggs and Dalal [14], SVM were used along Histogram of Oriented Gradients (HoG) to detect Humans. Although, this is not a biological cell detection task, HoG could be used in any object detection problem because of the fact that they are themselves just feature descriptors. Triggs and Datal's framework achieved almost a perfect classification of all the 1800 human images of the MIT dataset.

We remark that this approach might be applied to our project. However, one of the main disadvantages of Han's team [3] approach for instance, is the fact that there is no formal method to use for the selection of the parameters of the search. So we would have to determine them through trial-and-error. Furthermore, SVM are somewhat slow to compute and they require a large amount of both training and testing data. Also the choice of the kernel is purely heuristic [6]; thus we will not be using this method in our project.

Voronoi based segmentation

This technique is based on the mathematical principle of Voronoi diagrams. A Voronoi diagram is a specific decomposition of a given space, for instance an image that is determined by a specific group of objects within the subspace; in Computer Vision this means that each pixel is assigned to nearest centre in the set [6].

Jones et al. [28] presented a method for determining cell boundaries that uses Voronoi based segmentation. Their approach works by finding the cell boundaries between adjacent regions in which "seed" areas have been identified within the regions to be segmented. Jones et al. achieved this by defining a matrix within the image plane and by calculating the distances between the seed regions according to a metric. They used a Riemannian metric defined in term of the image I and the regularisation parameter λ:

$$G = \frac{\Delta g(I)\Delta g^T(I) + \lambda I}{\lambda + I} \quad (21)$$

Where I is a 2x2 identity matrix and g is a radius blur.

Then each image pixel is assigned to a cell according to its distance from the cells' nuclei. The authors found that this approach is very similar to the one of the Geodesic AC with only a reverse behaviour of the algorithm around the edges. Therefore, because of this similarity and the limitation of the AC discussed earlier in this chapter, we do not think that this approach would be used in our project. In addition, although Voronoi

segmentation has the advantage of detecting thin edges in contrast it totally ignores local image structure [6].

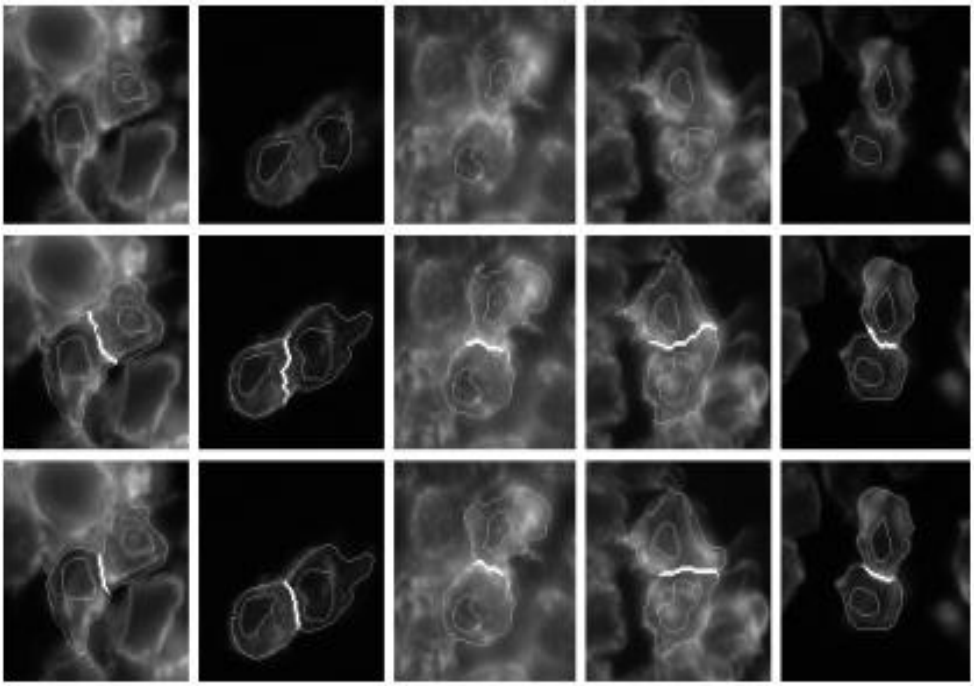

Figure 13. The top row displays the cell nuclei; the bottom is the obtained automatic segmentation and the bottom row represent a manual segmentation the of the cells [28].

Cell Detection Summary

In this section we have reviewed some of the Computer Vision algorithms used for biological cell detection. Also, we have found that 3 approaches are relevant to our project. The Level Sets approach is suitable to our project because of its superiority over other contour based method such as the standard Active Contours; Levels sets are parameter free, intrinsic and can easily follow changes in the cell topology. The latter could be useful to us in particular when we are looking for mitotic cells for instance. In addition, the Viola-Jones framework in particularly important to us because of the fact that it could be trained to automatically identify different cell events (i.e. mitosis). And finally, another technique that caught our attention is the threshold based segmentation because it is computationally less expensive and easy to implement that the other techniques and also it produced good results. In addition, Rapoport et al. [8] used a cell detection evaluation approach which we think could be applied to our project; they measure the performance of the cell detection algorithm by computing the *false acceptance rate* (FAR) and *false rejection rate* (FRR).

2.3.2 Cell Tracking

This section presents the different approaches that have been used for cell tracking. It is divided into 5 main parts. The first section covers the image registration approach to cell tracking; the second part introduces the Contour based approaches; section 3 investigates the Statistical/Probabilistic methods and the following two sections present successively the Graph based methods and the Image overlap cell tracking technique.

Image registration

This technique has been applied to wide range of problems including motion correction in medical imaging or motion detection in CV. The basic idea of this principle is to find a mapping function that will translate all the objects present in a source image to a target image. Thus, this would help to determine how far a pixel has move from one frame to another [6].

Hand et al. [15] investigated the use of an image registration model which uses the principle of optical flow. Because, such an approach would not be subject to the landmark constraints of the common technique (i.e. no landmark need to identify in the images). The team achieved this by registering each pixel based on the flow of the pixel intensities of the two images. In addition, this approach seems particularly attractive for biological cells because the motions of the pixels were limited to non-linear elastic motion. That is, even if the morphology of the cells within the image changes we would be able to register them.

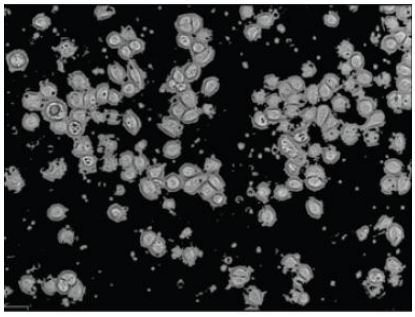

Figure 14. Segmented image with removed background [16].

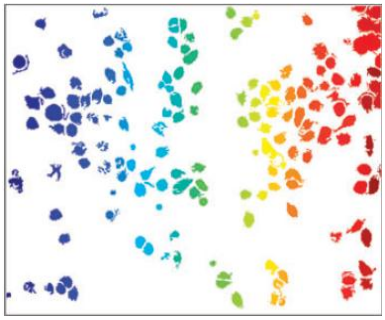

Figure 15. Final segmentation with each cell represented by different colour[15].

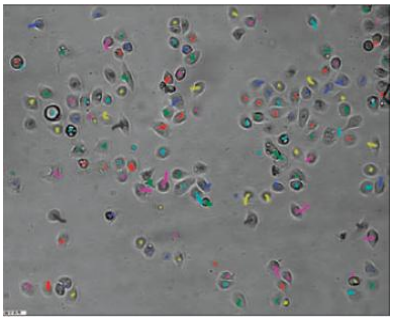

Figure 16. First tracking detected [16].

The relationship between the difference in pixel intensities and the mapping Δx and Δy of the two images was obtained using eq. 6.

$$f - m = \frac{\Delta x}{2}\left(\frac{\partial f}{\partial x} + \frac{\partial m}{\partial x}\right) + \frac{\Delta y}{2}\left(\frac{\partial f}{\partial x} + \frac{\partial m}{\partial x}\right) - \frac{\Delta \alpha}{2}(f + m) - \Delta\beta \quad (22)$$

Where f and m are matrices that contain the pixel intensities in the target and source images respectively; $\Delta\alpha$ and $\Delta\beta$ are included to compensate the overall intensity difference of the 2 images.

Hand et al. [15] demonstrated that image segmentation could be successfully applied to track biological cells. However, we remarked that although this approach might be relevant to our project, it has its limits. The technique is highly reliant on the quality of the segmentation of the image; so we think that if applied to our project, the resulting system would not perform well when we have touching or overlapping cells. This is even pointed out by the team but they dealt with the issue by artificially generating images with low cell density. Hence, for this reason, this system does not seem to be a safe option for us.

Contour-based methods

The concept of active contours has been introduced earlier in this chapter. This section talks about their application in cell tracking.

Degerman et al. [5] solved a tracking problem using a contour based approach. They solved the problem by associating segmented regions in subsequent image frames by solving the asymmetric problem using the next equation.

$$max \sum_{i=0}^{n} \sum_{j=0}^{m} a_{ij} X_{ij} \quad \forall (i,j) \in \Gamma \qquad (23)$$

Where a_{ij} represents a weight that indicates the befit of matching region I to region j in the next time sequence; Γ is the set of pairs (i, j) which can be matched and X_{ij} is determined by:

$$X_{ij} = \begin{cases} 1 \; if \; i \; is \; assign \; to \; region \; j \\ 0 \quad otherwise \end{cases} \qquad (24)$$

Degerman and his [5] team solved the asymmetric problem using the auction algorithm. That is, they matched the object region j at time *t+1* and the track region I at time t. Also, when i=j=0 they called the two regions dummy object and dummy track respectively. And except for the dummy regions all the assignments are 1 to 1 assignments and each of these assignments was weighted using:

$$a_{ij} = \left(c_d . d_{ij} + f_{ij}\right) . w_i \quad (25)$$

Where w_i is the weight for track I, d_{ij} and f_{ij} are calculated by Eq. 26 and Eq. 27 respectively; c_d is the relative weight between 2 terms.

$$d_{ij} = 1 - tanh\left(\frac{1}{d_0}\sqrt{\left(x_i(t) - x_j(t+1)\right)^2 + \left(y_i(t) - y_j(t+1)\right)^2}\right) \quad (26)$$

d_{ij} calculates the distance between centroid *j* at time *t+1* and the centroid belonging to I; f_{ij} measure the size difference between object j and its previous size *i*.

$$f_{ij} = exp\left(-\left(\frac{|A_i(t) - A_j(t+1)|}{f_0.A_i(t)}\right).\left(\frac{|A_i(t) - A_j(t+1)|}{f_0.A_i(t)}\right)\right) \quad (27)$$

Where f_0 is an empirical value set depending on the segmentation method.

However, they found that this approach if used on its own provided poor results- the accuracy was in the range of 66-76%. The accuracy here also was affected by the segmentation approach.

Shen et al. [1] combined PF with Gradient Vector Flow (GVF) AC for cell tracking. This approach seemed to progress more inside the boundary concavity and it also captured a longer range to guide the AC towards the boundaries. This approach although not tested on any data helped to retrieve features that are not covered by the PF- such as sharp corner.

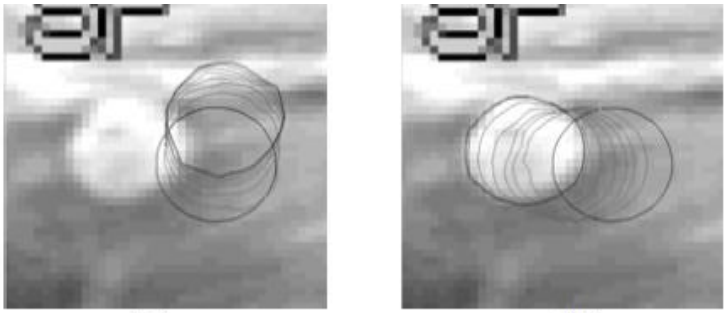

Figure 17. Standard GVF drift away from the cell (a) and the improved (version with Dirichlet) capture the leukocyte [15].

