# The Development Of Generative Bayesian Models For Classification Of Cell Images

A thesis submitted to University College London

for the degree of Master Of Philosophy

Ashraf El-Shanawany

## Declaration

I, Ashraf El-Shanawany, confirm that this thesis is an account of work carried out for the degree of MPhil between October 2005 and December 2008. The work is my own original work, except where appropriate references are given in the text. This work has not been submitted for consideration for any other degree.

# ABSTRACT

A generative model for shape recognition of biological cells in images is developed. The model is designed for analysing high throughput screens, and is tested on a genome wide morphology screen. The genome wide morphology screen contains order of $10^4$ images of fluorescently stained cells with order of $10^2$ cells per image. It was generated using automated techniques through knockdown of almost all putative genes in *Drosphila melanogaster*. A major step in the analysis of such a dataset is to classify cells into distinct classes: both phenotypic classes and cell cycle classes. However, the quantity of data produced presents a major time bottleneck for human analysis. Human analysis is also known to be subjective and variable. The development of a generalisable computational analysis tool is an important challenge for the field. Previously cell morphology has been characterized by automated measurement of user-defined biological features, often specific to one dataset. These methods are surveyed and discussed. Here a more ambitious approach is pursued. A novel generalisable classification method, applicable to our images, is developed and implemented. The algorithm decomposes training images into constituent patches to build Bayesian models of cell classes. The model contains probability distributions which are learnt via the Expectation Maximization algorithm. This provides a mechanism for comparing the similarity of the appearance of cell phenotypes. The method is evaluated by comparison with results of Support Vector Machines at the task of performing binary classification. This work provides the basis for clustering large sets of cell images into biologically meaningful classes.

## Acknowledgements

I would like to thank my biology supervisor, Buzz Baum, from whom I have learnt a lot. Simon Prince was my computer science supervisor during the period from 2005 to 2007, when the project was in its formative stages. Simon Arridge was my computer science supervisor from the beginning of 2008.

I would like to thank the EPSRC for their generous support.

I am grateful, beyond the power of words to express, for the unending support of my family and friends during this period.

CONTENTS

# LIST OF FIGURES

# 1. INTRODUCTION

Since the invention of the first microscope in the 17th century it has been possible to directly observe cells. Observation of cells is an extremely important source of information about living organisms. Cell imaging remains a mainstay tool of biological research, although the image quality and interrogative techniques have advanced very significantly since the first microscope. However, the mode of analysis of this source of information has not changed fundamentally. A trained researcher looks at and interprets the images based on his or her experience. The increasing scope of experiments has made this a rate limiting step, to the point that large amounts of the data are not comprehensively examined. However, the advent of affordable computers allows for an alternative mode of analysis. Computational analysis of cell images is a highly desirable goal for reasons of speed, objectivity, and standardization, but how to substitute a computer for a human expert is far from apparent. A major question is "what should computational analysis be seeking to achieve?" There are many possible answers to this question. A vital part of analysing cell images is frequently cell recognition and classification. In this thesis methods of recognising and classifying cell images are examined, and a novel method is developed. The developed technique is tested on a high content RNAi *Drosophila melanogaster* screen [40].

## *1.1 RNAi as interrogative tool*

RNA interference (RNAi) is a naturally occurring phenomenon in which gene action is "silenced" post-transcription via degradation of messenger RNA (mRNA). This process can be hijacked by a researcher to silence the action of specific genes.

In 1998 Fire & Mello [24] discovered that the presence of double stranded RNA causes degradation of homologous mRNA via endogenous cell processes. The mechanism of action which they expounded is known as RNA interference. RNAi can be used to diminish the translation of any protein from mRNA. This allows the design of genome wide experiments in which the function of every protein, that is coded for in a genome, can be analysed by preventing its formation in cultured cells and comparing the resultant cells with wildtype cells.

RNA interference occurs by long double-stranded RNA strands being degraded into 21-23 base pair double stranded segments, called small interfering RNAs (siRNAs), which is catalysed by the enzyme Dicer. One strand of the siRNA is incorporated into a complex with cellular proteins called the RNA induced silencing complex (RISC). The siRNA in the RISC complex acts as a targeting mechanism to degrade mRNA containing a region of base pair complementarity to the RNA strand in the RISC. Fire & Mello showed that the process is catalytic so only a few molecules of dsRNA are necessary to completely silence expression of a complementary gene [24].

Practically RNAi can be employed as a technique to "silence" expression of RNA based on its sequence. Levels of the associated protein are depleted over time following the corresponding degradation of the mRNA by the siRNA, and the temporal depletion of the protein. This can be more difficult to interpret than the loss of function due to gene knockout. However, this weakness can also be viewed as an advantage allowing for the possibility of measurement of the degree of loss of function at differing protein levels. With the availability of genomic sequence data, RNAi has been shown to be effective as an injected tool for preventing the expression of specific mRNA sequences in *Drosophila*, Caenorhabditis elegans, plants and, more recently, mammalian cell cultures.

RNAi offers massive time benefits over conventional mutational analysis methods. The ability to selectively target multiple genes for silencing based on sequence alone makes a far wider scope of possible experiments feasible. Additionally, since there is no need to develop mutations to the required genes in order to study the effects, there is no restriction to classic (more experimentally tractable) genetic models, such as yeast, so it is possible to experiment on a wider range of plants and animals. Critically, it can also be used in cell culture, allowing for much easier application. This bypasses the need to keep organisms *in vivo* in order to do genetic knockdown studies upon them.

### *1.1.1 Non-specificity*

Gene products other than those of the targeted gene can also be affected by RNAi; this is referred to as off−target effects. Any siRNA chosen to deplete expression of a given gene will also act upon any other gene with the same sequence strand contained within its coding region. The problem is more severe than this: sequences with only partial homology to the duplex RNA can also be degraded. Sequences with homology as low as 16 matching base pairs can be degraded. One experimental method to mitigate for this effect is to use multiple siRNAs in independent experiments for each gene. This gives two distinct tests of knocking out the same gene. If the same effect is noted with both siRNAs, then it is very likely that the targeted gene is responsible

for the observed effect. It is generally agreed that the chance of two genes sharing two regions of sequence homology is sufficiently low that this method suffices to validate the effect of a gene.

### *1.2 The RNAi genomic-dataset*

As described above, RNAi can be performed on cells in culture, so that RNAi can be used as a tool to investigate the entire genome of an organism. In such a process each putative gene can be targeted for RNAi silencing by bathing cells in an appropriate siRNA. A genome-wide screen has been carried out in *Drosophila* S2R+ and $Kc_{167}$ cell lines [40]. Automated images were taken of the resultant cells five days after treatment.

The purpose of the experiment was to study which genes affect the morphology of cells. Hence, the proteins tubulin and actin, as well as DNA, were stained prior to imaging. In all of these images, the DNA has been stained with 4',6-diamino-2-phenylindole dihydrochloride (DAPI), tubulin has been stained using a tubulin antibody coupled with fluorescein isothiocyanate (FITC) and actin has been stained with an actin antibody coupled with tetramethylrhodamine isothiocyanate(TRITC). These dyes show emission spectra peaks giving blue, green and red light, respectively. Both tubulin and actin are structural polymeric proteins with a dynamic nature in living cells. Throughout this thesis, colours in cell images will consistently represent the same cellular structures, unless otherwise indicated. An example of the cell staining is shown in Figure 1.1. This cell is a single cell extracted from an image containing many cells, since sites containing many cells were imaged in the screen, rather than individual cells. Example images from the *Drosophila melanogaster* screen are shown in Figures 1.2 & 1.3.

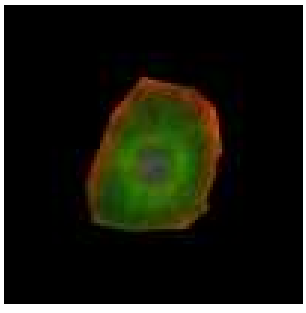

*Fig. 1.1:* **Microscope image of an individual fluorescently stained cell:** The actin edge of the cell can be seen as a red border, the nucleus is blue and the tubulin part of the cytoskeleton is green.

The goal of the screen was to discover the effects of each gene in the fly genome on cell morphology. To do this 58 plates were prepared, with each plate containing 396 wells. Each well contained a specific RNAi treatment condition.

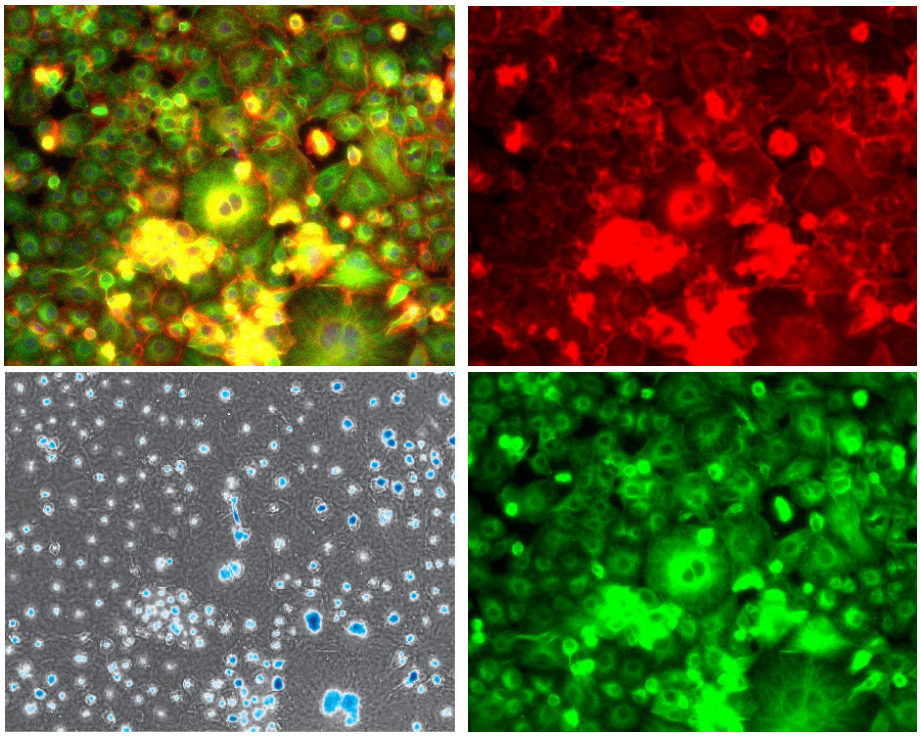

*Fig. 1.2:* **RNAi screen images**: Top left: Merged RGB image of typical fly cells (S2R+ cells). The actin appears red (stained with FITC), tubulin green (stained with TRITC) and DNA blue (stained with DAPI). Top right: Red channel of this image. Bottom left: Blue channel of this image. Bottom right: Green channel of this image

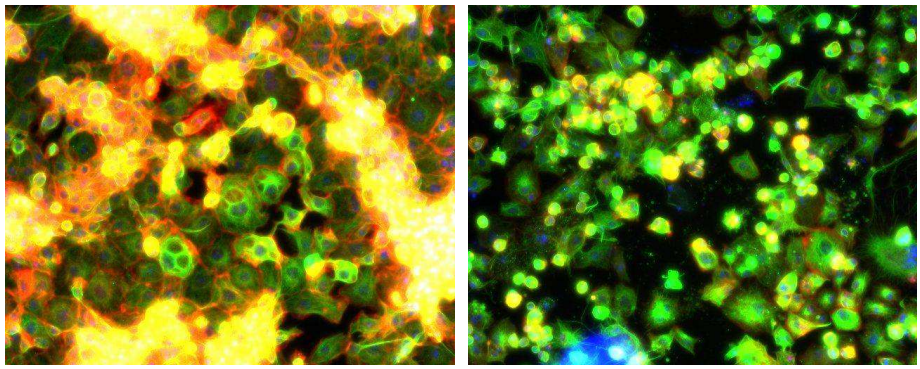

*Fig. 1.3:* **Example abnormal cells (S2R+ cells)**: Left: Multi-nucleate cells - The cells have failed to divide and complete cytokinesis. Right: "Round" cells - The cells do not adhere as strongly to the surface, so appear rounded. Also visible in the bottom left is a blue staining artifact.

From each well, two images were obtained from two different locations in the well. Each site within a well typically contained order of 100 cells. On each plate there was also at least one control well, in which no dsRNA was used. Throughout the project the cells in the control wells are referred to as wildtype cells. This setup of the image dataset is shown in 1.4.

The problem of computational analysis of such a set of data is non-trivial. Cell images in general present a difficult analysis task even for skilled human experts. As noted in [72], "Different operators provided with the same data

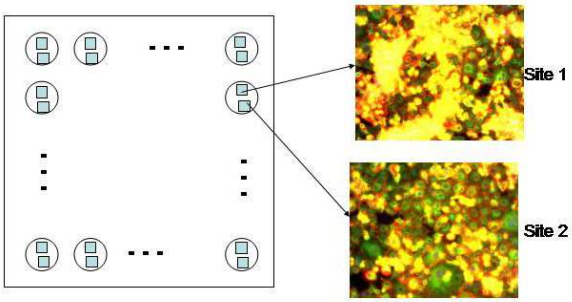


*Fig. 1.4:* **Genome wide screen setup**: A single plate is shown on the left. The plate is covered in wells, and each circle represents a well on that plate. Images were taken from two locations in each well, referred to as site 1 and site 2. Sample site 1 and site 2 colour combined images are shown on the right.

set commonly report significantly different results on activities as simple as cell counting. Indeed, the same operator often reports different results on the same image on different days". Humans are also poor at quantitative analysis of image features. The image set [40] took six weeks of man time to analyse by eye. More time could have been spent on each image, which would have yielded more detailed information.

In theory each different gene knocked down via RNAi can yield a different phenotype. This means there are potentially order of 20,000 phenotypes in the dataset. Given that the overall aim is to partition the image set into distinct phenotypic classes this strongly influences the computational approach that we develop. The method that we develop must be flexible enough to handle many potential classes of image, some of which may be only subtly different. Further the method should not make many assumptions about the content of the images, since there is great variation across the image set. The goal here is to develop generalisable algorithms that allow for computational analysis that will provide quantitative, consistency and speed advantages over human analysis.

## *1.3 Overview*

Thousands of fly cell images containing cells with different genes knocked out have been produced [40]. RNA interference (RNAi) has been used as a tool to knock out the genes. Analyzing large image datasets in an automated fashion is an open problem. The contribution of this thesis is to describe the development of a novel generative cell model that is described in Chapter 4. The application of this new model to this specific dataset is demonstrated by using it for cell classification.

Chapter 2 outlines the relevant literature. Chapter 3 describes the segmentation of normalised cell regions from the images. The segmentation is

achieved through the use of binary thresholds, morphological operators and distance transforms. Chapter 4 describes the development of a novel patch-based generative model. The model is developed from a Gaussian model of data. I build up to the eventual model in discrete stages, adding one extra feature to the model at each stage. For each model I demonstrate how to use the Expectation Maximisation algorithm to learn the parameters of the models. Working through the Expectation Maximisation equations at each stage of development highlights the similarities between the models and what has been added to the model, allowing the eventual model to be related back to a simple Gaussian model of data. Chapter 5 contains classification results and a comparison to the performance of Support Vector Machines on the same training and test data. Chapter 6 explains how this work can be used to create an overall classification system for a large dataset with a large unknown number of unknown classes of objects, such as we find in our dataset [40].

Figure 1.5 shows a paradigm for advancing understanding of cell biology. The work presented here seeks to address the top right and bottom left components of Figure 1.5.