Ray and Acton [15] also investigated the use of AC for cell tracking. They developed a tracking algorithm that combined a Partial Differential Equation (PDF) and GVF, together with a shape size and an implicit resampling constraint. They approach suggested that the use of GCF constructed with a Dirichlet type boundary condition, provide an AC tracker that is robust for a wide range of conditions (see figure 11).

Li et al. [16] use AC for a multi-target tracking system that can simultaneously follow hundreds of thousands of cells in the most challenging environments. In 2006, the team adopted a Level-Sets (eq. 28) tracking approach which consisted of keeping track of certain regions of the cells such as the background.

$$\emptyset^k(x) = \begin{cases} < 0, & if\ x\ \in \Omega_C\ \ (cell\ region) \\ = 0, & if\ x\ \in \Gamma^{k-1}\ (countour) \\ > 0, & if\ x\ \in \Omega_0\ (background) \end{cases} \quad (28)$$

They also used an auxiliary region labelling function to monitor the cell regions in order to match the motion pattern of the cells; the objective of this approach was to minimize the following energy function which is a combination of 3 terms.

$$E = E_{region} + uE_{edge} + vE_{motion} \quad (29)$$

Where E_{edge} draw the contour to the high gradient magnitude, E_{motion} attracts the contour to the predicted mode of the cell's centroid while E_{region} matches the image appearances.

$$E_{region} = \sum_{n\epsilon\{0\}UN^{k-1}} \int_{\Omega_n} -\log(I(x)|\Omega_n\, dx, \quad (30)$$

$$E_{region} = \sum_{n\epsilon\{0\}UN^{k-1}} \int_{\Omega_n} g(x)\delta(\phi^k(x)) \mid \Delta\phi^k(x)dx, \quad (31)$$

$$E_{region} = \sum_{n\epsilon\{0\}UN^{k-1}} \int_{\Omega_n} \left(v - logp(I(x), x|S_n^{k|k-1})\right)\, dx \quad (32)$$

Where *u, v, g* are real coefficient, *δ(.)* is the Dirac delta function and *g(.)* is an edge indicator. The probability distribution in Eq. 31 represents the pixel intensity distribution in each region and the one in Eq. 32 matches the pixel intensity to the predicted motion filter. Moreover, they devised a method to prevent cell merge during tracking while still allowing cell splitting.

In 2008, Li et al. [16] improved their work by adding an interacting multiple models which will be discuss more in details later in this chapter. This second approach achieved an accuracy rate of around 86.9-92.5%. And, Li and his team pointed out the importance of the use of topological constraints for cell tracking.

In this section, we have introduced AC for the purpose of biological cell tracking. We noted that they have produced some very interesting results but all the approaches we have seeing are computationally expensive and they have the potential to be stuck in local minima; although Level Sets tend to perform better that the standard AC. Nevertheless, for our project we do not think that they will be used for cell tracking on their own as they are bound to errors and simpler techniques that we will discuss later in this chapter have been used to achieve better results.

Statistical/Probabilistic methods

Statistical object tracking techniques use a probability inference framework to build the trajectory of the object/cell from the given image sequence. This approach usually involves two steps: estimation of the state of the object and update of the object representation [22].

Statistical Thresholding method

Kachouie et al [18] introduced a statistical thresholding method for cell tracking. First they extracted the cell centres of the candidate cells from the image sequence. These were obtained using a probability map which was then thresholded using an empirical variable T. These cell centre candidates were associated together using a joint probability distribution. After that, the team computed a validation gate in order to determine the distance of each cell centre at time t-1 based on its Euclidean distance from time t. During the process the cells were assumed to be doing a random walk modelled by:

$$P_v \sim N(0,\sigma) = \frac{1}{\sqrt{2\pi}}\ exp\left(-\frac{D_e^2}{\sigma^2}\right) \qquad (33)$$

Here, D_e is a validation variable based on the distance between the identified cell centre at time t-1 to the cell centre candidate at time 1.

However, this method seems to be very limited just as pointed out by the authors themselves. The method may suffer from the fact that association of the cell centre could be ambiguous. In other words, the method could fail if we have some close-by, dividing or overlapping cells. Also, the method has not been tested on any data so this method as it is presented in this paper we think is not appropriate for our project.

Kalman Filters (KF)

KF are both very popular in the computer vision community. KF estimate the state of a linear system where the state is assumed to possess a linear Gaussian distribution [22]. Just like the PF, the KF is a Bayesian estimation technique that is used to track stochastic systems that are observed using noisy sensors [45]. The KF is composed of 3 probabilistic models:

- ***The system model*** describes the evolution in time of the current state vector u_t; the transition between states is modelled as the known transition matrix T plus some Gaussian noise with covariance Q_t.

$$u_t = T_{t-1}u_{t-1} + \eta_t, \quad \eta_t \sim N(0, Q_t) \text{ (34)}$$

- ***The measurement model*** which relates the measurement vector d_t to the present state using a measurement matrix H_t to which some Gaussian noise is also added .

$$q_t = H_t u_t + \varepsilon_t, \quad \varepsilon_t \sim N(0, R_t) \text{ (35)}$$

- The prior model which describes the prior knowledge about the state of the system at $\overline{u_0}$ and its covariance P_0 before any measurement is taken.

$$E[u_0] = \overline{u_0}\,, \qquad cov[u_0] = T_0 \text{ (36)}$$

Tang et al. [20] investigated the use of Kalman Filters in active cell tracking. They argued that they can be used to successfully track biological cells. However, they tested their method on only 3 active cells in 3 image frames- which we think is very limited if we were to be able to do a conclusive assertion about its performance. In addition, Chen et al. [1] pointed out that KF could collapse during the tracking process because of the image inferences such as occlusions and background clutters does not have a (linear) Gaussian distribution. These can influence the measurement of cell edges so subsequently reduce the accuracy of the technique; and this is one of the limitations of the KF they give poor results when the state of the system does not follow a Gaussian distribution or/and if the systems itself is nonlinear. Therefore, because of these reasons we will not be using KF in our project.

However, we think that PF will be very useful to our project. Although they suffer from some problems such as they are computationally expensive- the latter is proportional to the number of particles; it is difficult to determine how many particles are actually needed and selecting the best resampling methods is always a challenge but we think that their advantages outweigh their limitations.

Mean-Shift

Mean shift are simple iterative algorithms that are used to determine the mode of a dataset using the principle of kernel.

Debeir et al. [21] investigated adaptive mean-shift kernels for cell tracking; in this particular work, the authors used a unique set of kernels with adaptive sizes but they introduced a variation in their weights to determine their combination. Therefore, a given state of weight $W= (W_1, W_2,...,W_n)^T$ is associated a specific grey level pattern. And a variation in their grey level would indicate an object being detected. Hence, the team proposed an adaptive combination that enabled them to pass from one combination to the other.

$$W^*=\alpha W + (1-\alpha)\, W' \quad (37)$$

α: function of object current size

Finally, they merged the cell trajectories in order to deal with cell duplications that occur during mitosis. This type of tracking even though it could be used to track biological cells would not be recommended for our project because the second approach suffers from the constancy of the kernel shape which is a problem from all kernel filters. The problem above has being dealt with in CV using contour-based tracking method; these were covered earlier in this chapter. Furthermore, Mean-Shift is not very robust; it can easily lose track of its target when there is a large variation in spatial information of consecutive frames [22]. In addition, the authors have failed to provide any independent testing of their data. The publication does not report the accuracy rate of these two approaches.

Overlap based tracking

Rapoport et al. [8] investigated how the overlap of cells in subsequent frames could be used to track them. They derived from the segmentation process a binary representation of the image-they called this image the “cell mask”. Then to map the cells they calculated the intersection between adjacent cell masks. Furthermore, they determined the backward and forward overlaps which are the ratios between the intersection and the area of the cell mask in subsequent image frames.

$$forward\ overlap = \frac{(A_t \cap A_{t+1})}{A_{t+1}} \quad (38)$$

And, $$backward\ overlap = \frac{(A_t \cap A_{t+1})}{A_t} \quad (39)$$

Where A is the cell area.

In addition, the cell pair with the largest overlap was selected to be the unique successor of the cell-provided that both overlaps exceed a certain threshold. In addition, the cell path fragments were built on the basis that a cell must only have a one and one only successor or predecessor. That is, every cell only belongs to one path and overlapping paths were stored inside a matrix for later validation.

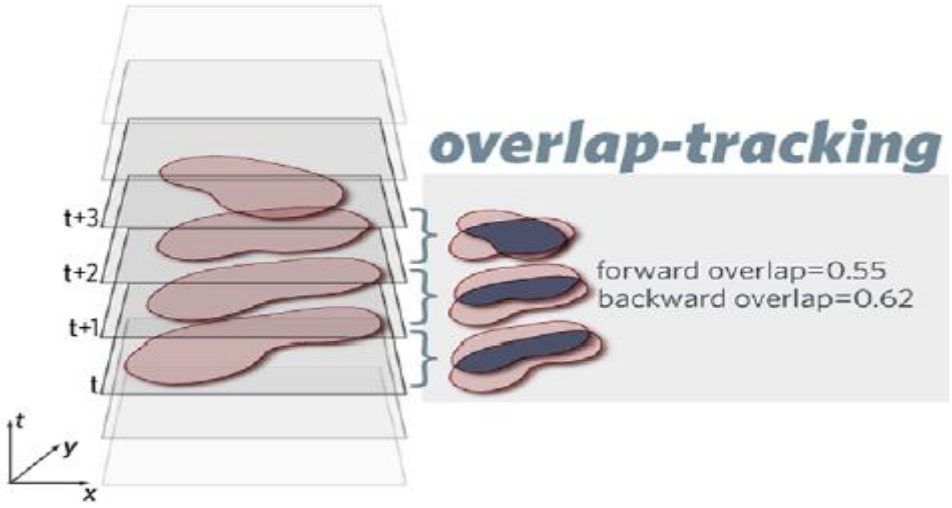

Figure 18. Movement of detected cell at different time frames [8].

Looking at the way this technique works we can argue that the system would not perform well with overlapping or even touching cells. But the authors managed to improve that limitation by adding a validation step to the system. Hence, we think that a validation step would be a great importance in our project. But the choice of the validation process would be one of the challenges in our project. Another thing to point out in this project if the fact that the authors went to great length in order to correct the results of the reference data sets; they employed a team of scientist to manually remove errors from the datasets A and B that contain 244.850 and 80.500 single cells respectively. This type of error correction would be infeasible in our project because of its expensiveness in term of human resources. Therefore, the one think that we would be using in our project is the idea of having a validation algorithm to correct the errors that could occur when we build the cell trajectories. Hence, the technique they used to

evaluate their algorithm could be applied to our project. The team computed what they called the "*Complete Path Detection rate*"; that is the number of correct complete paths divided by to total number of complete paths. The ratio calculates the tracking trustworthiness which we think that could be used to evaluate the tracking algorithm in our project.

Cell Tracking Summary

In this section we have introduced several cell tracking techniques. However, we think that only two methods could be applied to our project. The first technique is the Particle Filter method mainly because it performs better than the Kalman Filter approach which in contrast to the Particle Filter assumes that the system is linear and that the system could be modelled by a Gaussian. The second approach is not a full method for cell tracking but the work of Rapoport et al. [8] has raised the importance of the validation of the cell tracking results (i.e. the paths/trajectories built) which we think will be of great importance to our project. For instance, we think that their cell trajectory/ path evaluation method could be used to check the results of our project.

2.4 Summary

In this chapter we have introduced the introduced the theoretical background (section 1&2) of the main techniques used in the project. In addition, we have made a review of the different approaches used in the literature. And we found that The VJ and PF methods are the most suitable for our project. That is because the former has a high detection rate which make it very suitable for real-time recognition and also it is very flexible as it can be training to recognise any rigid object (i.e. cell ends in our case). And we think that PF are the most suited for our project. Because, they assume that the model of the system is nonlinear and that it cannot be modelled as a Gaussian; we think that this is significant to our project mainly because of the clutter or noise that can occur in the image in particular in fluorescent/noisy condition conditions.

Furthermore, we decided to combine VJ and PF because we wanted to have a clear separation between the cell end detection, the cell model establishment and the tracking process. This independence between these 3 steps would allow us to build a more flexible and robust system as it will help us to implement validation methods at each step. This means that the performance of the system would improve progressively because at each step as the unwanted data will be discarded leaving only the good one.

3. Methods

This chapter introduces the methods/techniques used to detect the Yeast cells' ends. Also, it describes our approach for model establishment and validation. Moreover, it explains the techniques we have investigated in order to improve both the detection and the tracking of the cells.

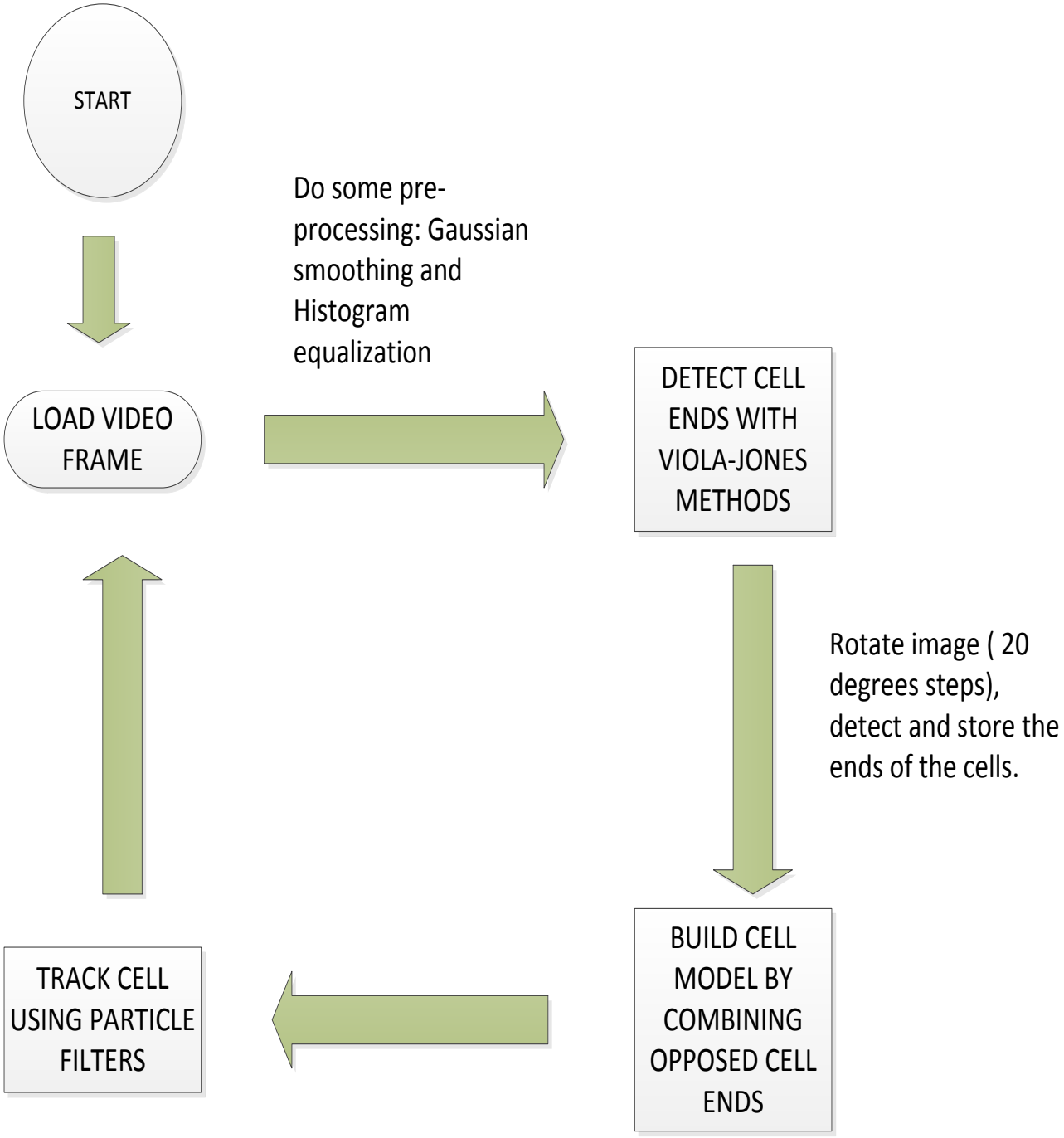

Figure 19. System overview.

3.1 Pre-processing:

Prior to the VJ training and the detection process, all the images and the training samples were denoised using a 7x7 Gaussian filter.

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

Figure 20. 7x7 Gaussian Kernel.

Also, they were all histogram equalized in order to achieve a more uniform distribution of the pixel intensities around the image-this is illustrated in the following images Fig. 21-23.

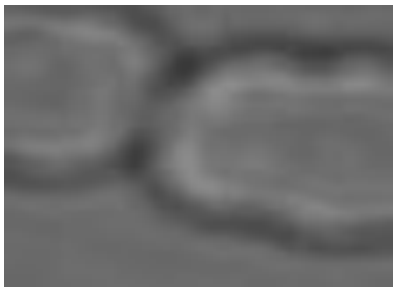

Figure 21. Original sample image.

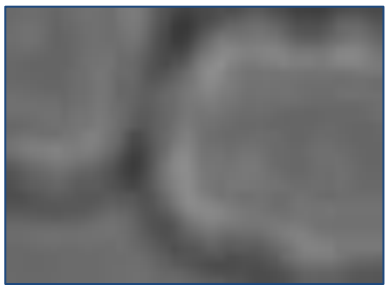

Figure 22. Gaussian Denoised image.

Figure 23. Histogram Equalised image.

We noticed that when the histogram equalisation is applied to the image the Yeast cell's edges are sharper-i.e. we can easily identify the cell's boundaries. In addition, we remarked that the pixel intensities are more uniformly spread this is very useful during the VJ training because VJ is sensitive to differences in pixels intensities.

3.2 VJ Training

During the training phase, 186 positive samples (i.e cell ends) were collected using a combination of template matching and manual collection. The collected samples were all facing relatively the sample direction (see Fig 21). In addition, we constructed the negative samples that are required for the VJ training from our video frames. These

negative samples were constructed by removing the Yeast Cells' ends from each of the image frame using a Gaussian Blur- the latter was done in order to prevent the addition of any extra structure to the image. Because, adding any new structure could affect the performance of the VJ training/detection and using this method we constructed 500 negative samples were generated. The training was done using the build-in *haartraining* function of the OpenCV Library. However, at some point in the development we had to edit the source code of the OpenCV library so that we could merge the files returned by the ***opencv_createsamples*** function in a single file as the current version of the library is not able to do that.

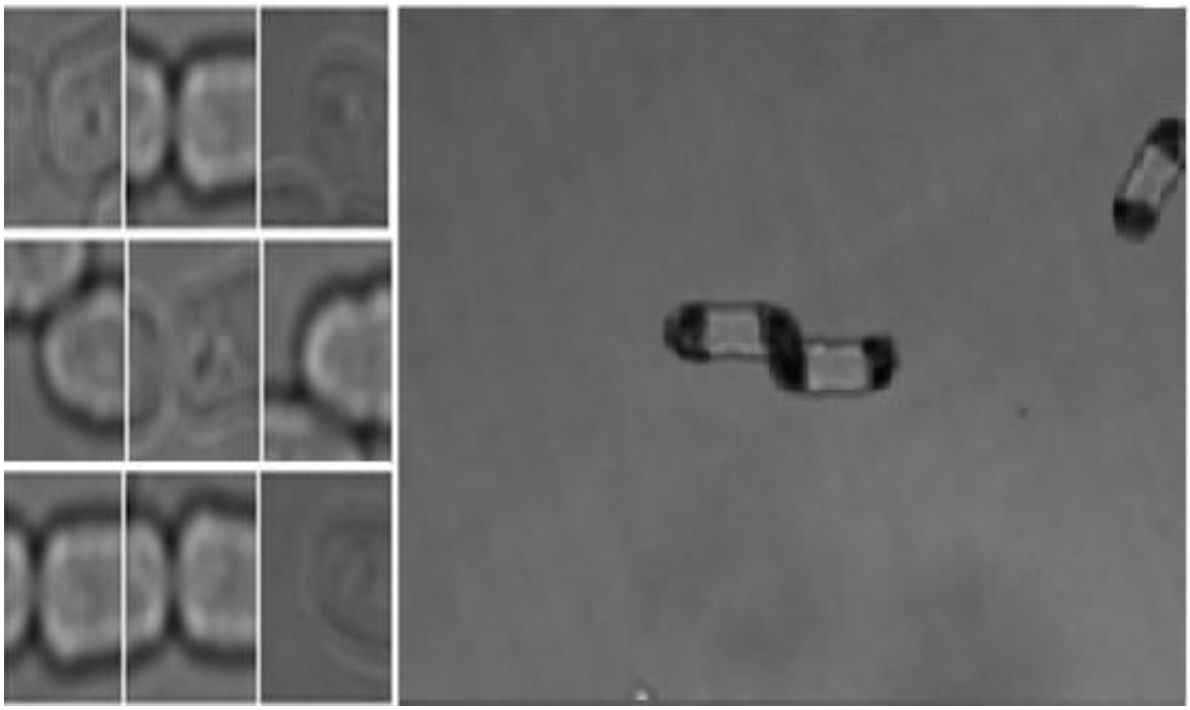

Figure 24. Positive samples (left) and Negative sample (right).

Furthermore, because of the computational complexity of the VJ training, the latter was conducted on the University of Bristol's supercomputer which is equipped with an OpenCV module. After, running several experiments, the following training parameter were selected. And we will discuss the tried settings in a later chapter.

Positive Samples	**Negative Samples**	**Minimum Hit rate**	**Maximum False alarm rate**	**Number of Stages**	**Sample Width**	**Sample Height**	**Weight Trimming**
186	500	0.99	0.4	20	20	20	0.95

Table 1. VJ training settings.

Minimum hit rate: is the minimum desired hit rate for each stage classifier.

Maximum false alarm rate: maximum desired false alarm for each stage of the classifier.

Weight trimming: specifies whether and how much weight trimming should be applied.

Model establishment

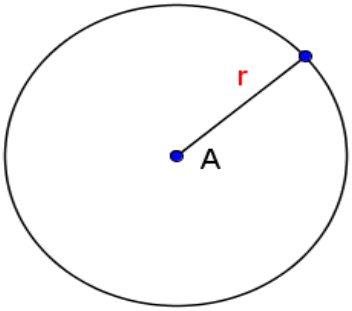

Figure 25. Every detected cell end is modelled as a circle.

This task has been one of the most complex task of the detection process. This is due to the fact that although the Yeast cells' ends can be detected directly by the VJ classifier, the task of building a model of the cell that can be used by the PF is not trivial. That is because the VJ classifier was only trained to detect one end of the cell (Fig. 25) and each of the detected cell's end is modelled as a circle. So during the detection process, to detect the corresponding opposite end, we rotated each video frame 8 times in 20 degrees steps-Fig. 39.

For instance if the set A contains all the cell ends from 0 degrees to 160 degrees and set B contains the cell ends from 180 degrees to 340 degrees (i.e. B(i)= A(i) + 180), we combined all the cell ends detections at A(i) to the ones detected at B(i). In order to achieve this, we used a naïve approach which consist of greedily combining the closest opposite ends (i.e. A(i) and B(i)).