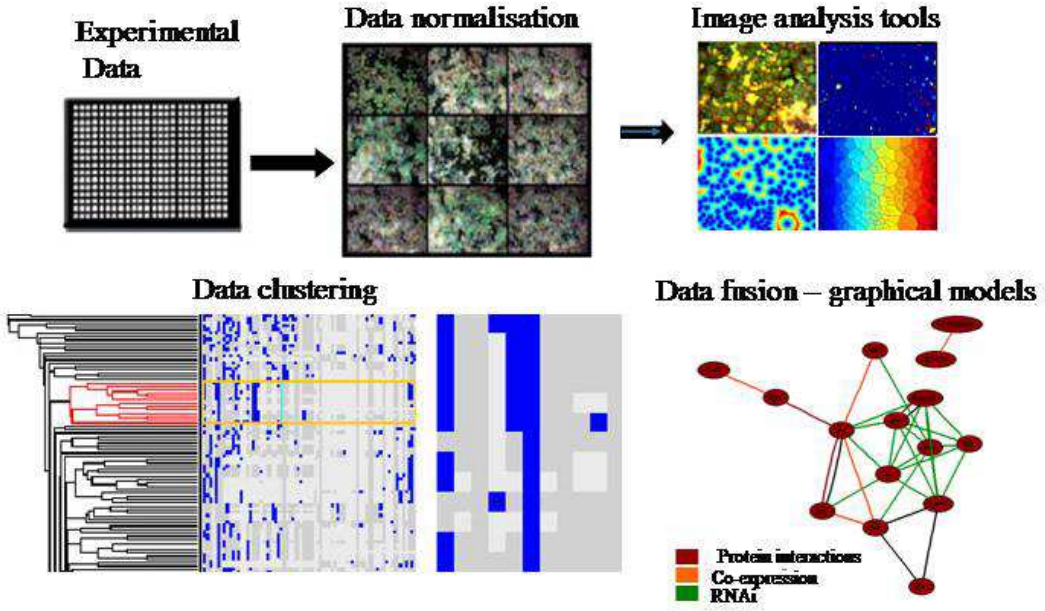


*Fig. 1.5:* **Overview of the paradigm from biological experiments to computer assisted interpretation.** Top left: Experiments. Shown is a plate containing many (384) wells. Each well contains cells treated with a different double-stranded RNA. Top middle: Results. Pictures of the cells in each well (taken using an automated microscope). The raw data is normalised, depending on the experiment. Top right: Image processing and segmentation. The images are processed and individual cells are segmented. Bottom left: Classification. The cells are clustered via a classifier. Bottom right: Biological Interpretation. Genes are grouped together and placed into pathways.

# 2. METHODS REVIEW AND PUBLISHED APPLICATIONS

There are a number of components to analysing an image. Here we consider that the overall aim is object classification and so the problem can be split into image processing, segmentation and classification. There is overlap between these problems, for example it could be argued that segmentation is only meaningful in the context of some classification. Even at the simplest level, segmentation could be considered as classification of an image into foreground and background. Despite the ambiguity, it is usually apparent from context which category a given method falls into, and the separation into image processing, segmentation and classification is a useful way to view the overall problem. Additionally, note that each of these steps is frequently an underconstrained problem. For example, depending on the context, the "correct" classification of objects in an image can be different.

An overview of a number of biological image processing techniques is given in [72]. Although it is possible to develop new filters and wavelets, this is not considered in this thesis, and only well known filters and image processing methods are used. An excellent description of many image processing techniques can be found in Russ's book [59].

## *2.1 Segmentation*

There are many possible definitions of segmentation. The definition used here is a process that defines regions of interest. In this section, I describe segmentation schemes applied to biological problems and explain how they relate to our dataset.

The watershed algorithm is an algorithm that can be used to separate touching objects. A significant number of papers have been written on the use of the watershed algorithm, and in particular on the use of the watershed algorithm for segmenting cells from images [2, 4, 23]. Typically the images are more simple, and the cells more regular in shape than those in the dataset considered in this thesis (see Chapter 1).

There are many different versions of the watershed algorithm, some of which are written as a flooding analogy and some which are written as a drainage analogy. A form of the algorithm is described here, and it is important to note that the assumptions it makes will be valid to varying degrees across the range

of possible biological images. The name comes from the idea of interpreting the brightness of a pixel as a physical height and imagining where water would flow on this landscape. The outline of a binary watershed algorithm is given below. Objects are assumed to be brightest in the centre and diminishing in intensity moving away from the centre. So the centre of an object can be imagined to be a mountain peak. Between two touching objects there will be a trough, where water flowing down each mountain would meet. The places where water would meet are called watershed lines and are used to separate the objects. The classic watershed scheme [59] for separating touching objects is to iteratively erode a thresholded image. In the thresholded image pixels are either on or off. At each step if a separate object disappears then the last position to disappear is denoted an Ultimate Eroded Point (UEP). The iterations continue until every point has been removed. Then starting with the UEPs the image is dilated, setting pixels to on as long as they were on in the original thresholded image, and also as long as turning that pixel on does not cause two previously separate objects to join. This process has the effect of drawing boundaries between separate objects in the thresholded image. The process relies on objects separating due to erosion before they disappear. This works best for convex objects: a single concave object is liable to be split in two by the process. Although our cells are not, in general, convex, this technique is used as an initial method of segmenting cells. An alternative form of this algorithm works by using distance transforms instead of iterative erosion. This approach is faster and is described and implemented in Chapter 3.

There are a number of variations and applications of the watershed algorithm [23], [46], [63]. Watershed algorithms work well for separating individual objects that touch but do not overlap too much, but tend to over-segment clusters of cells and result in irregular boundaries [23].

Hence, for many cell images other methods must be developed and used either in isolation or in tandem with watershed algorithms. Discussed below are examples of alternative algorithms that have been applied to biological images.

A major problem in trying to segment biological images is the vast differences that can occur between different images. Hence, most attempted solutions in this area are very problem specific. The work done usually makes assumptions which are only applicable to the data considered. Segmentation of our images is difficult due to touching cells, clumped cells, variability of cell appearance, and the size of the dataset.

Wu et al. [76] discuss a way of "optimally" segmenting cell images. They define a constructed image based on a pixel labeling as cell or background. This labeling is obtained by first selecting a point inside the cell (say $p_1$) and

defining, as a function of angle, the distance to the edge of the cell. Pixels which are closer to $p_1$ than the edge are defined as cell. They then define a cost function for minimising the mean squared error between the original and the constructed image. This is used to define an optimal constructed image via an iterative process of selection of two parameter vectors. One parameter vector controls the form of the distance function, and the other controls the pixel intensity of pixels once they have been labeled as either cell or background. The final segmentation is obtained by a threshold on the final constructed image. The method is fundamentally based on the assumption that cells are convex regions, so this method is not directly applicable to our data-set.

In many situations, cells form elliptical shapes. In [78] a genetic algorithm is used in order to detect ellipses. An approximate estimate for the size of cells within an image could be obtained by the number of nuclei, and excluding background and clumps of cells from this count. Excluding background and clumps of cells yields the monolayer of cells in the image. The number of nuclei in monolayer is the same as the number of cells. In a fixed monolayer area, the area of each cell varies inversely with the number of cells. Hence in a monolayer the number of nuclei can be used to give a proxy measure for cell size. Cell edges in our images are, usually, denoted by a sharp intensity gradient in the red channel. Hence it would be possible to assign a score (objective function) to closed contours by the gradient in the red channel and the closeness of the area contained within the contour to the approximate cell size. The format for processes could be as follows: (i) Choose a start place (ii) Trace out a contour. The mutable elements of the processes could be the value of coefficients adjusting how far away from a nucleus a contour starts, and the ratio of coefficients dictating whether the contour follows an intensity gradient or distance from the nucleus. It would be possible to implement the above procedure using dynamic programming using manually selected coefficients. A genetic algorithm could give the basis for selecting these coefficients. Additional components could be added to the objective function such as penalising contours which include regions of background. Genetic algorithms can be used to solve optimisation problems. They will not, in general, find an optimal solution.

Campbell et al. [10] aim to identify the types of shellfish larvae present in a sample of water. The images are segmented by binarisation into object and background using a threshold based on the image intensity histogram. Wu et al. [75] describe an iterative thresholding procedure for segmenting cells from noisy images. Two image regions are assumed: cellular regions and background regions. It is assumed that each region should have one pixel intensity and that variation from this intensity is due to independently distributed Gaussian

noise. This assumption does not hold for our images.

There are a number of adaptive threshold selection techniques. The Expectation Maximisation (EM) algorithm (see Appendix A) can be used to model underlying distributions in any given region [72]. Wu et al. [77] tackle cell segmentation across sequences of images. Their approach is two step: first an approximate region about the cell is cutout, referred to as the approximate region, and then the cell is accurately segmented within this region. Part of the benefit of this approach is that it diminishes the problem of varying illuminance across the image. [72] also notes that global thresholds for light microscope images may perform poorly due to variations in imaging conditions across single images, and also between images obtained under different experimental setups. The approximate region is found via a threshold based on local variance. A window is passed over each pixel and the variance of the image within the window is calculated. This gives a double peaked histogram of variances. The peak with the higher variance corresponds to background while the narrower, higher peak corresponds to object pixels. A global threshold is then defined, by Kittler & Illingworth's method [37], on the variance histogram to distinguish object and background. This method defines a larger region as object than Otsu's method [53], and hence has less chance of clipping part of the cell out, which is not desired. On this approximate region another global threshold is used, chosen via Otsu's method. A connected components algorithm is then run. The largest non-background component is taken to be the cell. Holes in the cell region are filled in by identifying small disconnected background regions. The boundary of the cell is smoothed via a closing operation. This method produces good results for uncluttered cells, however, our images contain touching and overlapping cells. This method could be used for accurate determination of the nuclear region in our cell images, but that is not a primary aim.

Given the scope of the problem, the literature is mainly notable for a lack of good cell segmentation solutions. Papers invariably address a specific dataset. Two notable pieces of image analysis software that have been designed for application to cell images, CellProfiler [12, 13] and Calmorph [52, 70, 61], are considered next. They segment cells and then measure features of those cells.

### *2.1.1 CellProfiler*

Cellprofiler is open source, modular software written in Matlab. It collates a number of algorithms to measure cell properties and provides a user interface. The software use a pipeline concept to allow flexibility in the calculations made. The pipeline can be modified to include or exclude certain modules. A typical pipeline consists of illumination correction, object identification, and

measurement phases. Measurements made include calculation of areas, shape, intensity and texture. Some of these measurements are calculations such as area and perimeter, while others are texture descriptors, based on filters such as Zernike polynomials, Haralick and Gabor filters.

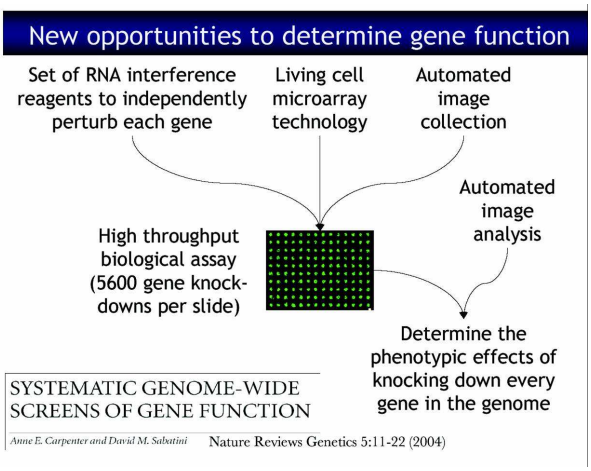


*Fig. 2.1:* **Image analysis shown in the scheme of determining gene function.** Cellprofiler has been designed to carry out the automated image analysis stage. It is this niche that we also seek to occupy

A good showcase of the type of result that can be obtained using CellProfiler is given in [12]. The use of CellProfiler needs to be biologically informed in terms of what parameters are measured. That is to say that the choice of which aspects of an image are measured should be decided by the specific topic being studied. The strength of this approach is that it directly informs biologists of numbers they are interested in. The weakness is that a researcher must know exactly what they are looking for. This precludes clustering cells based on an unexpected variable, such as might occur in a genome wide screen.

Statistical significance is calculated via Z and V factors. By supplying positive and negative examples of some feature, such as large nucleus or normal nucleus, the Z and V factors can be used to determine which of the many calculated measurements are most significant for distinguishing between them. The test assumes that data is taken from a normal distribution. The Z factor is:

$$Z = \frac{(\mu_p - \mu_c) - 3(\sigma_p + \sigma_c)}{\mu_p - \mu_c} \tag{2.1}$$

Where $\mu_p$ & $\sigma_p$ are the mean and standard deviation of the positive distribution, and $\mu_c$ & $\sigma_c$ are the mean and standard deviation of the control distribution. Values of the Z factor range between $-\infty$ and 1. A value of 1 denotes an ideal assay with no overlap of the positive and negative distributions. The V factor is a similar statistical test.

The largest algorithmic contribution of CellProfiler is provided by their cell

segmentation technique [13]. The idea follows from that of an edge stopping function. Hence a cell region should only be expanded from the centre until a significant gradient is reached. Their method is based on propagation methods of distance. Propagation is a distance transform (see chapter 3) version of the watershed algorithm. Their contribution is phrasing the distance as a Riemannian metric, allowing addition of a "regularity parameter" which can be adjusted. A Riemannian metric is not a true metric, but instead a specification of a matrix from which distance can be defined. In 2 dimensions, consider a matrix,

$$\begin{pmatrix} g_{1,1} & g_{1,2} \\ g_{2,1} & g_{2,2} \end{pmatrix}$$

Then define a distance between points a and b by:

$$d(a,b) = \int_a^b \sqrt{\sum_{j=1}^{2}\sum_{i=1}^{2} g_{i,j}\frac{dx^i}{dt}\frac{dx^j}{dt}}dt \tag{2.2}$$

The $x_i$ represent the coordinate system, and t is a dummy variable. Although this can be extended to arbitrary dimensions, we have no need to do that here. Note that the matrix corresponding to the Euclidean metric is the identity matrix. The matrix used to define distance by Jones et al. [13] is

$G = \frac{\nabla g(Im)\nabla g^T(Im)+\lambda I}{1+\lambda}$ Where I represents the identity matrix, Im represents the image and $\lambda$ is a constant termed the regularity parameter. g is a blurring function which is applied to the image. Increasing the regularity parameter, $\lambda$, biases the metric to account less for image gradient and more for Euclidean distance.

A major claim of [12], which I agree with, is that there are no adequate cell image analysis software packages currently available. Commercial software is frequently lacking in flexibility and sometimes in accuracy. A notable point about CellProfiler is that it has been developed as a tool for biologists. The user interface has been designed for use without knowledge of the algorithms. This is an essential design since many of the people who will use software like this are not programmers or mathematicians. Although CellProfiler calculates many statistics about cells, it does not contain a classification procedure. Instead, the classification method is left up to the user.

### *2.1.2 Calmorph*

Calmorph [52], [70], [61] is freely available software written in Java, and similar to CellProfiler in approach. Calmorph has been designed to study budding yeast cells. The segmentation task is significantly easier for yeast as there is

little, if any, overlap of cells. In the Calmorph manual the following recommendations are made "In the images, the cells should not be too close together, and the brightness of the cell wall images should be as uniform as possible", "Incorrect data may result if more than two cells are stuck together" and "is highly probable that incorrect data may result for cells with shapes that differ greatly from an ellipse". Given these restrictions the segmentation task can be very well tackled by a threshold to remove background. The limitations on type of image to be analysed mean Calmorph is not applicable to our images, but it is still excellent for the specific task for which it was designed. Although it invokes the use of thresholds to decide certain factors which could have been decided by a more detailed manner, it is frequently necessary to make such judgement calls in biological problems. It is a good example of how to design a biological analysis tool and is described below for this reason.