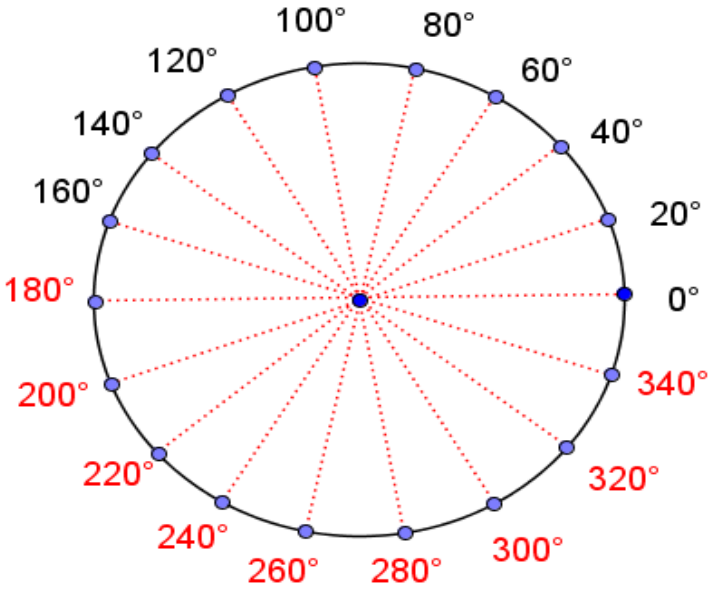

Figure 26. Greedy model establishment search space.

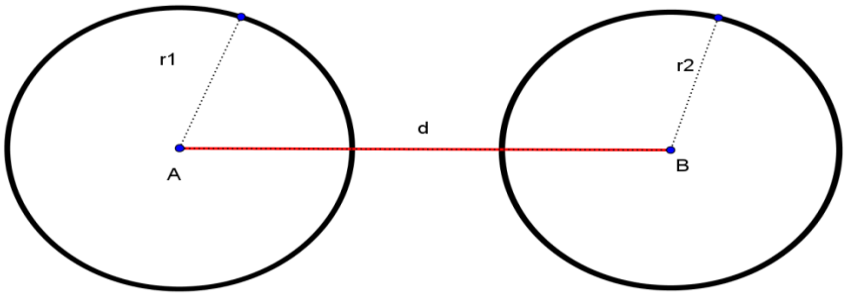

Figure 27. Yeast cell model- The cell ends A and B are combined together to form our model.

Furthermore, the reason why we decided to combine the models as we did, is because doing it this way allows us to learn the orientation of the cell of interest with respect of the x-axis of the image plane. And this information can then be passed to the PF filter so that we can learn how the cell orientation has changed over time- this type of feature could be useful when trying to infer the cells behaviours for instance.

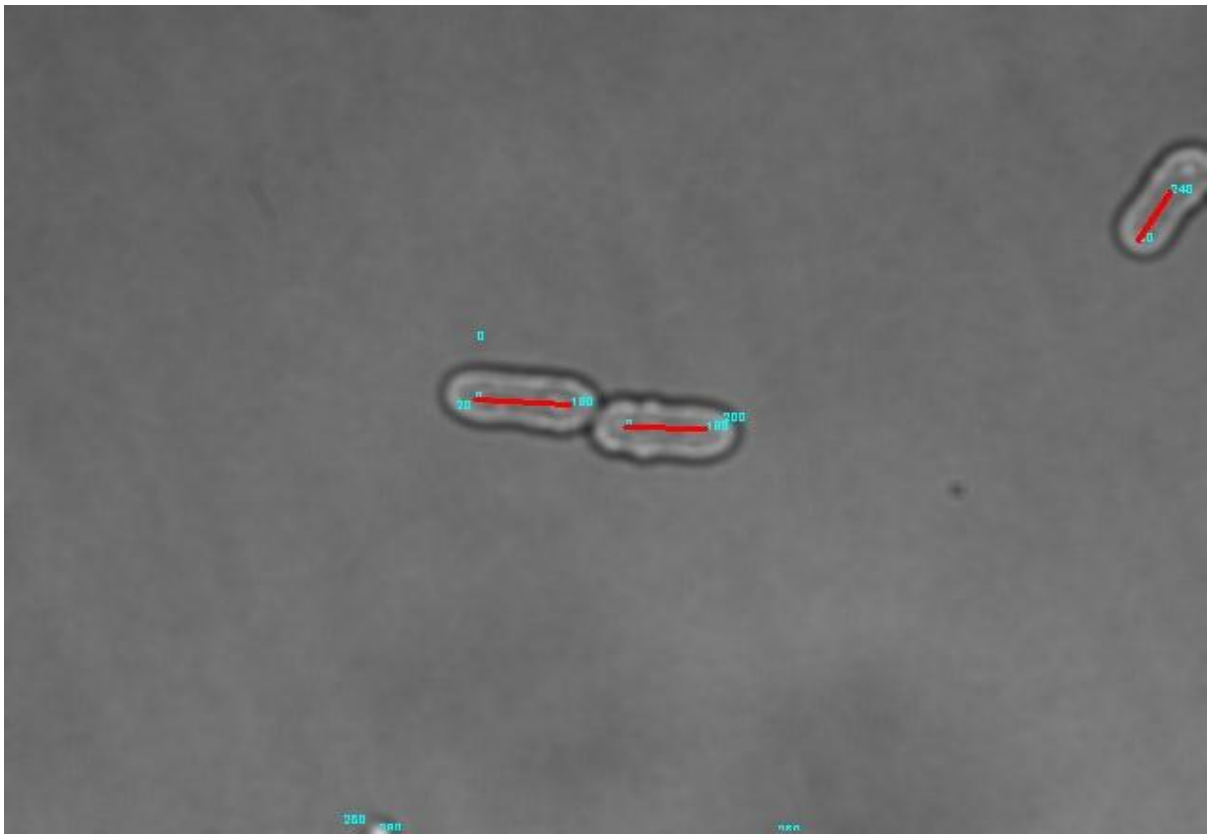

Figure 28. Constructed models- From left to right the first cell is constructed by combining detections at 0 degrees and 180 degrees, the second cell is constructed the same way as the first cell and the last cell is formed by joining detection at 60 degree and 240 degrees. The numbers in the image that are not connected by a red line represent detected cell end that have not been validate or that cannot be combined to form a model.

However, before applying the greedy model establishment, we had to convert/translate the pixels' coordinates of the detected cell ends at different orientations of the image frame to the standard orientation (i.e. 0 degrees to 180 degrees). This was achieved by applying the following transform which maps the centre of the image to itself.

$$\begin{cases} x' = xcos\alpha + ysin\alpha + (1 - cos\alpha)C_x - sin\alpha C_y \\ y' = -xsin\alpha + ycos\alpha + sin\alpha C_x + (1 - cos\alpha)C_y \end{cases} \quad (40)$$

Where (x',y') are the translated pixel coordinates, (C_x,C_y) are the coordinates of the centre of the image; (x,y) are the coordinates of the detected cell end at a specific orientation and α is the angle at which the detection was done (i.e. the orientation).

However, this greedy approach is very inefficient when used on its own because it sometimes builds/combines models that cannot exist in reality (see Fig. 29). Hence for this reason we devised a validation strategy which is discussed in the next section.

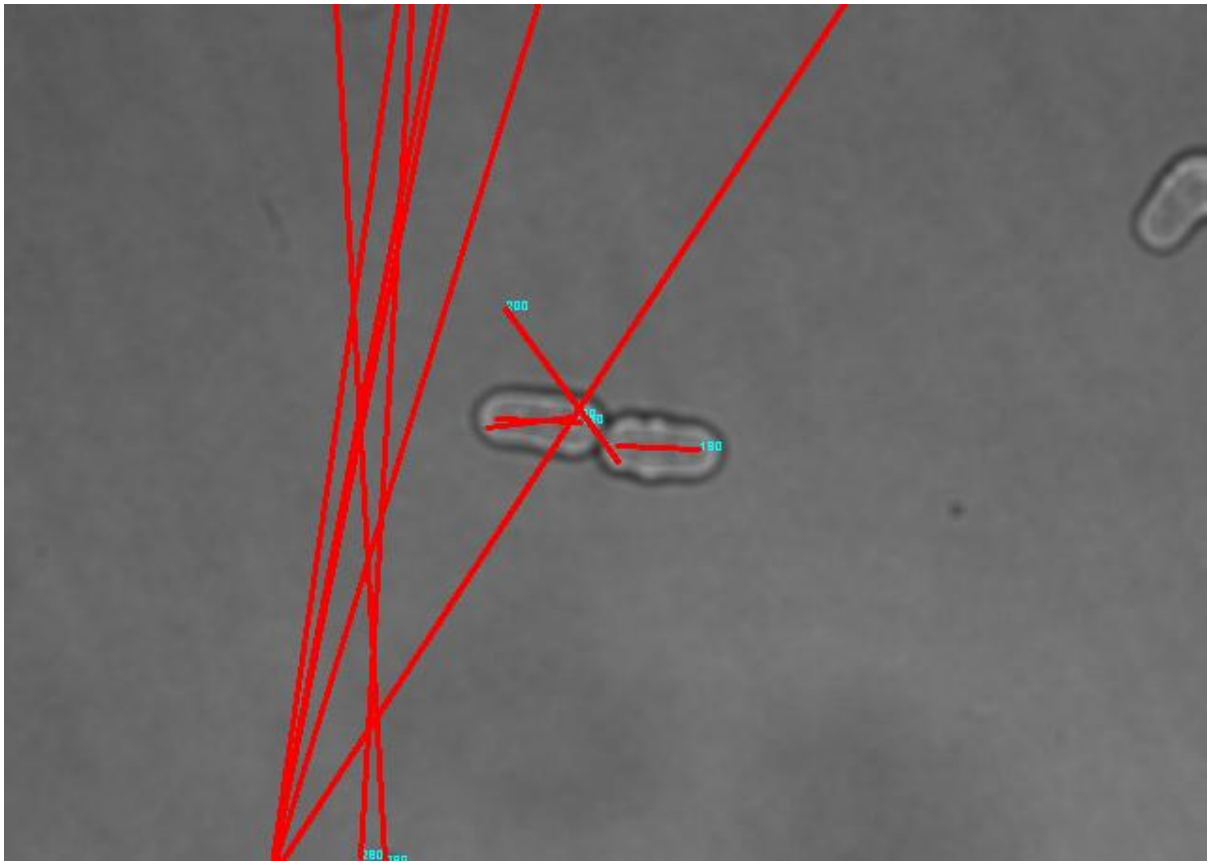

Figure 29. Examples of badly formed cell models.

Model establishment enhancement/Validation

As pointed out earlier, using just the naïve/greedy approach we noticed that some of the formed models cannot exist in reality (i.e. overlapping or crossing models). Therefore, in order to solve the problem we initially used a ***canny edge detector*** in order to extract the sharp edges of the cells and consequently filter out the models that cross the cells' boundaries. But as you can see in Fig. 30 we realised that the canny edge detector was not adequate for this task because not all the cell edges are formed; in fact, it provided us with some very ambiguous results.

Figure 30. Canny edge detection output.

Then, we used a distance constraint between the created models. That is for a new cell model to accepted or to be valid, it has to be located at (***0.5 * radius***) of any of the previously validate cells. We remarked that using this simple heuristic we were able to eliminate many of the wrongly formed models.

```
IF ((dist(NEW_MODEL,ALL_MODELS)<RADIUS_NEW_MODEL * 1.5) && NO_LINE_CROSSING==FALSE && d>65) {
                    DISCARD_NEW_MODEL();
}ELSE{
                    STORE_NEW_MODEL();
}
```

Figure 31. Model validation Heuristic.

However, the validation describes above does not guarantee that crossing models cannot be constructed. Thus, we added a method to avoid crossing models by constructing models which do not have any other models along their line segment. This was achieved by iterating through the line segment of the cell model and looking for red pixels; since our models are plotted in red this mean coming across any red pixel mean that if the new model is plotted it will cross with another already constructed model- this new constraint improve our heuristic(Fig. 32). Moreover, we set a maximum distance between the two ends (i.e. A and B) of each cell/model to ***65 pixels;*** which is the average cell/model length in our dataset. The latter was extracted from the Yeast cells' data because we noticed when the distance between two cell ends is greater than 65 it is highly likely that the cell would have already split.