Comprehensive analysis of budding yeast mutant traits is an aim of Ohya et al [52]. This is done using triple stained cells: cell surface protein, actin and DNA were each stained with fluorescent dye. Using the Calmorph software five hundred and one parameters were measured. Examples of actin parameters include number of actin patches, location of actin patches, size of actin patches, brightness of actin patches and gravity center of actin region. Cell parameters include, gravity point of cell, distinction of new end and old end, and cell size. DNA parameters include, gravity center of nucleus, brightest point, the most far point from the gravity center, nuclear size, the sum of nuclear brightness and maximum of brightness of nucleus. The idea is to identify and characterise mutants by measuring their cell parameters. As a sample, the following are mentioned in [70].

From these 501 parameters, 254 were judged to be statistically reliable. Statistical reliability was decided via the ShapiroWilk Test for Normality. The value for this test statistic had to be greater than a threshold for the parameter to be accepted as reliable. One hundred and seventy five of the two hundred and fifty four parameters were defined as independent as they had correlation coefficient of less than 0.9. Alternatives would have been to weight the parameters according to independence, or to define a transformation to an orthogonal space, but, in my opinion this approach is fine given the large number of parameters involved. For wildtype (normal) cells, distributions were defined for each of these parameters.

Ohya et al. estimate that the number of deletion strains in the screen having an effect on morphology is given by the number significantly abnormal in at least one parameter. A stringent parameter of $p < 0.0001$ was used to determine significantly abnormal. Similar to alternatives suggested above, alternatives to the use of this threshold can be envisaged, such as a ranking

either by the parameter in which cells are most abnormal or a multivariate score. The method seems appropriate for finding results conservatively. It is likely to be a conservative estimate, since a mutant phenotype might not be extremely different from the normal population in any of the measured parameters, but only those which are picked up using the given p value. A previous study gave 4,718 mutants from a systematically constructed gene deletion collection. Given a population size of 4,718 randomly selected genes, the conservative estimate would be expected to produce $4718 \times 175 \times 0.0001 = 83$ positive results, assuming independence of parameters. The estimate Ohya et al. found is 2,378 putative morphological mutants, of which 544 were from deletions of genes with unknown function. The parameters measured give a basis for characterising the effect of gene deletions. The point worth stressing is that identifying 544 genes with unknown function as being important in morphology is a very useful result, irrespective of the fact that some of the methodology is not as tight it could be.

These results validate the software as a useful tool for identification and description of new mutants in yeast. However, the cells in our images would not be segmented accurately by their methods, as explained above. The quality of the measurements of the parameters in Calmorph is entirely dependent on the accuracy of the segmentation, and so would not give accurate results for our images. Hence Calmorph is noteworthy but not applicable to our images. This software provides the best case scenario for this type of direct feature measurement based analysis.

### *2.1.3 Segmentation summary*

Successful published applications of computational analysis of biological images primarily depend on segmentation of the cells from the images. For some types of images such as those of yeast, segmentation is an easy task compared to the task of segmenting cells in our images. The main problem with the preceding approaches is the lack of a segmentation procedure which is sufficient to calculate cell features in our images accurately.

## *2.2 Classification*

Classification of objects in an image can be defined as assigning a class label to each pixel in the image. In certain circumstances it may not be necessary to label every pixel, but instead to label regions of a certain class, or another similar sub-problem. In this section I describe biological classification schemes and explain how they relate to our problem.

Neural networks have also been used as classifiers of objects in biological

images [51]. In [51], a user clicks on regions of the image denoting fluorescently stained lymphocytes, and regions far from selected regions are selected at random to denote non-lymphocyte regions. A neural network (local linear map (LLM)-type) is then trained using the selected regions. The features used from the selected regions are the eigenvectors obtained from PCA decomposition of the covariance matrix of the image patches. They report 90-95% correct classification of true positives, and 80-87% correct classification of true negatives. We will not use neural networks, but an interesting item from this paper is the method of selection: asking a user to input regions of images in which they are interested. This could form the first stage of a detection, segmentation and recognition system, where the task to be performed is specifically guided by the user's interests.

Campbell et al. [10] aim to identify the types of shellfish larvae present in a sample of water. After segmentation, classification proceeds using rotationally invariant features and then Fisher linear discriminant measures. The method is good, but is facilitated by ease of segmentation for Campbell's images. The applicability of the method is restricted due to an inadequate segmentation procedure for our images.

Automated detection of live cells in a tissue culture is examined in [68]. The main objective of this paper is to distinguish cells from cellular debris which exists in their cultures of the cell line they use. Their algorithm uses 18 features to distinguish cells from other objects. Examples of features are "average depth of minima relative to background" and "maximum peak height between minima". A linear discriminant function is applied using feature measurements to determine whether a detected object is a cell or cellular debris. The direct relevance of this paper to the RNAi project is limited, since the setup is so different. However, it demonstrates the type of technique that can be profitably employed if we could determine features of many different cells. Accurate determination of features of cells requires accurate segmentation of the cells. This is not tackled in [68] since the goal is cell detection rather than cell classification.

In general, the existing literature is specific to a given dataset. The procedures are not generalisable. Hence it seems appropriate to develop our own classification procedure for our dataset. Our aim will be to develop a method which is generalizable, as well as being useful for our dataset. To do this, I looked instead to decision theory, classification and machine vision literature.

### *2.2.1 Generative Models*

Given the difficulties with the methods outlined above, we approach the problem by building a generative model of cells. Once the framework has been

developed this approach has the benefit of being generalisable to any type of image. Generative models, and the development of a novel biological model are described in Chapter 4. So far, there has been less application of generative models, compared to the methods discussed above. Carravick et al. [11] use Latent Dirichlet Allocation [8] to derive a hierarchical model of lung images. Latent Dirichlet Allocation is a generative probabilistic model taking inspiration from latent semantic indexing. Probabilistic latent semantic indexing can be used to classify objects into classes without prior specification of the classes to be used [67]. We aim for this type of result, but initially specify model classes.

Jojic et al. have presented a generative model of an image, based on modeling an image as a collection of patches. We decided to develop a new method based on the work of Jojic et al. [33], fully described in Chapter 4, and apply it to analysis of images of cells in our dataset. Our model is less sensitive to segmentation than measurement-based schemes which rely on accurate segmentation. In the next chapter, a segmentation procedure based on the methods seen in the literature is developed. The segmentation procedure makes use of thresholds, morphological operators and a form of the watershed algorithm. The segmented data will be used as input to the generative model developed in Chapter 4.

# 3. PRE-PROCESSING, CELL DETECTION AND APPROXIMATE SEGMENTATION PROCEDURES

The purpose of this chapter is to define regions of interest in the cell images. Ideally these regions would coincide exactly with cell boundaries. However, the problem is doubly under-constrained. A given image can have many segmentations and a given data value can be caused by many conditions. Insertion of semantic knowledge in the specific situation can help provide a satisfactory solution to each problem. A method to find the cells in the images and to approximately segment them from the image is described. The purpose of this is to extract cell regions so that they can be used as an input to train and test the classifier developed in the next chapter. The stages involved are preprocessing, detection of the approximate cell centre, and then cutting out an approximate cell region based on Euclidean distance to the nucleus. This method was found to give results which were too variable for the range of images in our dataset. Instead we developed a simpler method which operates by cutting out circular regions around the centre of the nucleus of cells.

## *3.1 Preprocessing*

### *3.1.1 Preprocessing: Method*

The images are colour images of size 640x512 pixels. A typical image will contain order of 100 cells. The intensity range of the images varies greatly across the entire set. Part of the intensity difference is due to the nature of the cell, possessing more or less of a staining target (e.g. more actin). However, some of the intensity difference is not due to any property of the cell, but instead due to experimental variability or error, such as the level of staining, a non-uniform washing process and the position in the well. The washing process is designed to remove floating, that is cells which have not adhered to the surface of the plate. However, this process is not uniform, and indeed involved a "banging" component whereby plates would be banged manually on a bench top in order to remove cells which had not adhered to the surface. This was then followed by rinsing off the plate, before fixing the cells and staining them. The staining can be affected by slight variations in the quantity of antibody-dye product added. Ideally we would like to only see the intensity differences

which are due to genuine cell differences. Hence we look at schemes to mitigate the effect of other sources. That is we seek to boost the signal to noise ratio. To do this we will look at normalisation and illumination correction of the data. Illumination correction may be appropriate for images such as ours, since due to experimental procedure there may be consistent illumination variance across the images, independent of the content. Images could be normalised on a per image basis, or on a per plate basis. There is a strong argument against normalising across all images, since the variation between images is such that it is not expected that the average brightness and/or total brightness is the similar for each image. However, since there are 396 images per plate, it is reasonable to expect that plates should have similar average brightness levels. We therefore choose plate based normalisation. Initially we tried these procedures before any other pre-processing. However, there are large regions of the images in which cells have clumped together. It is very difficult to obtain any information about individual cells in these regions, and hence clumps can be considered as noise cluttering the signal. Before normalisation, I therefore removed clumps from the images as described below.

The large swathes of orange in the cell images, see (Figure 3.1) are cell clumps: regions of the plate where many cells are on top of each other and hence give a bright response. The signal in these clumps come from many cells and there is very little information for a trained biologist in these regions. Although clumping itself may be considered to be a phenotype, beyond the existence of clumps, there is not much more information that can be gleaned from clump regions. For this reason clumps will not be excluded from further analysis, although this is an issue which could be re-visited after examining the more information rich regions of the images. Regions of high green and red intensity denote these clumps. Regions above 90% of maximum intensity in the image in both the red and green channels are considered to be part of a clump. A binary mask is created to denote clump region via thresholding at 90% and the use of logical operations. A connected components algorithm is run on the mask in order to label each region of the mask. The area of each connected component is found at the same time. Components with an area less than 100 pixels are removed from the mask.

Next the morphological operation of opening is performed to clean the image. Visually it was found that the following erosion and dilation structuring elements worked well. The erosion structuring element was:

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

The dilation structuring element used was:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Erosion then dilation was performed iteratively eight times, using the above structuring elements to create a final mask. This mask is inverted and elementwise multiplied by the original image matrix to yield the original minus clump and background regions. Although closing involves erosion and dilation using the same structuring element, it was found that the above procedure gave better results, in terms of removing non-nuclei regions while retaining regions judged (by eye) to be real nuclei.

Next the mask is further tidied up using a size filter. Only regions of foreground larger than 20 pixels in area are maintained as foreground regions. The sums are done per colour channel. This is a slow process, but since it is only performed once it is an acceptable computational cost.

The average pixel intensity varies from plate to plate. Of course, we would like the pixel intensity to vary with the level of a given protein. However, since the allocation of phenotypes to plates is random, we would expect the average pixel intensity of a large number of images to be approximately equal. Hence, I implemented inter-plate pixel normalisation. This was done using a scaling factor on the remaining region, which will be referred to as the monolayer. The scaling was performed according to equation 3.1.

$$\mathbf{I_n} = \mathbf{I_o} * (P_{max}/P_c) \tag{3.1}$$

Where $\mathbf{I_n}, \mathbf{I_o}, P_{max}, \& P_c$ are the normalised monolayer intensities (a vector of pixels), the original monolayer intensities (a vector of pixels), the largest sum of monolayer intensities (summed over all images) over a plate (a scalar value), and the sum of monolayer intensities (summed over all images) of the plate containing the image being normalised (a scalar value). The effect of this is to normalise the average pixel intensity of the monolayer in each plate up to the largest in the set. This process is designed to remove variation across plates (we argue that the average pixel intensity across a large number of images is a constant) and transforms the data to reflect this. Normalising the plates up in this manner is effective for this purpose, as can be seen in Figure 3.2. Each colour channel is considered independent for the purposes of normalisation since they were obtained one at a time in the experimental stage. Hence the above process is repeated per colour channel.

### 3.1.2 Preprocessing: Results

To display the results of the preprocessing, an image which has a large amount of cell clumping is shown in Figure 3.1 The original image, which contains many clumps, is shown on the left, together with the same image but with clumps and background removed shown on the right.

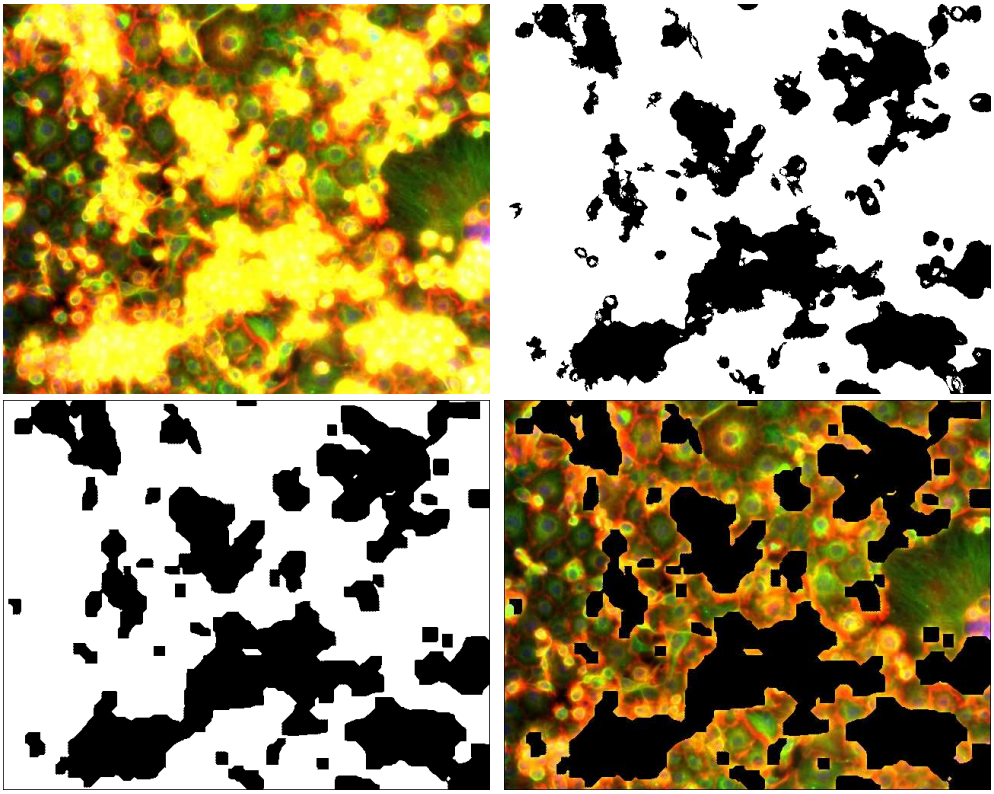

*Fig. 3.1:* **Preprocessing stages:** Top left: Original image. The orange/yellow regions are clumps of cells. Top right: Clump and background mask. Bottom left: Mask after 8 iterations of erosion and dilation (erosion then dilation, erosion then dilation, etc). Bottom right: Mask placed over the image to include only the regions in which we are interested.

The results of the inter-plate normalisation can be seen in 3.2.