3.3 Particle Filter for Model tracking

In order to track the Yeast cells every model that is validated by the heuristic is passed to the PF. Also, every particle is modelled as a line segment with its extremities being the two points A and B centres of the circles in Fig. 27. In addition, every particle contains the cell's orientation and the weight parameter. Moreover, for this part of the project we developed our own PF using the C language which is the basis of the OpenCV library. We decided to implement our own PF, despite of the fact OpenCV has an in-built function, because we wanted to have more flexibility during the development process- this means that we could easily tweak the PF in order to make it more performant or evaluate its performance more easily.

During the initialisation stage, the particles are uniformly distributed around the observed model (or the newly validated model) with their weights set to zero. Furthermore, the motion model/dynamic of the system is devised using the ***Box-Muller transform.***

The ***Box-Muller transform*** is used to generate a pair of independent and uniformly distributed random numbers with mean ***μ=0*** and standard deviation ***σ=1***, given a source of randomly distributed random numbers [33]. The transform is expressed as follow:

$$z_1 = \sqrt{-2lnx_1} * \cos(2 * \pi * x_2) \quad (41)$$

$$z_2 = \sqrt{-2lnx_1} * \sin(2 * \pi * x_2) \quad (42)$$

Where x_1 and x_2 are uniformly and independently distributed between 0 and 1.

Consequently, the extremities (i.e. A and B) were move according to the following equation/dynamic:

$$x(n+1) = x(n) + z_1 * step * std \quad (43)$$

$$y(n+1) = y(n) + z_1 * step * std \quad (44)$$

Where the ***step*** refers to the expected distance travelled by the cell from frame to frame; in this project we used ***step=10*** and ***std*** is the standard deviation (i.e. the spread of the particles around the observation)- the latter was set a 0.5.

Moreover, in order to compute the weights of each particle against the new observation, the circles centres (i.e. A and B) of the observation/detection were modelled as Gaussians with standard deviation equal to 1.

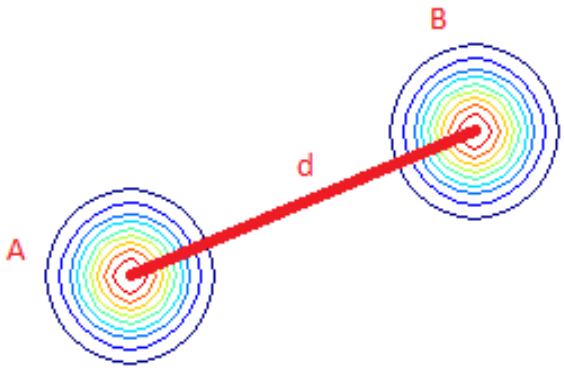

Figure 32. 2D Gaussians fitted around the points A and B.

And the end of every particle is sampled against the correspond end of the observation/detection. And the weight of each particle is updated as the product of the weights of the two ends -the more a particle is closer to the observation the more its weight will be close to 1 (see Eq. 45).

$$Wi = P(Ai) * P(Bi) \ (45)$$

$$\text{Where} \quad P(.) = exp\left(-\frac{(Aix-Ax)(Aix-Ax)+(Aiy-Ay)(Aiy-Ay)}{2}\right) \ (46)$$

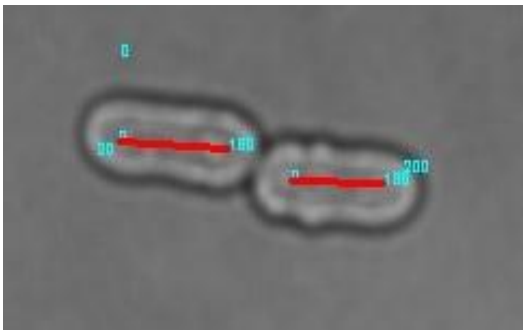

Figure 33. Detected/Observed Yeast cells- the number that are not linked together represents the orientation of the models that have not been approved by the validation heuristic.

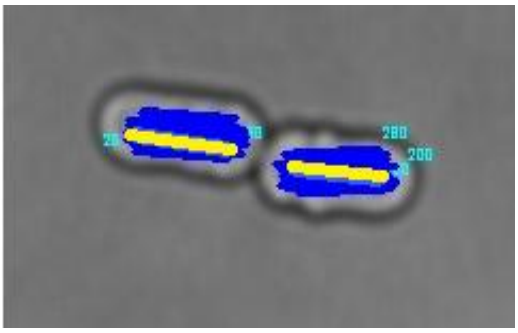

Figure 34. Tracked cells- the dark blue lines represent the spread of the particles around the new observation and the Yellow line is the estimated position by the PF.

Furthermore, the multi-target tracking data association problem was dealt with using the *Nearest Neighbour approach*. That is the new observations is associated or correspond to its closest

Particle filter Enhancement

Here also we tried to improve the performance of the PF tracking by exploring two approaches:

- ***Adding a damping factor to the weights of the particles:*** This means adding a small value to the weight of each particle when updating it. This will allow more particles to survive during the resampling process.

  $$W_k = W_k \quad + c \ (47)$$

  Where c is the damping factor.

- ***Local features tracking:*** The idea behind this approach is to improve the PF by tracking local features around each of the detected Yeast cells. Then, using that local information to prediction the position of new observations in subsequent video frames. The new estimated observation will then be used to compute the likelihood of the particles. We realised that this approach is very interesting as it is independent of the VJ detection/prediction of new observation. This mean we would only detect a cell once and it subsequent likely locations/observations will be computed by this approach.

In order to achieve this, we used the ***"Good features to track"*** method proposed by Shi and Tomasi [48]. The latter is one of the most popular techniques to such point and we combined it to the iterative Optical Flow method proposed by Lucas and Kanade for tracking.

The good feature to track works by considering the luminance *f(x,y)* of each pixel *(x,y)* within the image as a height denoted z – this allows the image to be represented as a curved plane. Then, the features points that are located where the rate of change of the z component is large are the best features for tracking. This rate is calculated by the equation below which is an operator of feature extraction [35].

$$Operator = \min(\lambda_1, \lambda_2) \quad (48)$$

$$C = \begin{pmatrix} G_\sigma(f_x^2) & G_\sigma(f_x f_y) \\ G_\sigma(f_x f_y) & G_\sigma(f_y^2) \end{pmatrix} \quad (49)$$

Where λ_1 and λ_2 are the eigenvalues of the matrix C; f is the input image. f_x and f_y are x component and y component of differential coefficient, respectively, and G_σ means Gaussian smoothing with standard deviation σ.

Furthermore, the ***Optical Flow tracking*** is based on the principle that the luminance remain constant along the motion trajectories which mean that the total derivative of the image is zero [47] The Optical flow is express as follow:

$$E_x v_x + E_y v_y + E_t = 0 \ (50)$$

Where E_x , E_y and E_t the spatial and temporal derivatives of the image are function and v_x and v_y are the components of the underlying optical flow.

However, because the Optical Flow contains two unknowns (v_x and v_y), it is impossible to compute the estimate of the optical flow field. But ***Lucas and Kanade*** introduces a method that can solve the previous problem by assuming that the motion field within a small window is constant and this constraint is used with the Optical Flow Equation to construct a least squares problem and hence estimates of the vector (v_x, v_y) [47]; this is the method used in this project to track the local features.

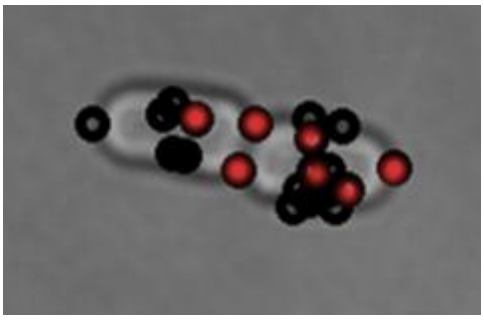

Figure 35. Detected good feature points on frame at time t.

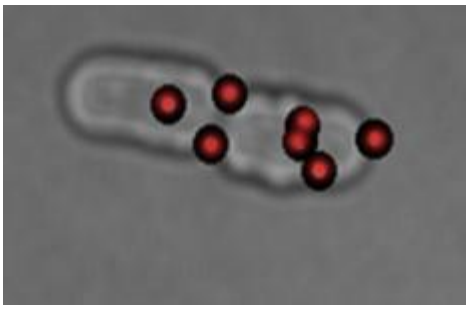

Figure 36. Tracked good feature points (i.e. in red) in subsequent image frame (t+1) using Optical Flow.

3.4 Summary

In this chapter, we have given an overview of the implementation of the system. Also, we have discussed both the challenges we encountered during the project as well as the solutions we proposed. For instance, to prepare our data for the VJ we did some pre-processing to remove the noise; and, to spread the pixels intensities uniformly around the image using histogram equalisation. Then the VJ was trained to only detect cell ends; the latter were extracted both manually and using template matching. The detected cell ends were grouped to form the cell models; which were then passed to the PF. Moreover, during the training of the VJ we edited the source code of the OpenCV library because the current version cannot merge the output of the ***opencv_createsample*** function into a single file.

In addition, we used a greedy approach to form a cell model by combining two opposite ends and we noted that building the Yeast cell's model without any validation is a weak approach so we devised a heuristic algorithm to tackle the problem. Moreover, we implemented our own version of the PF so that we can more flexibility over the code and also so that we will have a better understanding of how the PF works- the OpenCV implementation of the PF is ambiguous and we are not sure how it would perform in the case of multi-target tracking.

Likewise, we proposed a method of improving the performance of the PF by adding a damping factor the weights during their update. And another PF improvement was to extract local features (using the "*good features to track*" method) around the detected models. Then track them using Optical Flow; and, use their predicted location (by Optical Flow) in subsequent frames as the positions of the new observations of the cell model(s) of interest. Hence, by doing this the PF will be independent of the performance of the VJ in detecting new observations/models. The intuition here was to make the PF more performant and (almost) independent of the VJ detection.

4. Results

In this chapter we present the results obtained from the implementation phase. The chapter also discusses the challenges, successes and the short-comings of the techniques we employed.

4.1 Cell ends detection

We noticed that the results obtained during the detections of the cells' ends are not very good. But this is what was expected because detecting cell ends/corners is not a trivial task. And we think the main issue here is the fact that our training data is not very large (i.e. 186 samples). We tried to boost this training data to 2000 and 3000 samples by artificially generation samples from the original dataset; theses samples were created by adding random noise and random lighting conditions to the original samples. As we are aware the performance of the VJ training is proportional of the number of training data; the more training data we the more accurate our classifier will be[11]. However, this attempt the produce artificial samples was not successful because after running the VJ training for more than 10 days, the process failed to stop.- this means that the VJ could not find the thresholds that best separate positive and negative samples using the artificially generated data.