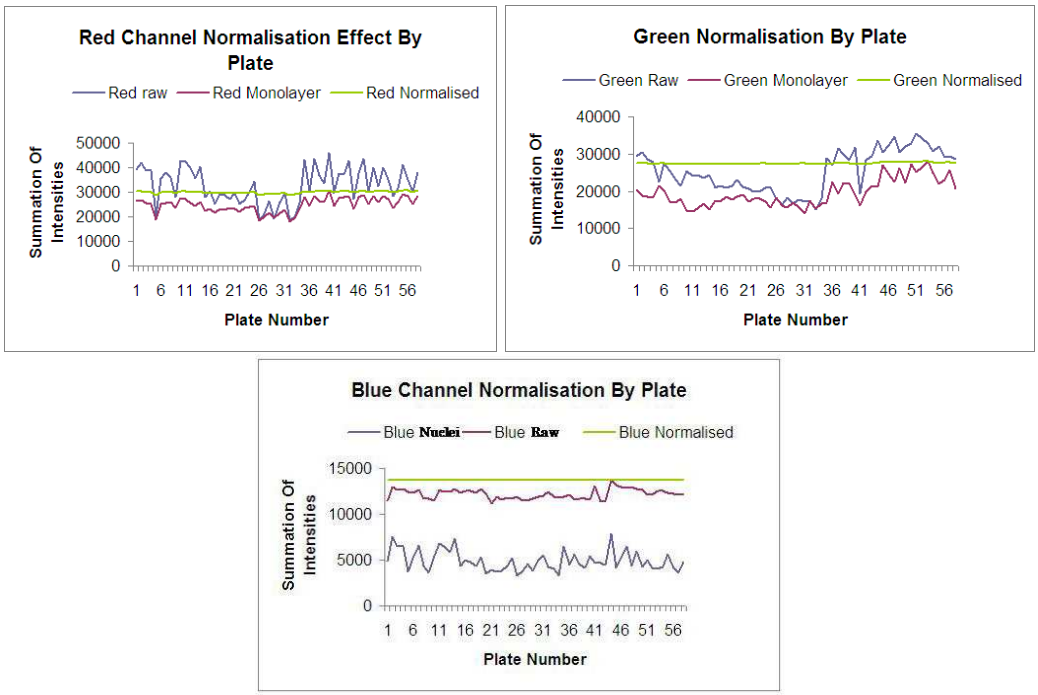


*Fig. 3.2:* **Normalisation** The raw, monolayer and normalised monolayer intensities are shown. Note the resultant normalised line across plates which is close to flat.

### 3.1.3 Preprocessing: Discussion

The preprocessing effectively removes clumps of cells and background regions from consideration, and then normalises the monolayer between plates. However, the clump removal stage also removes some mitotic cells which are not necessarily part of a clump of cells. This could be avoided by only removing large clumps of cells from the image. However, in the first instance we aim to examine (via the method developed in the next chapter) the shape of cells in interphase, so the exclusion of cells in mitosis is useful. Preprocessing potentially removes or flattens interesting parts of the data. For example the performing clump removal does not work equally well on each image. In some images the automated clump removal I have designed removes cells from the monolayer, while in others parts of the clump region remains. This is due to the method of choosing the threshold, which does not work equally well for all images. Otsu's method [53], was also tried, in order to set the threshold individually for each image. Otsu's method works by choosing a threshold to separate pixel intensities into two bins which minimise the intra-class variance of each bin. However, this did not in fact work as well as a fixed threshold, yielding similar problems to described above. The method intrinsically assumes a certain percentage of pixels are foreground and a certain percentage are background. However, the percentage of clump pixels varies from image to image, and the clump percentage of the image is sometimes significantly less than or more than 50 percent, ranging between 0 and 95 percent. This presents a significant variation from the assumptions made by Otsu's method.

An additional preprocessing stage that needs to be implemented in future is illumination correction, since the images tend to be dimmer around the edge. A module for performing this is found in CellProfiler [12]. The effect of missing out a global illumination correction step is minimised by normalisation of each region of interest. The greatest illuminance irregularities occur around the edges of the wells, so another way of reducing the problem of illumination correction is to remove a border around the edge of the well. However, this is not done in order to maximise the number of cells available for use in the classification model described in the next Chapter.

## 3.2 Cell Detection

The next task is to find the cells in the image. The example figures below refer to processing done on the image of cells shown in the top left of Figure 3.3. This task is simplified by the fact that DNA, which is confined to the nucleus, is stained blue, and every cell has DNA. The nucleus is the biological centre of a cell, and in most cases is close to the geometric centre. Bright regions in

the blue channel, see top right of Figure 3.3, denote the nuclei. Hence, nuclear detection can be performed using an intensity threshold.

### *3.2.1 Nuclei detection: Method*

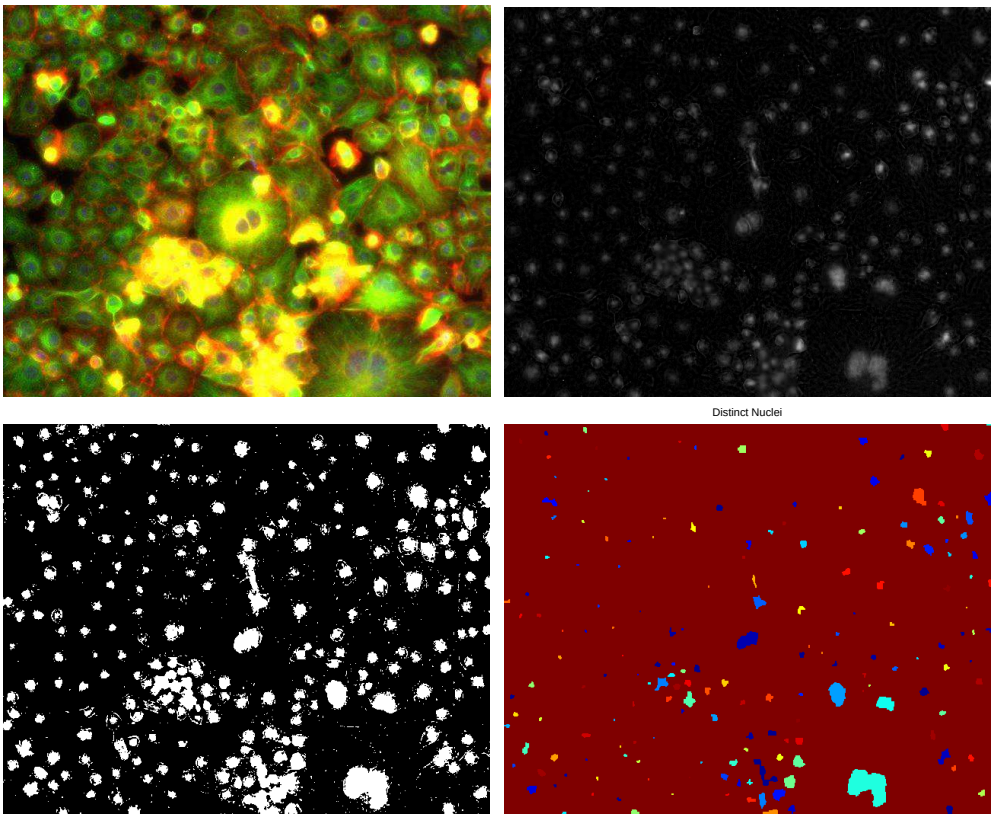


*Fig. 3.3:* **Nuclei Detection Stages:** Top left: Original. Top right: The blue channel, showing cell nuclei. Bottom left: Thresholded blue channel. Bottom right: Erosion and connected components algorithm used to label distinct nuclei.

The blue channel of these images consists of pixels belonging to nuclei and pixels not belonging to a nuclei. Pixels which belong to nuclei are considered foreground pixels, and all others to be background pixels. This is done by thresholding the image by intensity. A threshold is chosen for the image using Otsu's method [53]. Distinct foreground regions are then labeled via a connected components algorithm [59].

Very small regions of blue need to be excluded, since they are likely to be caused by noise, as shown in the bottom left of Figure 3.3. These small "nuclei" can be removed via erosion. The structuring element that was found to work well was:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Most nuclei (the exact proportion is image dependent) are not "deleted" by

this process. Although the process reduces the size of the remaining nuclei, the purpose here is nuclear detection rather than accurate determination of the nuclear region. The overall process is successful (for our purposes) as long as we can identify some nuclei in the monolayer. More nuclei is a bonus, but we would like to avoid false positive detection of nuclei. Visible in the bottom right of the images is a large nuclear region which is due to two nascent nuclei that have not yet completed mitosis. This is especially clear in the thresholded image in the bottom left image of Figure 3.3.

### *3.2.2 Nuclei detection: Results*

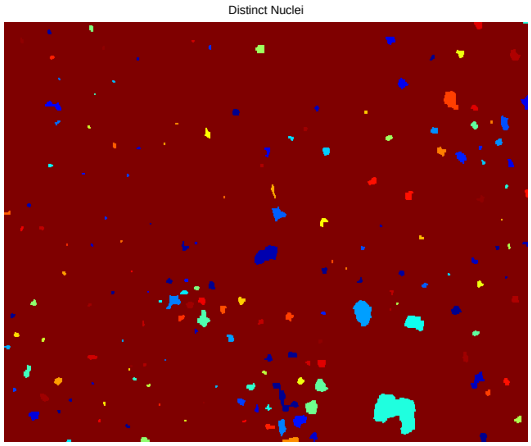


*Fig. 3.4:* **Nuclei Detection Results** Distinct nuclei are shown. For visual clarity each nucleus is given a different colour (However, note some of the shades appear similar)

### *3.2.3 Nuclei detection: Discussion*

The result is a good basis for finding the centres of mass of the nuclei, by equal weighting of the nuclear pixels, and hence a good approximation for the centres of the cells in the image. The detected centres of mass of nuclei tally well with what would be defined by a biologist as centres of mass of nuclei. The nuclei are not rounded like real nuclei. This is due to the small number of pixels which make up the nuclei, combined with the effect of binarising the image. The next stage is to define cell regions based on the detected nuclei.

## *3.3 Cell Extraction*

I will look at two methods of cell extraction: Voronoi partition of the image based on cell nuclei, and cutting out circular regions around cell nuclei. In each case the pre-processing and nuclei detection stages are as described above.

### 3.3.1 Cell Extraction by Voronoi Partition: Method

Cells are extracted by assigning points in the image plane as belonging to one nucleus or another, based on distance. This is done using distance transforms.

A distance transform is a "distance map" showing the distance of points in a plane to some specified object. The purpose of calculating the distance transform is to find a partition of the image plane which will provide approximate cell regions. There are fast methods of calculating the distance map which avoid the slow process of finding the distance of every pixel in the plane to every detected nucleus [15, 17, 19, 55]. Figure 3.5 shows a distance transform applied to a typical image of cells, based on the centres of detected nuclei.

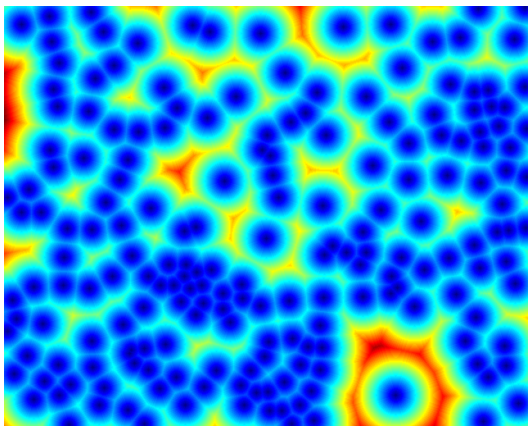

*Fig. 3.5:* **Distance transform displayed as a heat map.** Hotter colours represent points in the plane far from any identified nucleus centre

Points closer to one nucleus than any other are taken as belonging to the cell which contains that nucleus. This is called a Voronoi partition and provides the basis for cutting out cells from the image in the first instance. Figure 3.6 shows the image plane from Figures 3.3 & 3.5 partitioned by closest nucleus. Individual cell regions are then placed into boxes of size 100 x 100 pixels. This is considerably larger than the average size of a cell (approximated to be 30 pixels diameter for a typical wildtype cell), since there is large variability across the entire set.

The size of the nuclei found in the cell detection stage is recorded. Cell regions belonging to large "nuclei" are removed as they may be nuclei undergoing division or staining artifacts, such as can be seen in Figure 3.7. The choice about the largest acceptable nuclear size was decided in a heuristic manner, dependent on experience. Here a total pixel area of below 40 pixels was allowed. However, this stage is performed *after* the cell regions have been calculated via the Voronoi partition, so that the corresponding cell region is also removed. This avoids falsely enlarging neighbouring cell regions.

The quality of the cell region extraction is qualitatively assessed by com-

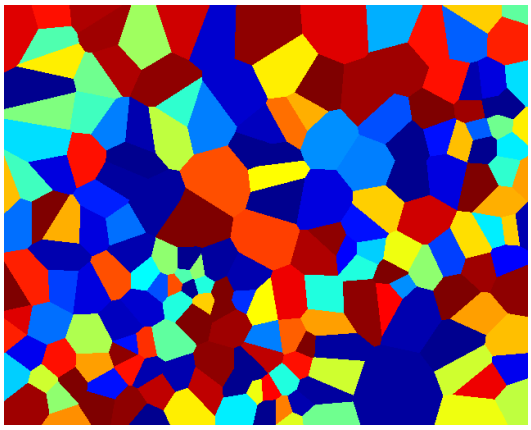

*Fig. 3.6:* **Voronoi partition of the plane by nearest nucleus.** These regions denote putative cell regions. Each cell is given a different colour.

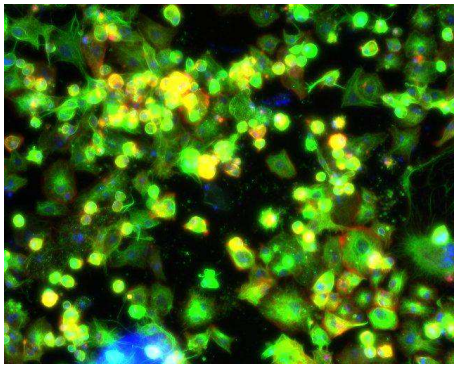

*Fig. 3.7:* **Example of a staining artifact:** a blue staining artifact is visible in the bottom left.

paring with hand segmented cells.

### *Polar Coordinates*

The cutout cells are converted from Cartesian coordinates into polar coordinates. Cells are naturally organised around the "central" point of their nucleus, and so polar coordinates is a natural way of representing them. Bilinear interpolation is used to define pixel values in the new coordinate system. The size of 100 x 100 pixels is retained for the polar coordinate system. The origin in the new coordinate system is chosen to be the centre of the nucleus.

### 3.3.2 Cell Extraction by Voronoi Partition: Results

Figure 3.8 shows examples of individual cell regions as defined by Figure 3.6

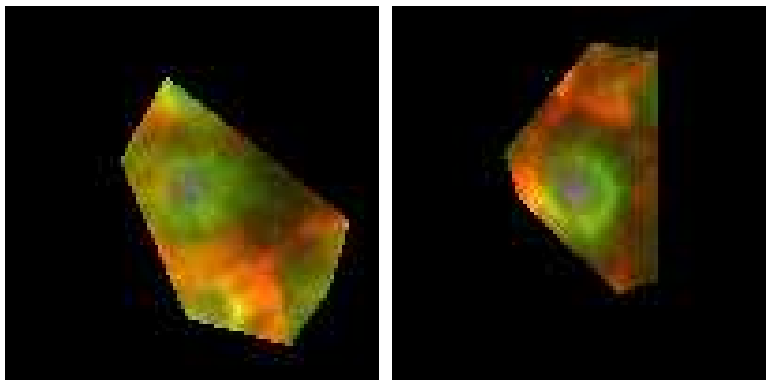

*Fig. 3.8:* **Example cell cutouts via a Voronoi partition**

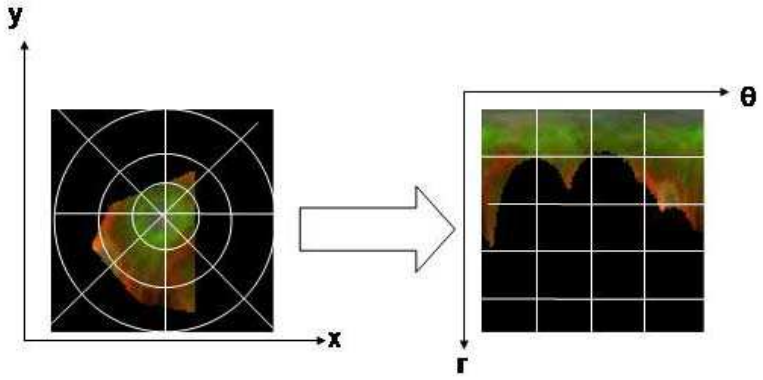


*Fig. 3.9:* **Polar Coordinates.** Conversion of a cell cutout from cartesian coordinates on the left to polar coordinates on the right

### 3.3.3 Cell Extraction by Voronoi Partition: Discussion

The process of cell extraction from image to partition of the image plane is shown in Figure 3.10. The cell regions cut out are different from the regions that would be cutout by a trained biologist. However, the aim here is not accurate segmentation of the cells, but rather to define regions of interest that provide training samples for the model described in the next chapter. The Voronoi regions necessarily contain parts of the cell membrane, which is vital when studying cell morphology, since the cell edge is one of the main factors that defines cell morphology.

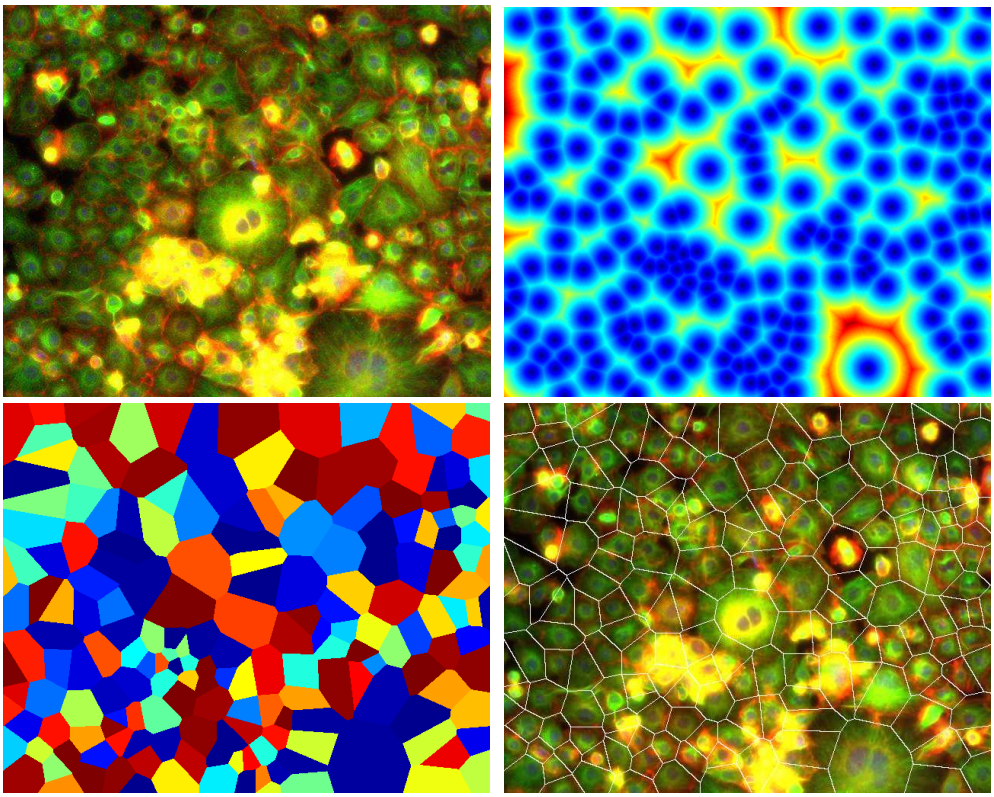

*Fig. 3.10:* **Voronoi Cutout Summary** Top left: Original image. Top right: Distance transform displayed as a heat map. Hotter colours represent points in the plane far from any identified nucleus centre. Bottom left: Voronoi partition of the plane by nearest nucleus. These regions denote putative cell regions. Each cell is given a different colour. Bottom right: Borders from Voronoi partition shown overlaid on the original image

Initially the angular nature of the regions which inexactly match the cells would appear to be the largest problem with the method. However, recalling the purpose of the segmentation, the unpredictable nature of the cutout region is in fact a larger problem. Due to the nature of the cells, Voronoi partitioning performs variably depending on the specifics of the image. This variation is not, in general, a direct function of the cell morphology, which is the property we are analysing. In effect this introduces an additional source of arbitrary noise into our data. Since the segmentation is intended to provide training data for model building we would like to remove any sources of arbitrary noise. Hence,

I next tried a simpler method of cell region extraction that is more predictable.

### *3.3.4 Cell Extraction by fixed regions: Method*

As in the case of Voronoi cutouts, the centre of mass of cell nuclei 3.2.1, provide the seed points for selecting cell regions. For each detected nucleus a 42x42 square is defined with its centre at the centre of mass of the detected nucleus. This is smaller than the above 100x100 containing box, since now I do not aim to guarantee that the entire cell is in the box, but instead that a representative portion of the cell monolayer centred on a nucleus is in the box.

#### *Polar Coordinates*

As above the cutout cell regions are converted from Cartesian coordinates into polar coordinates. Cells have an intrinsic radial symmetry to them, and so this is a natural way of representing them. Bilinear interpolation is used to define pixel values in the new coordinate system. The size of 40 x 40 pixels is retained for the polar coordinate system. The origin in the new coordinate system is chosen to be the centre of the nucleus.

### 3.3.5 Cell Extraction by fixed regions: Results

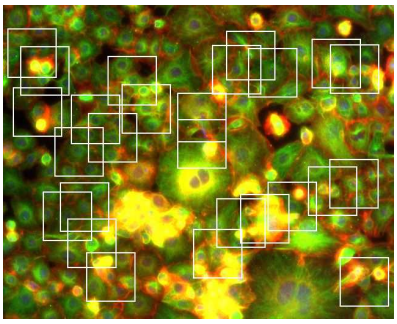

*Fig. 3.11:* **Square Cutouts** A sample of the square cutouts is shown on an example image. Not all square regions are shown for clarity.

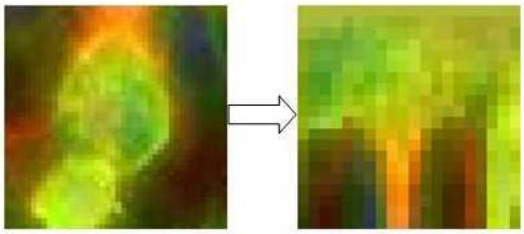

*Fig. 3.12:* **Transformation Of Square Cutout Into Polar Coordinates** An individual square cutout on the left converted into polar coordinates on the right.

### 3.3.6 Cell Extraction by fixed regions: Discussion

The process of selecting square regions around nuclei and then defining a circle of fixed size around them is a simple approach, but one which works well for our purposes. It shows insensitivity to vagaries in the images. For our purposes the cell regions extracted are sufficient in the first instance, since the aim is to provide training data for model learning, rather than exact segmentation.

## 3.4 Discussion

The method developed in this chapter has largely been presented as a means to an end. That is as a means to obtaining approximate training and test data for the generative models developed in the next chapter. There are numerous areas in which the results produced by the methods in this chapter may interfere with further analysis. These are discussed in this section.

In the preprocessing section of this chapter, certain regions of the images are removed. In particular cell clumps are removed. The size of the dataset

is very large, as discussed in Chapter 1, and hence an automated process for removing the cell clumps must be devised. The method used to detect cell clumps was searching for regions of very high pixel intensity. A clump removal algorithm could either remove too much of the image, removing portions of the image which are not clumps of cells, or alternatively remove too little of the image, and leave regions of clump in the processed version of the image.

In the case where too much of the image is removed, this causes two problems. The first is that legitimate regions of the cell monolayer are removed, which reduces the number of cells available. This in itself has two effects. In the instance where the phenotype of the image is being inferred, reducing the number of cells available with which to make this judgment would reduce the accuracy of determining the phenotype in the image. Alternatively we may be using the cells in the image to learn the parameters of a model of that phenotype. Reducing the number of cells available may (depending on the model) reduce the efficacy of the learnt model in describing that phenotype. The specifics of how much data is required will depend on the model, however, for the models that we develop, as much training data as possible is desirable. In general more training data is considered beneficial when training models. A second effect of removing some cells is that it may alter the profile of the detected cells in the image. That is, the cells which are removed by an overly stringent clump removal algorithm may all share characteristics not shown in the remaining cells. For example this could be seen in an image in which there are two types of cell present; wildtype cells (seen in those cells which have not had their mRNA sufficiently depleted to show a phenotype) and cells displaying the phenotype associated with knockdown of the mRNA (seen in those cells in which the targeted gene has been effectively knocked down). This will hinder the ability to recognise any phenotype present in the image.

Conversely, allowing too much of the clump regions to remain brings with it problems of its own. Any clump regions in the images will be detected as cell regions, since they have underlying nuclei. This will yield cell regions being taken forward for analysis which are not recognisable as cells. If these regions were used as training data when learning a model of a phenotype then this will contaminate the result, giving a hybrid model of the phenotype plus clump region.

It is desirable to avoid each scenario, and to only remove those regions of the images that would be defined as clumps of cell by eye. However, across the image dataset both problems are encountered. For some images regions of clump remain, while in others legitimate cell monolayer regions have been removed. A difficulty common to all of the preprocessing methods is encountered at this point. That is that actually estimating the errors made, and

taking steps to reduce any errors. Errors are defined by reference to what a human would if removing the clump regions by hand. Due to the size of the dataset it is infeasible to find this for every image. Methods of reducing any error could be explored, such as using adaptive thresholding algorithms. These algorithms could then be evaluated against a subset of the whole set which had been "correctly" marked by hand. Carrying out such an analysis would be useful, and may reduce downstream effects associated with clump removal. However, thresholding algorithms are not the focus of this thesis, and the process of clump removal was performed as a sensible step to reduce the effect of cell clumps. It is noted that more work could be done to reliably remove clump regions, but it is not further explored here.

## *3.5 Chapter Conclusion*

Adequate definition, for our purposes, of regions of interest has been achieved. The segmentation is far from ideal, as compared with cell regions segmented by a human expert. However, from experience, the regions chosen sufficiently capture cell regions so that the regions of interest of cells of different phenotypes can usually be distinguished by eye.

Many possible improvements and alternatives to the segmentation procedure could be explored, including altering specific values used in the preprocessing, such as the size and shape of structuring elements, or the level of the thresholds. This part of the process is specific to the images used, and may be explored more fully at a later date if it is found that the automated segmentation described in this chapter is limiting the accuracy of the classification of cell images. The extent to which the automated segmentation is problematic can be assessed by comparing classification using automatically obtained cell regions as against classification using hand segmented cells. In the next chapter the framework for the classification procedure is described.

# 4. BLUEPRINT: A NOVEL GENERATIVE MODEL FOR AUTOMATED ANALYSIS OF CELL FORM

This chapter describes the development of a generative model that will be used to classify cells. The purpose of the model is to represent an image class, and to be able to compare the similarity of images in terms of the statistics of local image variation. The model we develop uses unsupervised learning, avoiding the need for explicit feature extraction. This is an advantage over other methods which presuppose what they are looking for. The model was inspired initially by epitomes [33]. Although epitomes were first tried, in order to place the model in context and aid in explanation, a thought progression from simpler models is presented. This chapter initially defines generative models, and defines patch models. Next a systematic progression of models from simply fitting a Gaussian to pixel data up to the epitome model is described. At each stage of model development I show specifically how the parameters of each model can be learnt. This thought process is extended and make some specific adaptations to define the blueprint model which operates by representing any image as an ensemble of patches.

## *4.1 Generative Models*

A generative model of some data $\mathbf{x}$ is:

$$\mathbf{x} = \mathbf{m}(\mathbf{y}) + \xi \tag{4.1}$$

The data is modeled as consisting of a deterministic part, $\mathbf{m}(\mathbf{y})$, where $\mathbf{y}$ is a vector of variables, plus a stochastic component, $\xi$.

For example a "Mixture of Gaussians" model of pixel intensity of an image is a generative model of an image. Each pixel in the image is modeled as coming from one of the Gaussians in the mixture. The Gaussians can be learnt using a learning algorithm such as the Expectation Maximisation algorithm [21] (see Appendix A).

If the model is written probabilistically (see Figure 4.1), then the likelihood of a data vector being generated by the model can be found.

For example, given n different models of cells, $\mathcal{C}_1$, $\mathcal{C}_2$ ... $\mathcal{C}_n$, we can find the likelihood, $P(x|\mathcal{C}_i)$, of each model for a given cell, $x$. Then by using Bayes'

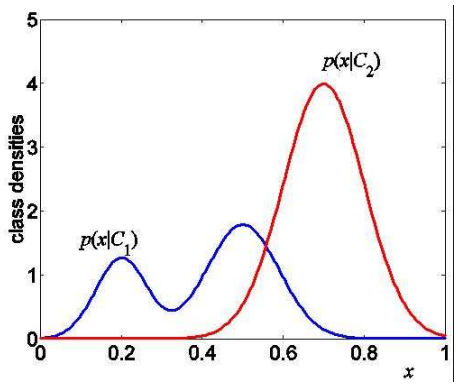


*Fig. 4.1:* **Generative Models** Graph from two generative models showing the probability of a variable, x

rule, equation 4.2 gives a probability that an image belongs to class j.

$$P(\mathcal{C}_j|\mathbf{x}) = \frac{P(\mathbf{x}|\mathcal{C}_j)P(\mathcal{C}_j)}{\sum_{k=1}^{k=n} P(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)} \tag{4.2}$$

In the next section we consider generative models of images.

## *4.2 Patch Based Models*

An image patch is any connected region of an image, but here all patches will be square regions. A patch library is a collection of image patches. A patch library can be considered as a generative model of images as follows. Given a patch library we model an image by first tessellating it into patches, as shown in Figure 4.2 b & Figure 4.2 c. Then each patch is formed by directly copying a patch from the patch library. In general the patch from the patch library will not fit the image patch exactly, and so some stochastic, or error, term must be assumed.

Figures 4.2, 4.3 and 4.4 collectively demonstrate the idea. In 4.2a.) a pictorial representation of a patch library is shown. The deterministic part of the generative model is that each image patch is copied from a patch in the patch library. The discrepancy between any image patch and the best fitting library patch is described as an "error term" and is modeled by the stochastic term in equation 4.1. In Figures 4.2b.)&c.) test images are shown divided up into patches. Each of these patches will be modeled by a patch from the patch library. Figure 4.3 shows a single patch from test image 1 being modeled by a patch library. The patch library models this patch well, so only a small error term must be assumed by the model in order to match the patch in the test image. Figure 4.4 shows a single patch from test image 2 being modeled by a patch library. In this case the patch library contains no good matches for the chosen patch, hence a large error term must be assumed to allow the patch

library to model the test image patch.