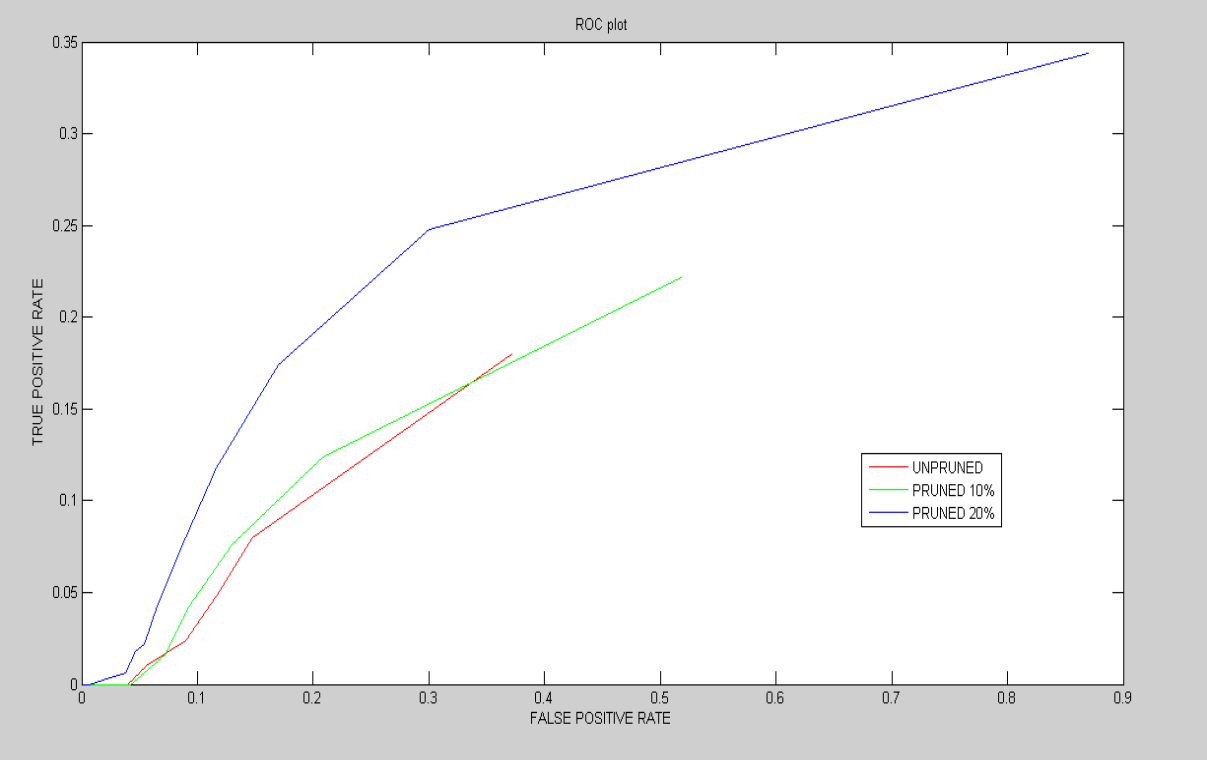

Figure 37. ROC plot of the VJ classifier (tested on artificial generated examples). The plot shows how the classifier performs on randomly generated Yeast cells' data.

Moreover, we investigated how the performance of the VJ classifier would change after pruning its decision tree. That is by reducing the thresholds of the stages of the classifier; we investigated 10% and 20% reductions. And we noticed that the performance of the VJ classification increased as we pruned the tree. Another reason that motivated us for doing this was to reduce the risk of over-fitting since we trained our classifier on a relatively small dataset. Thus, we wanted to make sure that it could recognise data it has not seeing before and for that reason we tested the classifiers on artificially generated samples- see Fig. 37.

4.2 Model tracking

Furthermore, we realised that the validation step during the model establishment was crucial to the project. The reason is the fact that without it many false positives of the VJ cell ends detection process could be joined together; this is illustrated in Fig. 29. And this is not want is wanted but we remarked that after applying the validation heuristic on the same image frame most of the unwanted build models are discarded-see Fig 38.

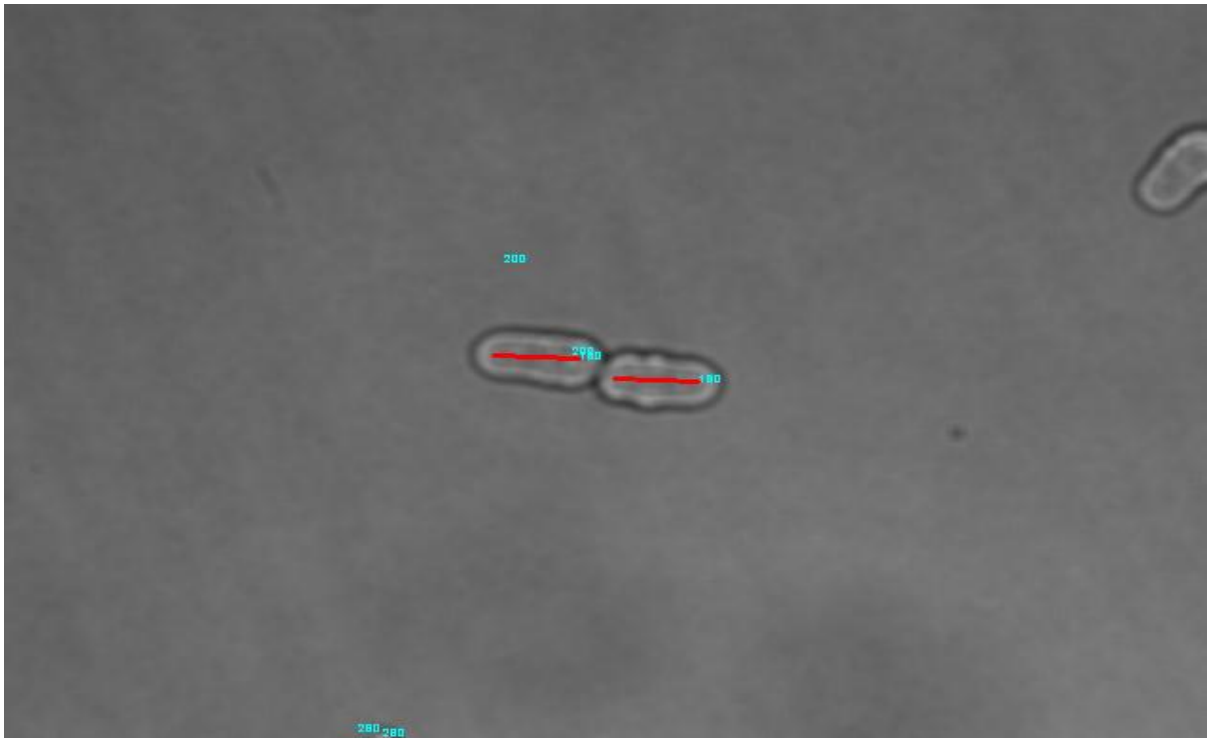

Figure 38. Validated models after validation heuristic has been applied to the greedy model establishment method; the numbers in the image that are not joined together have been filtered out by the validation heuristic.

We spotted that during the tracking of the established models, the PF outperformed both the basic model and the basic model (without validation) with the validation heuristic, after running it over our data set-see Fig 39.

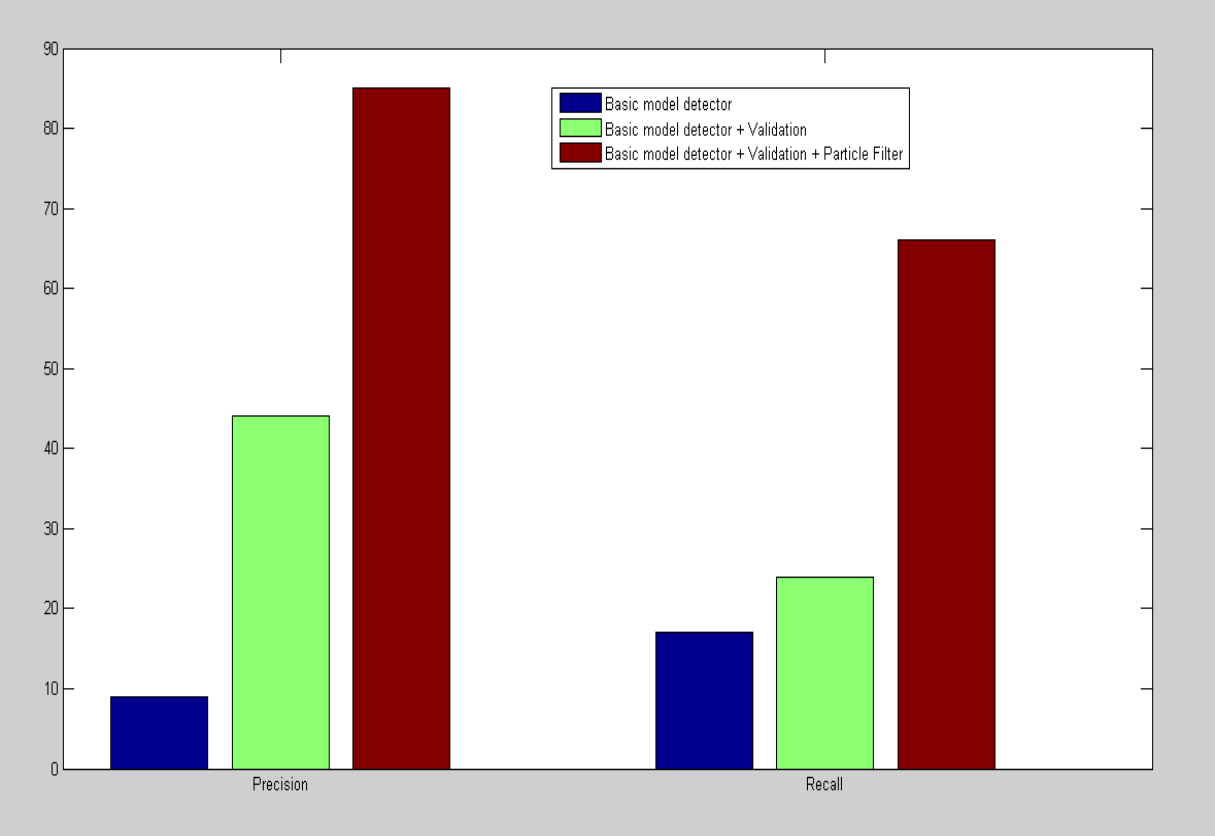

Figure 39. Model detection performance.

In fact, tracking using the PF returned more relevant results (60%) and we obtained a precision of 85%- the latter is almost an 8 times improvement on the basic model detector and double the performance of the basic model with validation. This improvement is due to the fact that even if the VJ failed to detect any cell ends (i.e. no model is build) the PF has already build the likelihood about were the cells are; in this situation the previous positions of the cell are used as their current positions. This advantage is strengthen by the fact that the Yeast cells do not move very often and when they do so the distance travelled is not very distant from their previous locations- this is illustrated in Fig. 40-42.

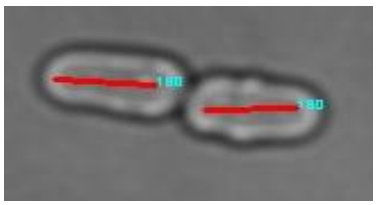

Figure 40. Validated models at time t.

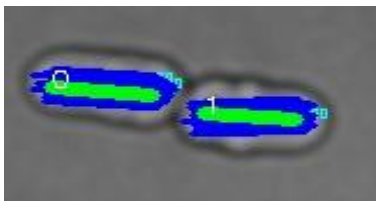

Figure 41. PF prediction in subsequent frame (t+1_- the blue plot represents the spread of the particles around the new observation.

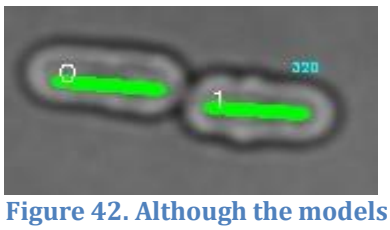

Figure 42. Although the models are not detected (t+2) the PF predictions are still valid.

Furthermore, in order to check statistical significance of our findings, we did some statistical testing on the results. We used the ***Student's T-test*** method to compare the performance of the methods presented in Fig. 39. Since, the algorithms are tested on the same data we did a paired t-test and we opted for one that is two-tailed. The Student t-test is used to check the following null hypothesis- that is how likely that the difference between the precision/recall of the methods has a mean of 0.

	T-test on precision	**T-test on recall**
PF detection vs. Basic model detector (without validation)	0.000198401	0.0001642
PF detection vs. Basic model detector with validation	1.43265×10^{-10}	2.92362×10^{-5}

Table 2. T-test results.

We found that (see Table 2) that the probabilities of all the t-tests we did are inferior to our p-value (i.e. 5%). Thus, we rejected the null hypothesis which means that the test has confirmed the statistical significance of our results. Therefore, it is unlikely that our results have happened by chance.

4.3 Yeast cell trajectories and PF improvement

To evaluate the cells' trajectories built by the PF we compared them to a benchmark (i.e. ground truth). We manually annotated the ground truth positions of 2 Yeast cells over 19 frames and we compared them to the predicted positions of the PF. We noticed that, when we add a ***dumping factor*** to the PF, when one cell is detected it does not move; this is due to the fact that, during the resampling process of the PF, many particles survive. So, when a model is established it will stay on almost forever at the same position if no new observation is not matched to it (i.e. it is not updated).