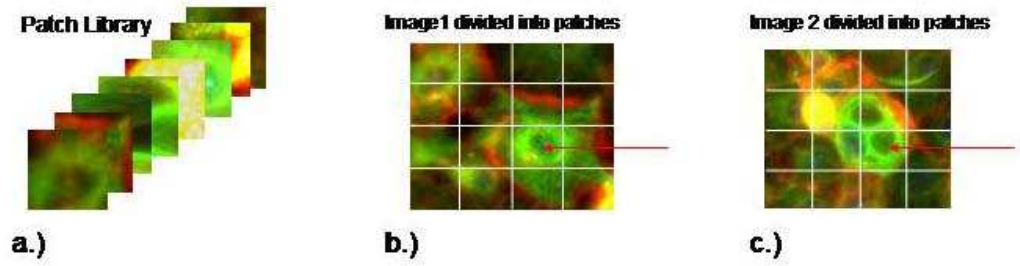


*Fig. 4.2:* **Patch Library Model** a.) The model: a patch library. b.) Test image 1 divided up into patches. The red arrow points to the patch modeled in Figure 4.3. c.) Test image 2 divided up into patches. The red arrow points to the patch modeled in Figure 4.4

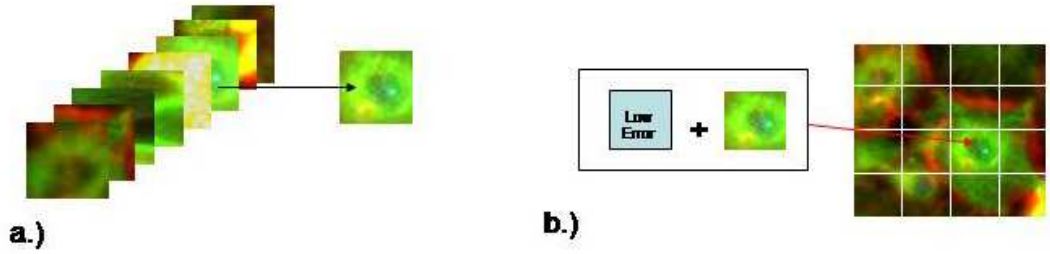


*Fig. 4.3:* **Using a patch library model to describe an image** a.) The best patch is extracted from the patch library in order to model a patch in test image 1. b.) A small error term is added to the patch in order to model a patch in test image 1.

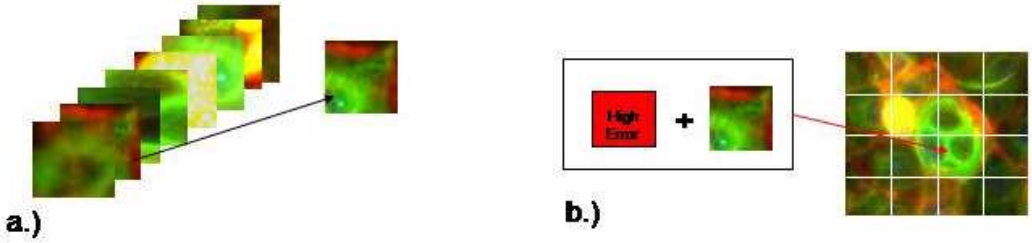


*Fig. 4.4:* **Using a patch library model to describe an image** a.) The best patch is extracted from the patch library in order to model a patch in test image 2. b.) A large error term is added to the patch in order to model a patch in test image 2.

The relative size of the error terms is the interesting part for our purposes. The error term measures how well the best matching patch in the patch library models a given patch in the image. The cumulative error that must be assumed in order to model the whole image, summed over all image patches, can measure how well the patch library models the image. An alternative, developed here, is to define probabilities of image patches based on the degree of error.

The major limitation of a patch based library, as described, is that how well an image is fitted by a patch library is highly dependent on the tessellation of the image. For example, consider an image region which is exactly fitted by a patch model under a given tessellation, see Figure 4.5. Now consider what happens if we translate the tessellation slightly. In the second case the image region is badly fitted by the given patch library model. It is desirable to remove this dependence on tessellation choice, which is arbitrary, and such a model is built up next.

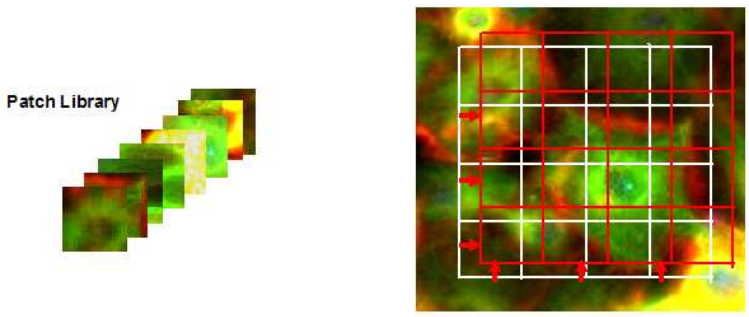


*Fig. 4.5:* **Patch library dependence on image tessellation** Left: Patch library model. Right: An image region to be modeled by the model. Two tessellations are shown, one shown in white, the other in red. The red one is identical to the white one except it is translated up and to the right. Under the white tessellation, the model gives an exact match for the image region. Under the red tessellation, the patch library models the image poorly.

### *4.2.1 Continuous image representation of a patch model*

If all the patches in the patch library are placed next to each other to form a kind of image, then this results in many more patches being available for selection, and reduces the problem of selection of image tessellation discussed above (see Figure 4.5). This image representation of a patch library is shown in Figure 4.6. The purpose of this representation is to reduce the effect of the choice of image tessellation on how well a given model represents that image. This is desirable since image tessellation is an arbitrary choice which is not informative about the image. The idea is that a slight shift in image tessellation can be accounted for by a slight shift in the region of the model

used to describe the image, as shown in Figure 4.6.

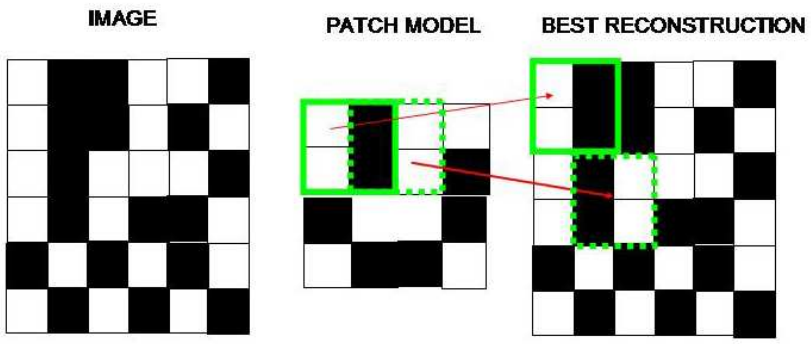


*Fig. 4.6:* **Continuous patch library model**: Shown are two selections of 2x2 patches used to model patches in the image. In a naive patch library, two separate patches would be required to do this.

## *4.3 Building patch based models from Gaussian models*

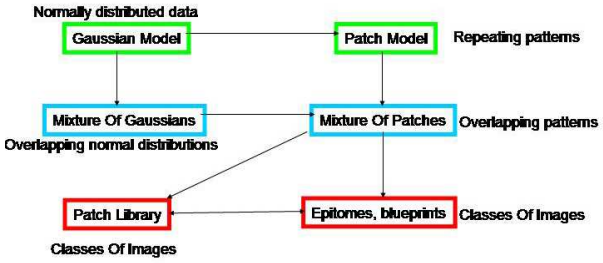


*Fig. 4.7:* **Hierarchy of models**: How different models relate to each other and mixture of Gaussians models. Next to each model is an example of what type of data it is suited to modeling

In this section we build up models of starting with a Gaussian model of data. From this it is possible to either build a mixture of Gaussians model, or create a patch model which consists of a number of Gaussians, but in fixed spatial relation to one another. The single patch model is a single multi-dimensional Gaussian fitted to the data. From a mixture of Gaussians model or a patch based model we can move to a mixture of patches model. A mixture of patches model can be interpreted as a patch library. The mixture of patches model is a mixture model of multi-dimensional Gaussians. (Note that the probability distributions need not be normal distributions, but they are assumed to be normal distributions for clarity of explanation. The model progression described could be written without reference to any particular form of probability distribution, but since alternate distributions are not explored in this thesis, such a level of generality is not deemed necessary).

In all of the following models, each "position" in a model contains two parameters; a mean and a standard deviation. The models are intended to be "learnt" using training data. For a given model form, the learning process adjusts the parameters of the model to increase the probability of the training data being generated by that model. The learning process (use of the Expectation Maximisation algorithm) is described for each possible model. This is added for clarity so that the process for learning the more complex models is apparent by reference to the process for the simpler models. Specifically, we appreciate it is unnecessary to use the Expectation Algorithm to learn the parameters for a single Gaussian model, but we do so to demonstrate the learning process, and help explain the hidden variables in the models (mappings from the model to the image).

For each model explained, an example of how well the model can be used to reconstruct a target image is given. Image reconstruction is not the overall purpose of the models, but the image reconstruction figures are supposed to give a visual understanding of what kind of data the different models are good at describing.

### *4.3.1 Mappings from the model to the image*

All the models described below use mappings from the model to the image. The (set of) mappings from model coordinates to image coordinates are transformations from one set of coordinates to another, as shown in Figure 4.8. In this problem setup, the hidden variables, referred to in the description of the EM algorithm in Appendix A, are the mappings. The purpose of the mappings are to determine which sections of the model are being used to describe a given piece of image data. The mappings are exhaustive, so that for every position in the model there is a mapping from the model to each image patch (subject to the defined image tessellation).

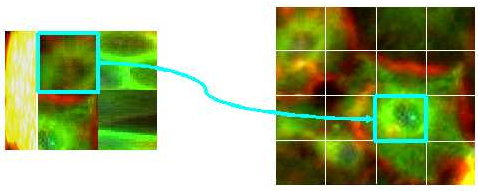

*Fig. 4.8:* **Mappings from model to data:** A mapping of a patch in the model, on the left, to a patch in an image on the right.

### 4.3.2 Gaussian model of data

The simplest Gaussian patch model is when the patch size is 1x1. That is, each position in the model is independent of its neighbours. Within this the most simple model would be one consisting of only one probability distribution. The best distribution to model a given image would then be one which has a mean equal to that of the mean pixel intensity, and a variance equal to that of the variance of pixel intensity. This is equivalent to fitting a normal distribution to any set of data, and is shown in Figure 4.9.

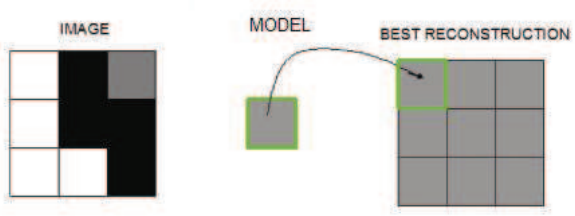


*Fig. 4.9:* **Simplest possible patch model:** A Gaussian model of pixel intensities. On the left is the image from which the epitome is trained. The model mean is shown in the middle. On the right is the best reconstruction of the original image that this model could produce.

Although the model can be found directly via a maximum likelihood calculation, I demonstrate how to learn it using the Expectation Maximisation algorithm. It is important to note that this could be done using a maximum likelihood calculation which would be more straightforward in this case. In particular the description of the mean and standard deviation of the normal distribution as a function of the mapping is unnecessary in this instance since there is only one mapping. However, for the models we will define later there are more mappings, and the values of the mean and standard deviation are functions of the mappings. For reasons of continuity of thought the mean and standard deviation are described as functions of the single mapping. The probabilistic generative model is then as follows:

$$P(a_r|m,\mathbf{e}) = N(a_r|\mu(m)\sigma(m)) \tag{4.3}$$

$$P(A|m,\mathbf{e}) = \prod_r P(a_r|m,\mathbf{e}) \tag{4.4}$$

where $a_r$ is the pixel intensity of the rth pixel in the image $A$, and N(x;y,z) is the probability of observation x coming from a normal distribution with mean y and variance z. The patch model, $\mathbf{e}$ is characterised by the parameter vector $\Theta$, which consists of $\mu$ and $\sigma$. Here, $\mu$ and $\sigma$ are the mean and variance of the Gaussian. The mappings, M, specify which part of the model is being used.

Hence, the mean and variance are functions of the mapping. In this instance, there is only one component to the model, so there is only one mapping. The EM update equations are as follows:

E-Step:

$$P(m|a_r, \mathbf{e}) = P(a_r|m, \mathbf{e})\frac{P(m)}{P(a_r)} \tag{4.5}$$

where $P(a_r)$ is calculated by marginalising over all mappings:

$$P(a_r) = \sum_{m \in M} P(a_r|m, e)P(m) \tag{4.6}$$

However, there is only one mapping, so $P(m|a_r, \mathbf{e}) = 1$. It should also be noted that $P(m) = 1$, and that these terms have been included in the equations for reasons of continuity with the models that are built up later in the section.

The M-Step is a weighted maximum likelihood calculation:

$$\hat{\mu} = \frac{\sum_r \sum_{m \in M} P(m|a_r, \mathbf{e})a_r}{\sum_r \sum_{m \in M} P(m|a_r, \mathbf{e})} \tag{4.7}$$

$$\hat{\sigma} = \frac{\sum_r \sum_{m \in M} P(m|a_r, \mathbf{e})(a_r - \mu(m))^2}{\sum_r \sum_{m \in M} P(m|a_r, \mathbf{e})} \tag{4.8}$$

Substituting $P(m|a_r, \mathbf{e}) = 1$ into the above shows the mean and the variance of the data are the maximum likelihood estimates of the parameters for the data. We note again that a maximum likelihood estimate could have be used instead of using the EM algorithm. The details of the EM process have been included to help gain familiarity with the mappings, and the model syntax that we are using.

### *4.3.3 Mixture of Gaussians model of data*

The next simplest model would be one consisting of two probability distributions, again with patch size one in each case. Such an model could be good at modeling a two tone picture, whereby each position in the epitome models one of the tones. This model is a standard mixture of Gaussians model. This is shown in Figure 4.10. As can be seen the model can regenerate the original image well. Only the top right pixel is not faithfully reproduced.