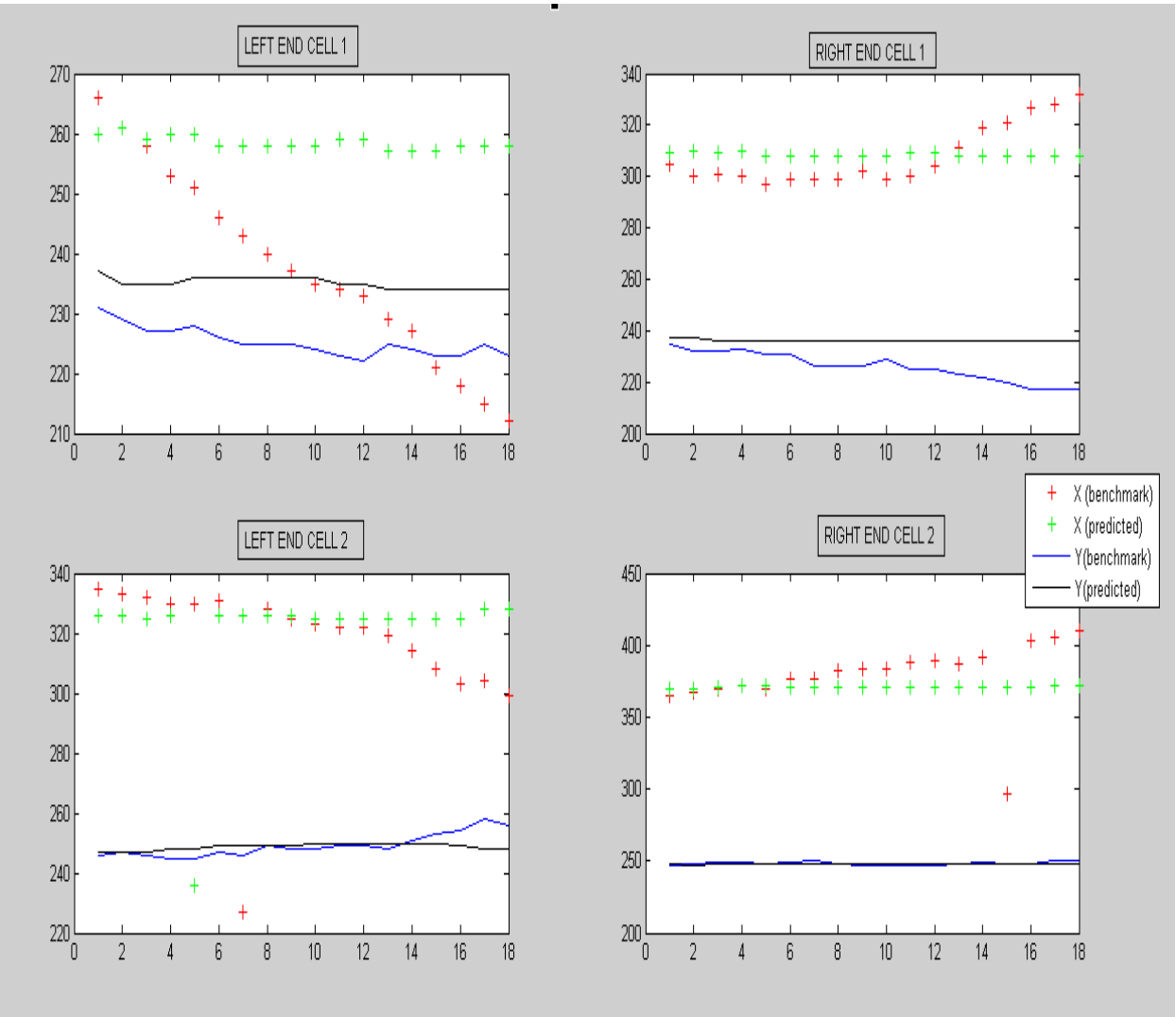

Figure 43. Yeast cell trajectories when using PF with damping factor; we notice here that the predicted position does not vary a lot.

However, this is not adequate for our project , since we have a validation heuristic that relies on a distance matrix , a model which stays on at the same position without being updated/removed will prevent other valid new observations/models from been accepted. Thus, we did not use the damping factor in the project and this explains why when the latter is remove the trajectories built by the PF follow better the dynamic the Yeast cells as more new observation/models are validated- this is illustrated in figure 44.

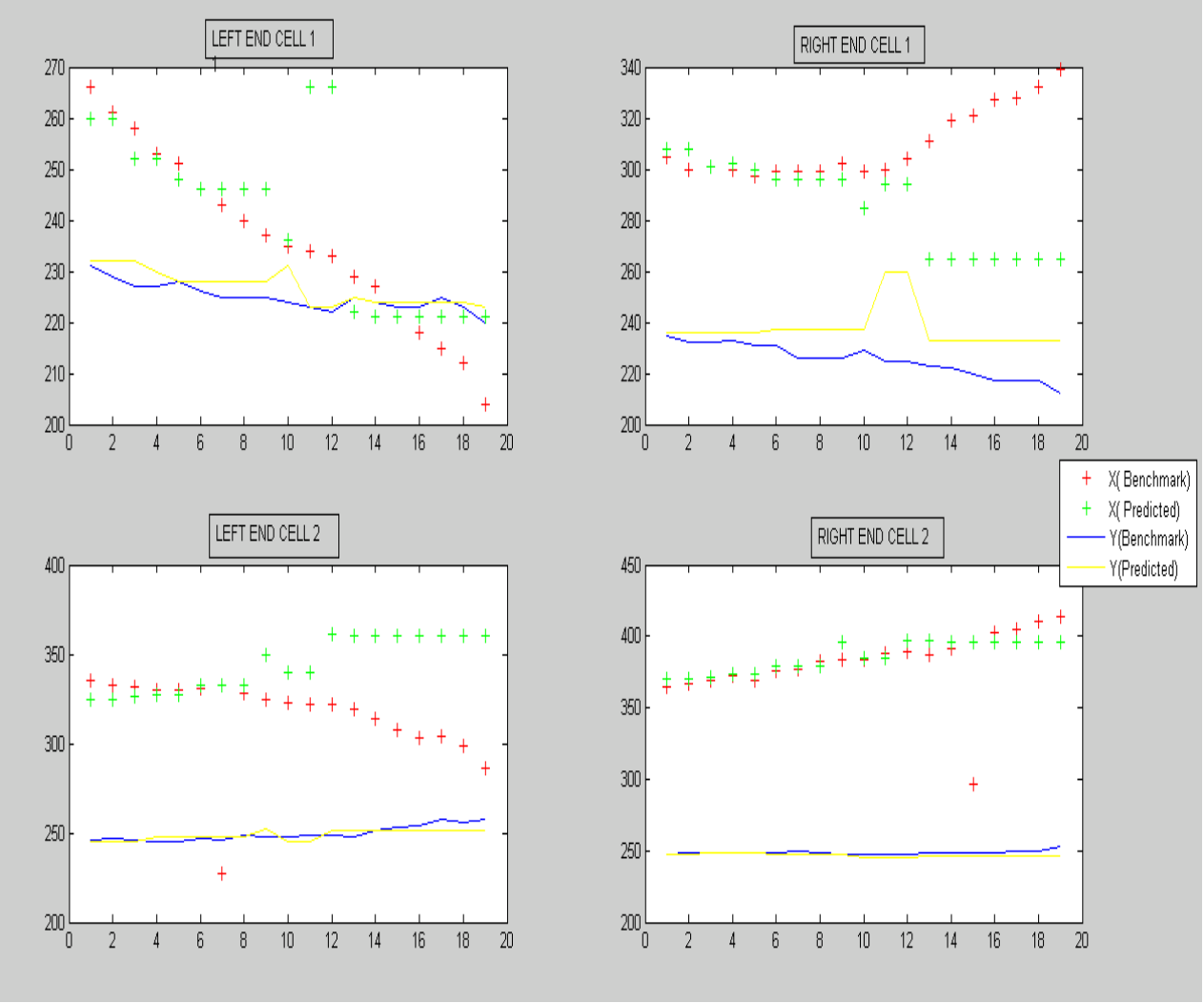

Figure 44. Yeast cell trajectories when using PF without damping factor.

The similarities between the trajectories built by the PF and the ground truths are summarised in the table below.

Left end cell 1	Right end cell 1	Left end cell 2	Right end cell 2	Overall similarity
91.66%	71.49%	68.88%	88.94%	80.24%

Table 3. Trajectories similarity results.

Therefore, we notice that the probability of losing one Yeast cell in 1 frame is approximately ***1.05%.***

4.4 Local features tracking

In addition, we remarked that the combination of the "*good features to track*" and the Optical flow provided descent results only in the first couples of frames. As the Yeast cells divide and the environment become more complex, the Optical flow tracking

returned very poor results (i.e. it was unable to track detected in the previous frames)- this is illustrated in figures 45 & 46.

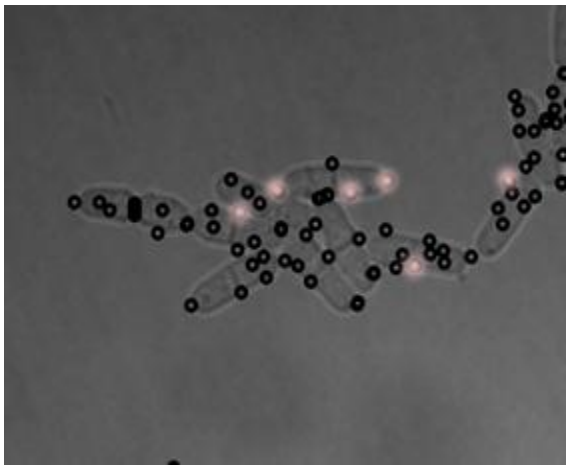

Figure 45. Detected good feature to track at time t- the tracked points at time t+1 are highlighted in light pink.

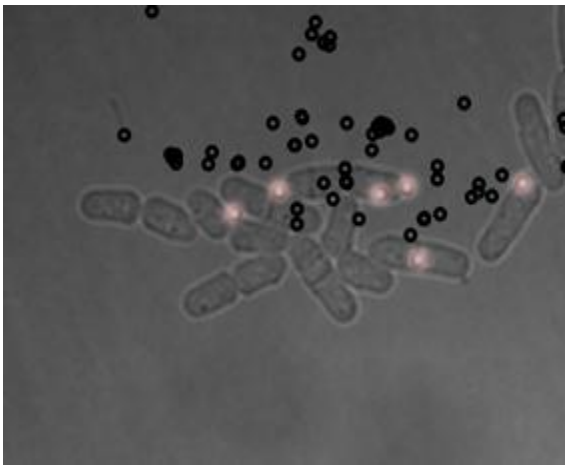

Figure 46. The Optical Flow fails to track the features from the previous video frame (i.e. frame X).

4.5 Summary

In this chapter we have presented the results we obtained during the project. In addition, we have highlighted the methods we retained and the ones we had to discard. In general, we have demonstrated that the VJ and PF can be combined together in the task of Yeast cells recognition and tracking.

Moreover, we presented the main challenges we encountered during the project. The first challenge was the fact that our training set was relatively small; we solved this difficulty by pruning the tree of the VJ classifier. And, we found that the tree pruning process improved the performance of the VJ cell ends detection. The latter technique was employed after the VJ failed to training on artificially generated positive samples.

The second challenge during the project was the model establishment; which we overcame by combining the greedy cell ends model formation with a model validation heuristic. We noted that the validation heuristic is paramount to our system as without it we would almost not have had any valid/realistic cell models.

We spotted that the PF significantly improved the tracking of the established models- it almost doubled the precision obtained by the basic model combined with the validation heuristic. And we confirmed the statistical significance of our results using the ***Student's t-test*** method; we found that in all cases the probability of the t-test is less that our p-value (i.e. 5%).

Also, we have realised that the both the damping factor and the local feature tracking (using good features to track and Optical flow) are not suitable for our project as they did not produce satisfactory results.