Using a patch size of one, and a model composed of n distributions translates directly into modeling a picture (equivalently a set of data) by n different normal distributions, and is exactly the same as a mixture of Gaussians model. The EM algorithm can be used to learn the model parameters. In this case, the governing equations are (as before):

$$P(a_r|m, \mathbf{e}) = N(a_r|\mu(m)\sigma(m)) \tag{4.9}$$

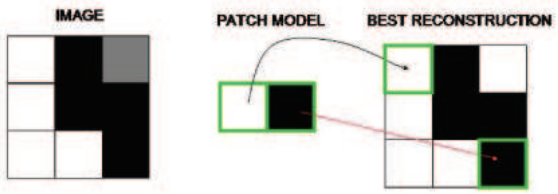


*Fig. 4.10:* **A model consisting of two single pixel patches:** The original image is on the left. The model mean is shown in the middle. The best reconstruction is shown on the right. In this case almost the entire original image is reproduced.

$$P(A|m,\mathbf{e}) = \prod_r P(a_r|m,\mathbf{e}) \tag{4.10}$$

The variables are the same as above. The key difference here is that there is now more than one mapping, m. The mappings are now drawn from a set of set of mappings, M. Nevertheless, the equations remain much the same. To find the value of a patch given a set of mappings, M, the sum of probabilities over all mappings in that set is found. The probabilistic model is now:

$$P(A|M,\mathbf{e}) = \prod_r \sum_{m\in M} P(a_r|m,\mathbf{e})P(m) \tag{4.11}$$

E-Step:

$$P(m|a_r,\mathbf{e}) = P(a_r|m,\mathbf{e})\frac{P(m)}{P(a_r)} \tag{4.12}$$

where $P(a_r)$ is calculated by marginalising over all mappings:

$$P(a_r) = \sum_{m\in M_r} P(a_r|m,e)P(m) \tag{4.13}$$

There is a different mapping set for each image pixel. This is to represent all possible mappings from the model to that image pixel. Hence, we write the mapping set as a function of image pixel label, $M_r$. Each mapping set, $M_r$ has n elements, where n is the total number of Gaussians in the mixture model. Here we assume a constant distribution for the prior over the mappings, $P(m)$, that is $P(m) = 1/|M|$. The calculation for $P(m|a_r,\mathbf{e})$ becomes:

$$P(m|a_r,\mathbf{e}) = \frac{P(a_r|m,\mathbf{e})}{\sum_{m\in M} P(a_r|m,e)} \tag{4.14}$$

The M-Step is again a weighted maximum likelihood calculation:

$$\hat{\mu} = \frac{\sum_r \sum_{m\in M_r} P(m|a_r,\mathbf{e})a_r}{\sum_r \sum_{m\in M_r} P(m|a_r,\mathbf{e})} \tag{4.15}$$

$$\hat{\sigma} = \frac{\sum_r \sum_{m \in M_r} P(m|a_r, \mathbf{e})(a_r - \mu(m))^2}{\sum_r \sum_{m \in M_r} P(m|a_r, \mathbf{e})} \tag{4.16}$$

### *4.3.4 Single patch Gaussian model*

The inspiration of patch models is to allow the patch size to be different from 1x1. Intuitively this is going to be a useful way of modeling data any time there are *local* patterns within that data. Also intuitively such a model will be *worse* at generating many images than a model of the same "size" which modeled single pixels individually. For pattern recognition this is a desirable effect as it makes the model more specific.

Figure 4.11 shows an example of a model which consists of a single patch of size 2x2. The patch model in this case is a multi-dimensional Gaussian model. In the specific case of 2x2 patches, it is equivalent to fitting a single 4-dimensional Gaussian to the image data, after the image has been decomposed into 2x2 patches. The dimensions of the 4 Gaussian distribution in the model are all independent of each other, so the covariance does not need to be estimated.

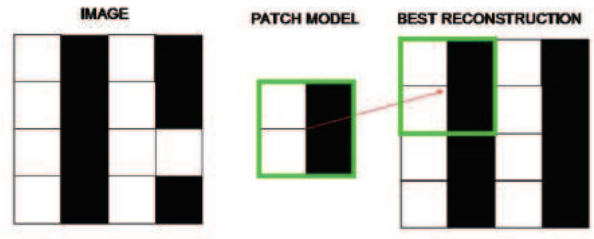


*Fig. 4.11:* **A model consisting of a single patch of size 2x2:** The original image is on the left. The model is shown in the middle. The best reconstruction is shown on the right.

The parameters of a single patch model can be learnt using the EM algorithm. The probabilistic generative model is now:

$$P(A_i|m, \mathbf{e}) = \prod_r P(a_r|m, \mathbf{e}) \tag{4.17}$$

$$P(A|m, \mathbf{e}) = \prod_i P(A_i|m, \mathbf{e}) \tag{4.18}$$

The variables are as above, except now the image, A, is separated into a number of patches, $A_i$, instead of being described as a collection of pixels. As in the single Gaussian case, there is only one mapping, for each image patch, m. A key difference is that the mapping, m, now links a set of positions in the model to a set of positions in the image. The probability of a pixel for a given mapping and model is still the same as for a single Gaussian though:

$$P(a_r|m,\mathbf{e}) = \prod_r N(a_r|\mu(m,r,e)\sigma(m,r,e)) \tag{4.19}$$

The mean and sigma are found by looking up the relevant position in the model. For a given mapping the image patch is the same size as the patch in the model that is being used. A single mapping aligns each pixel in the image patch with a "pixel" in the model. Hence the relevant mean for each image pixel now depends on the pixel index as well as the mapping and the specific patch model. This gives a different mean for each pixel in a single image patch.

The update equations for the EM algorithm are now:

E-Step:

$$P(m|A_i,\mathbf{e}) = P(A_i|m,\mathbf{e})\frac{P(m)}{P(A_i)} \tag{4.20}$$

where $P(A_i)$ is calculated by marginalising over all mappings, and the probabilities of all pixels within $P(A_i)$:

$$P(A_i) = \sum_{m\in M}\prod_r P(a_r|m,e)P(m) \tag{4.21}$$

Since there is only one mapping, the E step update equation is $P(m|A_i,\mathbf{e}) = 1$.

The M-Step is a weighted maximum likelihood calculation, found for each $\mu$ & $\sigma$ in the model:

$$\hat{\mu} = \frac{\sum_i \sum_{m\in M} P(m|A_i,\mathbf{e})a_{r(i)}}{\sum_r \sum_{m\in M} P(m|A_i,\mathbf{e})} \tag{4.22}$$

$$\hat{\sigma} = \frac{\sum_r \sum_{m\in M} P(m|A_i,\mathbf{e})(a_{r(i)} - \mu(m))^2}{\sum_r \sum_{m\in M} P(m|A_i,\mathbf{e})} \tag{4.23}$$

### *4.3.5 Mixture of Gaussian patches model*

The natural extension from a single patch model is to a mixture of patches model. Here a multi-dimensional mixture of Gaussians model is presented.

For an n patch model, the probabilistic generative model is now:

$$P(A_i|m,\mathbf{e}) = \prod_r P(a_r|m,\mathbf{e}) \tag{4.24}$$

$$P(A|m,\mathbf{e}) = \prod_i P(A_i|m,\mathbf{e}) \tag{4.25}$$

The probability of a pixel for a given mapping and model is still the same as for a single Gaussian:

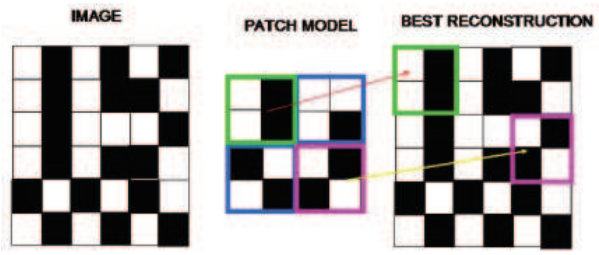


*Fig. 4.12:* **A model consisting of a number of patches of size 2x2:** The original image is on the left. The model is shown in the middle. The best reconstruction is shown on the right.

$$P(a_r|m,\mathbf{e}) = \prod_r N(a_r|\mu(m,r,e)\sigma(m,r,e)) \tag{4.26}$$

The update equations for the EM algorithm are now:

E-Step:

$$P(m|A_i,\mathbf{e}) = P(A_i|m,\mathbf{e})\frac{P(m)}{P(A_i)} \tag{4.27}$$

where $P(A_i)$ is calculated by marginalising over all mappings, and the probabilities of all pixels within $P(A_i)$:

$$P(A_i) = \sum_{M_i}\prod_r P(a_r|m,e)P(m) \tag{4.28}$$

As in the mixture of Gaussians case, there is a different mapping set for each image patch to be modeled, hence we write the mapping set as a function of image patch, $M_i$. The assumptions made have not changed from previous models, and the distributions probability distributions are still assumed to be normal. Once again assuming a flat distribution for the prior, $P(m)$, the E step update equation is now:

$$P(m|A_i,\mathbf{e}) = \frac{P(A_i|m,\mathbf{e})}{\sum_{m\in M_i} P(A_i|m,\mathbf{e})} \tag{4.29}$$

The M-Step is a weighted maximum likelihood calculation, found for each $\mu$ & $\sigma$ in the model:

$$\hat{\mu} = \frac{\sum_i \sum_{m\in M_i}\sum_{r(i)} P(m|A_i,\mathbf{e})a_r(i)}{\sum_i \sum_{m\in M_i}\sum_{r(i)} P(m|A_i,\mathbf{e})} \tag{4.30}$$

$$\hat{\sigma} = \frac{\sum_i \sum_{m\in M_i}\sum_{r(i)} P(m|A_i,\mathbf{e})(a_r(i)-\mu(m))^2}{\sum_i \sum_{m\in M_i}\sum_{r(i)} P(m|A_i,\mathbf{e})} \tag{4.31}$$

### 4.3.6 Epitomes

The models I have outlined so far are supposed to demonstrate a thought process for building up to the epitome model described by Jojic et al [33]. Viewing the epitome as a natural extension of the hierarchy of models I have described above provides a framework for understanding the epitome model. In this light the workings of the epitome model are more transparent than trying to understand the epitome model in isolation.

Jojic et al formulate a probabilistic generative patch based model which they call an epitome. An epitome describes an image by copying patches from a smaller image, the epitome mean, which captures most of the structures required to construct the original image. By allowing the patches used to be chosen from anywhere in the epitome mean the problem of tessellation choice in the image is diminished. Another contribution of epitomes is that of variable patch size. As in the previous models described in this section, each "position" in the model contains two parameters; a mean and standard deviation of a normal distribution. Positions in the epitome and positions in an image are related via a mapping of their coordinates, as in previous models.

The original image is considered to be tesselated into non-overlapping patches, as shown in Figure 4.2 b & Figure 4.2 c. Each patch is then modeled individually. Each patch is assumed to be generated by copying a region of the epitome and adding pixelwise error, as shown in Figures 4.3 & 4.4. Coordinate positions are denoted by (x,y) as in the Cartesian plane. Each position in the epitome contains two parameters, a mean, $\mu(x, y)$ and a variance, $\sigma(x, y)$, of a normal distribution. The mean contains information on colour and intensity, while the variance can be thought of as a level of uncertainty associated with that mean. An alternative description is that the variance part of the epitome specifies the tolerance to mismatched pixels between the related mean position in the epitome and an image pixel. The quality of match is defined as a probability that the data was drawn from a normal distribution of mean and variance specified in the epitome.

The mean part of the epitome is a special case of a patch library, forming a condensed representation of an equivalent patch library. It is condensed since due to its definition as a continuous model, and the form of the allowed mappings, it contains more possible descriptions of a piece of data than a patch model with the same number of parameters. Looked at in the opposite direction, the epitome mean image requires considerably less parameters than an equivalent naive patch library. For example an epitome mean of size 30 x 30 (that is containing 30x30 normal distributions), using a single patch size of 4 x 4, requires $30^2 = 900$ numbers to specify. The equivalent patch library would contain $26 \times 26$ patches, hence requiring $676 \times 4 \times 4 = 10816$ numbers.

If variable patch sizes are used, the saving becomes even greater.

Each part of the epitome must approximately represent a number of patches in the image, and so each epitome patch is an averaged version of a number of similar patches within the image. The epitome can be used to create a reconstructed approximation to the original image. The reconstruction will, in general, lack some fine detail that was present in the original image.

In summary, an epitome (and the other models described in this section) can be thought of as a way of "coding" an image; that is a condensed representation of an image. In this sense it is broadly similar to the many other approximations found in maths and computer science such as a finite sum Taylor expansion approximation of a function, or a description of a periodic function as a finite sum of sines and cosines, or a JPEG compression. It is only similar in its broad function, which is to provide a condensed and manageable representation of something which can be used (for some purpose) more easily. Our purpose for defining such a representation is to allow comparison of the similarity of different images.

The epitome allows the patch size to vary, and size is chosen when the epitome is trained. Figure 4.13 shows an example of an epitome consisting of multiple patches of variable size.

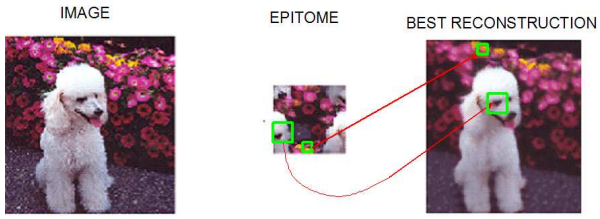


*Fig. 4.13:* **Epitome with variable patch size.** The original image is on the left. The epitome mean is shown in the middle, and the best reconstruction is shown on the right. (Adapted from [33] )

### *4.3.7 Epitome equations*

Consider an image I of size NxM, which is split into C different patches, $A_i$ consisting of n(i) pixels. For every $A_i$ there are mappings, $M_i$, from epitome coordinates to image coordinates in the patch. For a given epitome, e, and mapping the patch is modeled as being drawn from probability distributions given by the relevant epitome means and variances. For a pixel within the image patch, the probability of that pixel under the model is given by the probability that it was drawn from a normal distribution with mean and variance specified in the appropriate epitome position.

Under assumption of independence of generation, the probability of an entire patch given a mapping is the joint probability of all the pixels being drawn

from the specified independent normal distributions in the epitome. It should be noted that this assumption is contrary to the purpose of the models, which is to describe locally repeating patterns. However, in most situations there would be insufficient data to estimate the full covariance matrix for a patch model. In models using continuous representation (such as the epitome), it is even more difficult to learn the covariance as for each position in the model the number of other model positions with which it interacts is greater than the patch size.

Given this proviso about independence, the probability of a patch can be expressed as:

$$P(A_i|M_i, e) = \prod_{r \in A_i} N(a_r(i); \mu_{M_i(r)}, \sigma_{M_i(r)}) \tag{4.32}$$

where $a_r(i)$ is the pixel intensity of the rth pixel in patch $A_i$, and N(x;y,z) is the probability of observation x coming from a normal distribution with mean y and variance z. $\mu$ and $\sigma$ are the mean and variance at a specified position. The position being used is specified by a mapping, written as a subscript.

On assumption of independence of patch generation, the joint probability of a collection of image patches, a set of mapping sets, and a specified epitome is given by:

$$P(A, M, e) = P(e) \prod_{i=1}^{C} P(M_i) P(A_i|M_i, e) \tag{4.33}$$

Where A is the union of all the patches, $A_i$, and M is the union of all the mapping sets, $M_i$. No prior knowledge about the probability of a given epitome is assumed, hence P(e) is constant, and is ignored. The Expectation Maximisation (EM) algorithm is used to calculate a local maximum posterior over the means and variances in the epitome, in the presence of hidden variables, which are the mappings. The update equations for the EM algorithm are obtained by marginalising the logarithm of the joint distribution of data and parameters over the hidden variables. Finding the distribution of the posterior over the hidden variables given the data and a current estimate of the parameters allows a lower bound for the logarithm of the joint probability of the data and parameters to be found. This is described in detail in Appendix A.

The parameters in the epitome are first initialised to white noise by Jojic et al., while for learning I initialised model means to image data plus Gaussian noise to give them a "better" starting position, started with equal high variance for each normal distribution. This is useful to help avoid the model parameters getting stuck in any local probability maxima far from parameters

which describe the data. The ordering of the patches in the model is not specified. The final "appearance" of any part of the model depends on which part of the data it starts "near".