5. Concluding remarks, evaluation and Further work

5.1 Object detection

One of the main objectives of the project was to develop a system that can automatically and in real-time detect Yeast cells. And with the results we have obtained, we can argue that this objective has been successfully achieved. Also, with the development of the Yeast cell detection system, the main contribution has been our approach. In fact, we have been able to separate the VJ detection process and the model establishment. And in doing so, we have devised a novel model establishment and validation heuristic that have not, in our knowledge, been employed anywhere else in the literature.

Another suggestion, we raised during the planning of the project was the ability of the system to detect mitotic events (i.e. cell division) in order to build their lineage as a basis for later pedigree tree extraction. At present the system is sensible to mitotic events but it cannot build the lineage because at this stage every mitosis is treated as the creation of new cells. However, this could be solved by assigning (lineage wise) any newly formed cell to the closest cell that has disappeared for it to be validated. In addition, we used a distance constraint between the two ends of the cell in order to check if they still belong to the same model- after a certain distance the likelihood of the cell ends belonging to two different entities is very high and this threshold can be obtained from the data.

Moreover, during the VJ training, we have also extended the C interface of the OpenCV library so that we can merge the output of the ***opencv_createsamples*** in a single file- this was not available in the current version of the library.

5.2 Cell tracking

Another objective of the project was the ability was the ability to track the Yeast cells in order to build their trajectories. Also, this objective has been met and we have implemented our own PF using the C interface of the OpenCV library. This custom implementation was done so that we would have more flexibility over the implementation and also so that we would have a better understanding of the mechanism behind the PF. In addition we found the implementation of the PF in the OpenCV library very complex to set up and we were not sure how it would perform in the case of multi-target tracking. We found that the choice of the PF was a good choice as it significantly improved the performance of our system. And, also we concluded,

after analysing the performance of the PF (see Fig. 39), that it could be used as a basis for later pedigree tree extraction.

5.3 Further work

Some of the things worth investigating are for instance the amelioration of the cells' ends detection as this would have a significant impact on the overall performance of the whole system. We believe that the cell ends detection could be improved by pruning down further the tree of the VJ classifier. Doing so would in deed generate many false positives and true positives but the intuition here is to filter out all the false positives using a binary classifier such as ***Support Vector Machines*** (SVM). The SVM would be trained to discriminate between false positives and true positives and we believe that this approach could improve the overall performance of the detection system.

In addition, an interesting thing to investigate further is to try to learn/predict the mitotic events of the Yeast cell from their built trajectories. One could mark on the trajectories when the each cell undergo a mitosis and used other machine learning techniques such as ***Hidden Markov Models*** to try predict when the next mitosis is likely to happen.

Moreover, it would be interesting to investigate how the proposed system would perform against other object detection/tracking systems- one could compare it to the ***Generalize Hough Transform*** although we know that the Hough transform is computationally expensive.

5.4 Summary

During this project, we found that separating the Yeast cells' ends detection, the model establishment, the model validation and the PF tracking have been paramount to both to its successful delivery as well as to the analysis of the results. It has helped us to evaluate how the performance of the system has improved after each step.

To sum up, the main novelty and contribution of the project is the research, development and evaluation of a real-time visual system linking VJ and PF for Yeast cells detection and tracking. We expended on this framework further by devising models for Yeast cell establishment and validation (without a need for image segmentation).

6. References

1. Shen, H. et al., 2006. Automatic tracking of biological cells and compartments using particle filters and active contours. Chemometrics and Intelligent Laboratory Systems, 82(1-2), p.276-282.

2. Bise, R. et al., 2009. Reliably Tracking Partially Overlapping Neural Stem Cells in DIC Microscopy Image Sequences. Histopathology, p.67-77.

3. Forsyth, D.A. & Ponce, J., 2002. Computer Vision: A Modern Approach, Prentice Hall. Available at:

4. Ostergaard, S., Olsson, L. & Nielsen, J., 2000. Metabolic engineering of Saccharomyces cerevisiae. Microbiology and Molecular Biology Reviews, 64(1), p.34-50

5. Degerman, J. et al,. 2006. A Comparative Study between Level Set and Watershed Image Segmentation for Tracking Stem Cells in Time-Lapse Microscopy. MIAAB 2006, (C), p.1-5.

6. Szeliski, R., 2010. Computer Vision : Algorithms and Applications R. Szeliski, ed. Computer, 5(3), p.832.

7. Chen X., Zhou X., Wong STC.,2006. Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy. IEEE Transactions on Biomedical Engineering 53: 762–766.

8. Rapoport, D.H. et al., 2011. A novel validation algorithm allows for automated cell tracking and the extraction of biologically meaningful parameters. M. J. Goumans, ed. PLoS ONE, 6(11), p.e27315.

9. Cohen, A.R. et al., 2010. Computational prediction of neural progenitor cell fates. Nature Methods, 7(3), p.213-8.

10. Mouroutis, T., Roberts, S.J. & Bharath, A.A., 1998. Robust cell nuclei segmentation using statistical modelling. Bioimaging, 6(2), p.79-91.

11. Viola, P. & Jones, M., 2001. Robust Real-time Object Detection. International Journal of Computer Vision, 57(2), p.137–154

12. Li, K. et al., 2008. Computer vision tracking of stemness. 2008 5th IEEE International Symposium on Biomedical Imaging From Nano to Macro, p.847-850.

13. Han, J.W. et al., 2010. The application of support vector machine classification to detect cell nuclei for automated microscopy. Machine Vision and Applications, 0932-8092, p.1-11.

14. Dalal, N. & Triggs, W., 2004. Histograms of Oriented Gradients for Human Detection C. Schmid, S. Soatto, & C. Tomasi, eds. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR05, 1(3), p.886-893.

15. Ray, N. & Acton, S.T., 2002. Active contours for cell tracking. Proceedings Fifth IEEE Southwest Symposium on Image Analysis and Interpretation, p.274-278.

16. Li, K. et al., 2006. Online Tracking of Migrating and Proliferating Cells Imaged with Phase-Contrast Microscopy. 2006 Conference on Computer Vision and Pattern Recognition Workshop CVPRW06, 00(c), p.65-65.

17. Banitalebi, B. & Amiri, H., 2008. An Improved Nearest Neighbor Data Association Method for Underwater Multi-Target Tracking. Aerospace, p.0-3.

18. Kachouie, N.N. et al., 2006. A Statistical Thresholding Method for Cell Tracking. IEEE International Symposium on Signal Processing and Information Technology, p.222-227.

19. Li, P. & Wang, H. 2005. Probabilistic object tracking based on machine learning and importance sampling. In Proceedings of the Second Iberian conference on Pattern Recognition and Image Analysis - Volume Part I (IbPRIA'05), Jorge S. Marques, Nicolás Pérez de la Blanca, and Pedro Pina (Eds.), Vol. Part I.

20. Tang, C. et al. 2011. Tracking of Active Cells Based on Kalman Filter inTime Lapse of Image Sequences of Neuron Stem Cells. In WorldComp 2011 Proceedings.

21. Debeir, O. et al., 2005. Mitotic Tree Construction by Computer In Vitro Cell Tracking: a Tool for Proliferation and Motility Features Extraction. EUROCON 2005 The International Conference on Computer as a Tool, p.951-954.

22. Sinop, A.K. & Grady, L., 2007. A Seeded Image Segmentation Framework Unifying Graph Cuts And Random Walker Which Yields A New Algorithm. IEEE 11th International Conference on Computer Vision (2007), Im(15213), p.1-8.

23. Ulrich, M., Steger, C. & Baumgartner, A., 2003. Real-time object recognition using a modified generalized Hough transform. Pattern Recognition, 36(11), p.2557-2570.

24. Bhattacharyya, D. et al., 2009. Biometric authentication: A review. Biometric Technology Today, 2(3), p.13-28.

25. Sezgin, M. & Sankur, B., 2004. Survey over image thresholding techniques and quantitative performance. Journal of Electronic Imaging, 13(January), p.146 - 165.

26. Kurtzman, C.P. & Fell, J.W. 2006. Yeast systematics and phylogeny - implications of molecular identification methods for studies in ecology. In: Rosa, C.A. and Peter, G., editors. The Yeast Handbook. Germany:Springer-Verlag Berlin Herdelberg. p. 11-30.

27. Ke, Y., Sukthankar, R. & Hebert, M., 2005. Efficient visual event detection using volumetric features. Tenth IEEE International Conference on Computer Vision ICCV05 Volume 1, 1(18), p.166-173.

28. Jones, T.R., Carpenter, A. & Golland, P., 2005. Voronoi-based segmentation of cells on image manifolds. Computer Vision for Biomedical Image Applications, 3765, p.535-543.

29. Yilmaz, A., Javed, O. & Shah, M., 2006. Object tracking: A survey. ACM Computing Surveys, 38(4), p.13.

30. Bishop, C. (2007) Pattern Recognition and Machine Learning, Springer.

31. Viola,P.,Jones,M. (2001)Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 2001, 1(C), p.I-511-I-518.

32. Ephraim, T., Himmelman, T. & Siddiqi, K., 2009. Real-Time Viola-Jones Face Detection in a Web Browser. 2009 Canadian Conference on Computer and Robot Vision, p.321-328

33. Golder, E.R. & Settle, J.G., 1976. The Box-Muller Method for Generating Pseudo-Random Normal Deviates. Journal of the Royal Statistical Society Series C Applied Statistics, 25(1), p.12-20

34 Jaward, M. et al., 2006. Multiple object tracking using particle filters. 2006 IEEE Aerospace Conference, p.1-8.

35. Sugano, H. & Miyamoto, R., 2009. Parallel Implementation of Good Feature Extraction for Tracking on the Cell Processor with OpenCV Interface. 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, p.1326-1329

36. Li, G.L.G., Xu, Y.X.Y. & Wang, J.W.J., 2010. An improved AdaBoost face detection algorithm based on optimizing skin color model. Natural Computation ICNC 2010 Sixth International Conference on, 4, p.2013-2015.

37. Kodama, T., Yamaguchi, T. & Harada, H., 2010. A method of object tracking based on particle filter and optical flow to avoid degeneration problem. SICE Annual Conference 2010 Proceedings of, 1(3), p.1529-1533.

38. Berloo, R.V. et al., 2007. An Online Potato Pedigree Database Resource. Potato Research, 50(1), p.45-57.

39. Kratz, R.F. & Siegfried, D.R., 2010. Biology For Dummies, 2nd Edition T. Gallan, L. Lefevere, & J. Tebbe, eds., Wiley Publishing, Inc., Indianapolis, Indiana, Page 109.

40. Koschwanez, J. et al., 2005. Automated lifetime analysis of a single yeast cell.IEEE International Conference on Automation Science and Engineering 2005, p.13-18.

41. McMurray, M.A. & Gottschling, D.E., 2003. An age-induced switch to a hyper-recombinational state. Science, 301(5641), p.1908-1911.

42. Blake, A. (n.d.). Introduction to Active Contours and Visual Dynamics [ONLINE]. Available: http://www.robots.ox.ac.uk/~ab/dynamics.htmlBlake. [Last accessed 10th May 2012].

43. BBC. (n.d.). Disease - Microbes. [ONLINE] Available at: http://www.bbc.co.uk/schools/ks3bitesize/science/organisms_behaviour_health/disease/revise2.shtml. [Accessed 26 September 12].

44. National Health Museum. (n.d.). Understanding Gene Testing. [ONLINE] Available at: http://www.accessexcellence.org/AE/AEPC/NIH/gene14.php. [Accessed 26 September 12].

45. Isard, M. & Blake, A., 1998. CONDENSATION - Conditional Density Propagation for Visual Tracking. International Journal of Computer Vision, 29(1), p.5-28.

46. Matthies, L., Kanade, T. & Szeliski, R., 1989. Kalman filter-based algorithms for estimating depth from image sequences. International Journal of Computer Vision, 3(3), p.209-238.

47. Calway, A. 2012. Computer Vision Lecture notes. Department of Computer Science University of Bristol.

48. Shi, J. & Tomasi, C., 1994. Good features to track. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR94, 94(June), p.593-600.