### *4.3.8 Update equations for epitome learning in the EM algorithm*

In the Expectation (E) step (see Appendix A) we want to find the following distribution over the mappings:

$$P(M_i|A_i, e) = P(A_i|M_i, e)P(M_i)/P(A_i) \tag{4.34}$$

The prior probability over the mappings, $P(M_i)$ is assumed to be constant initially. Hence the update equation for the E step is:

$$P(M_i|A_i, e) = kp(M_i) \prod_{r \in A_i} N(a_i; \mu^{t-1}_{M_i(r)}, \sigma^{t-1}_{M_i(r)}) \tag{4.35}$$

The normalising factor $k = \frac{1}{P(A_i)}$ does not need to be known explicitly and instead the distribution is normalised by dividing by the sum of probabilities over all possible transformations. Note, this normalisation is done per image patch (that is per mapping set), not over the whole image. $\mu^{\mathbf{t-1}}$ and $\sigma^{\mathbf{t-1}}$ are the means and variances calculated at the previous iteration, t-1, at a specific position in the epitome.

The purpose of the Maximisation (M) step is to find the values of the parameters, the means and the variances, that maximise the likelihood of the patch given the current expected distribution of the mappings. The maximum likelihood calculation reduces to finding the mean and variance of the data weighted by the probability of the associated mappings. The M step is:

$$\hat{\mu} = \frac{\sum_{i=1}^{C} \sum_{r \in A_i} \sum_{\{M_i\}} a_r q(M_i)}{\sum_{i=1}^{C} \sum_{r \in A_i} \sum_{\{M_i\}} q(M_i)} \tag{4.36}$$

$$\hat{\sigma}^{\mathbf{2}} = \frac{\sum_{i=1}^{C} \sum_{r \in A_i} \sum_{\{M_i\}} (a_r - \hat{\mu_j})^2 q(M_i)}{\sum_{i=1}^{C} \sum_{r \in A_i} \sum_{\{M_i\}} q(M_i)} \tag{4.37}$$

In words, equation 4.36 reads, the mean equals the pixel intensity in the image multiplied by the expected probability of a mapping summed over all mappings, pixel positions and patches, divided by the sum over all mappings. Equation 4.37 reads similarly, except with a modification from mean to variance. Note, q($M_i$) is used in the above equations in place of $P(A_i|M_i, e)$ for brevity, since $P(A_i, M_i, e)$ and $P(A_i|M_i, e)$ differ only by the constant P($M_i$) in this setup.

## 4.4 Blueprints

The epitome paper [33] is primarily a description of the model and a method of finding the parameters of the model. Although the use of the model in segmenting foreground and background is demonstrated, other uses of the model are left for the reader to decide. Epitome style models can be used for image classification. Next I develop some specific adaptations to the epitome model to create a probabilistic model suitable for image classification on our image dataset (see Chapter 1). The class of models, which we have developed we call blueprints. A blueprints is an adapted version of a mixture of patches model, described above. Blueprints build on the concept of an epitome in three ways:

1. Blueprints model image classes instead of single images.
2. Blueprints are designed for use as a classification tool
3. Blueprints use priors over the mappings to improve classification performance.

A blueprint is a multi-dimensional Gaussian model of a set of images, with assumed independence of the different dimensions. As before the model is represented in continuous form so that patches can be selected from anywhere in the model. The model can be represented as an image, as for the models described previously. As for any digital image, there is an underlying pixel grid, shown in Figure 4.14.

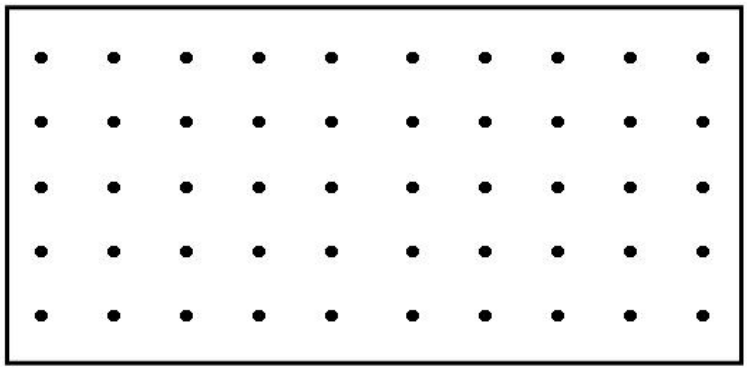

*Fig. 4.14:* **Model coordinate system** The underlying grid coordinate system for blueprints. Polar coordinates are used for blueprints. The angle increases left to right, and the radius increases top to bottom.

The blueprint model contains two parameters at each pixel position. At each pixel position in the model there is a mean value stored, as shown in Figure 4.15. There is also a standard deviation stored at each pixel position in the model. This is shown in Figure 4.16. For visualisation purposes this

allows a given blueprint to be shown as two separate images; a mean image and a variance (or standard deviation) image.

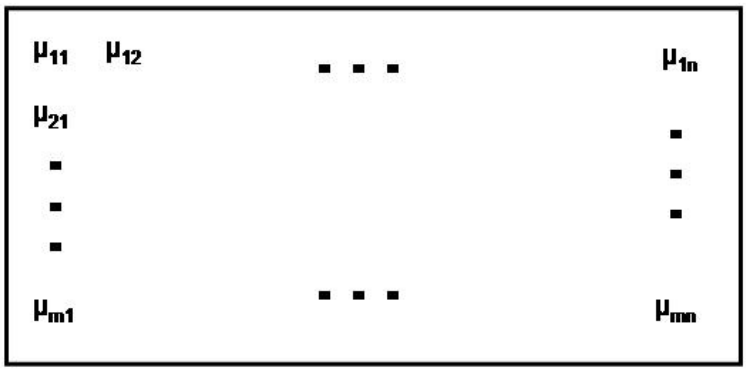


*Fig. 4.15:* **Model means** Each pixel position in the blueprint stores the mean of a normal distribution.

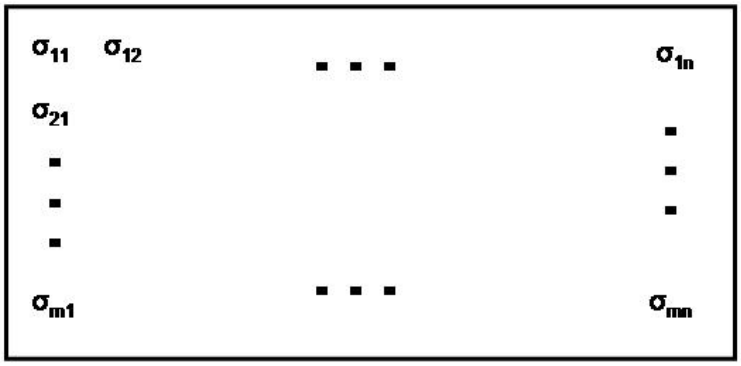


*Fig. 4.16:* **Model standard deviations** Each pixel position stores the standard deviation of a normal distribution.

Overall, the model can be thought of as containing a normal distribution (or more generally any probability distribution could be used, but here all probability distributions are normal distributions) at each model position. This summary view of the model is shown in Figure 4.17.

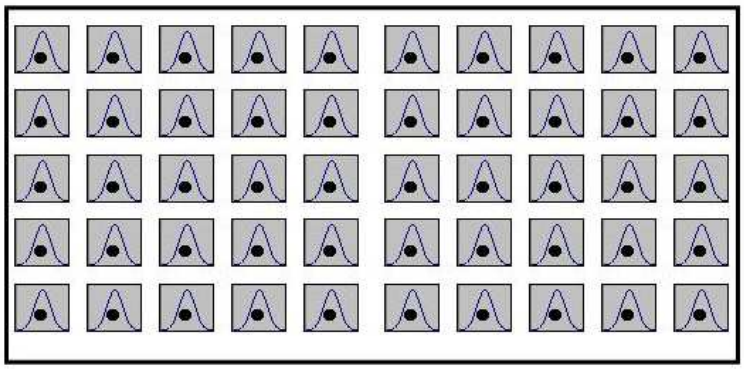

*Fig. 4.17:* **Model probability distributions** Overall, the blueprint model can be interpreted as a grid with a probability distribution stored at each pixel position.

Since we aim to perform image classification, blueprints are learnt from a set of images of the same class, rather than single images like epitomes. A number of blueprints from different image classes will be learnt in order to provide an overall classification tool for unseen images. As in the build-up examples described earlier in the chapter, the parameters of the model for given training data will be fitted using the EM algorithm. An adaptation of blueprints for the specific task of image classification is that a spatial prior is introduced to modify the probability of the mappings between the model and the data. This will take the form of a hard spatial prior although other priors are possible. Each of the above properties of blueprints are discussed in detail in the sections below.

The blueprint models will use a fixed patch size that will be determined in the next chapter. The blueprint models will all be based on cell regions (see Chapter 3) that have been converted into polar coordinates (see Chapter 3). This means that square image patches in polar coordinates actually represent "wedges" of the original cell region. This is shown in Figure 4.18

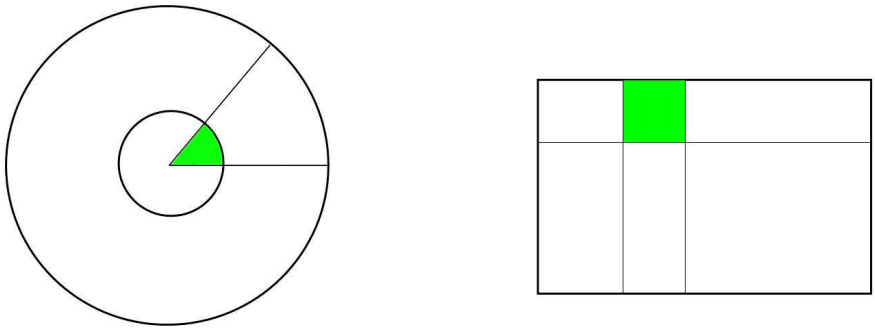

*Fig. 4.18:* **Coordinate transformation** Wedges of circle in the original image coordinates (left) are transformed to square regions in polar coordinates (right). These "wedges" of data are the basic units being modeled.

### 4.4.1 Blueprints: Models of image classes

Blueprints are used to represent a class of images which have some common characteristic. This means that to train blueprints a number of different images are used, and hence the update equations operate over all images in the training set:

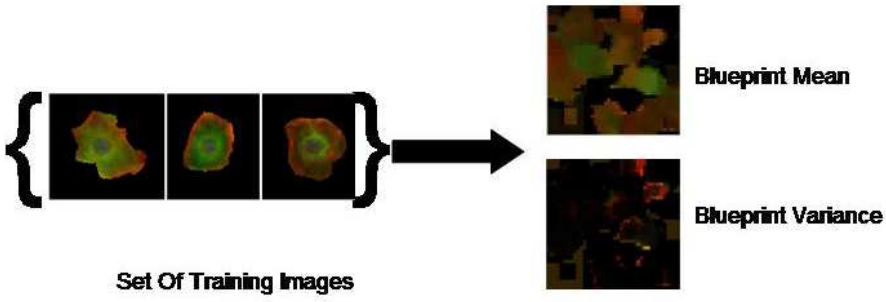


*Fig. 4.19:* **Blueprint:** A generative model trained using multiple images

$$\hat{\mu} = \frac{\sum_{l=1}^{L}\sum_{i=1}^{C}\sum_{r\in A_i}\sum_{\{M_i\}} a_{r,l} q(M_{i,l})}{\sum_{l=1}^{L}\sum_{i=1}^{C}\sum_{r\in A_i}\sum_{\{M_i\}} q(M_{i,l})} \tag{4.38}$$

$$\hat{\sigma}^{\mathbf{2}} = \frac{\sum_{l=1}^{L}\sum_{i=1}^{C}\sum_{r\in A_i}\sum_{\{M_i\}} (a_{r,l} - \hat{\mu_j})^2 q(M_{i,l})}{\sum_{l=1}^{L}\sum_{i=1}^{C}\sum_{r\in A_i}\sum_{\{M_i\}} q(M_{i,l})} \tag{4.39}$$

where the index l denotes the training object, and there are L training objects in total. This is done for each colour channel separately, and can be considered to be three blueprints joined into one. The E step calculation also remains the same except that now the set of mapping sets is much larger since the mappings go from the epitome to several different images, and a set of mappings is required for each image patch.

Shown in Figure 4.20 is the appearance of the blueprint after different numbers of iterations of the EM algorithm. Thirty training images and five iterations of the EM algorithm were used to train this blueprint. Each training image is an image of a cell (region of interest as defined in Chapter 3). There is only one class of cell in the images, so the model is intended to describe one phenotype. As will be seen later, a separate blueprint is learnt for each class.

### 4.4.2 Priors over mappings

Introducing a prior over the mappings biases which mappings will be used. This is used so that patch regions in the epitome are placed in similar positions, relatively, in the blueprint coordinates as they are found in the training images. This allows for specification of where features should be found in a

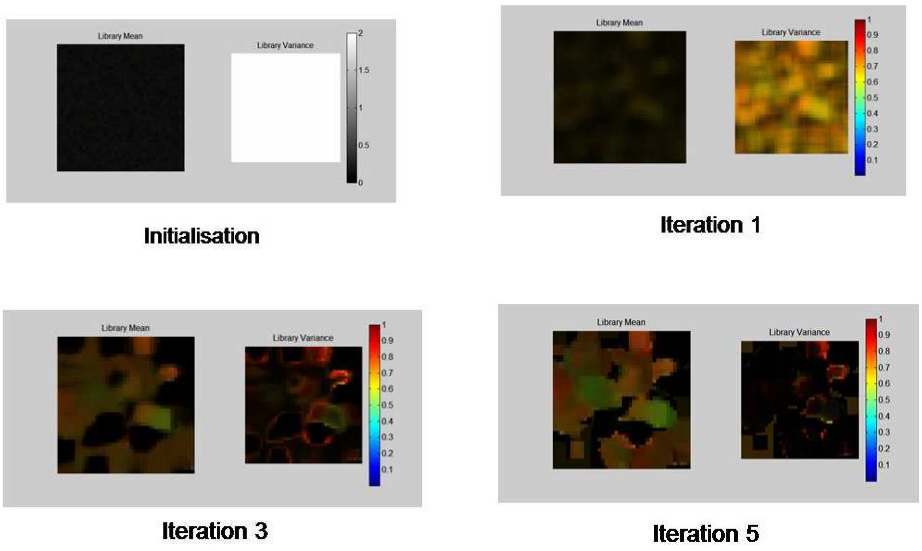


*Fig. 4.20:* **Learning the blueprint:** Shown in each sub-image is the mean and variance of the blueprint. Underneath is the number of EM iterations that have been performed.

given image class, and actually *reduces* the ability of the blueprint to model many images. This is a desirable quality since it forces the blueprint to more specifically model a particular image class. For the cells considered here, due to their rotational symmetry, the only position that matters is distance from the nucleus. Hence the prior that we introduce only biases the mappings in the radial direction, but allows for placement anywhere on the angular axis. I used a hard spatial prior allowing only mappings between the same radial distance, using a delta function: $\delta(i,j) = 1$ if $i = j$ and $\delta(i,j) = 0$ if $i \neq j$

### *4.4.3 Blueprints: Modified epitomes for classification*

The purpose of blueprints is to act as a representation of an image class to enable comparisons between different images.

Once a blueprint, B, has been learnt, the likelihood of B for a test image, I, can be determined as follows:

$$P(I|B) = \prod_i \sum_{m \in M_i} P(A_i|M_i, B) \tag{4.40}$$

This is found by performing the E step from the EM algorithm. This can be done for any number of blueprints allowing for comparison of which blueprint models an image best. This comparison can be normalised into a probability distribution for n blueprints, $B_1, ...B_n$ as follows:

$$P(B_k|I) = \frac{P(I|B_k)P(B_k)}{\sum_i P(I|B_i)P(B_i)} \tag{4.41}$$

If a discrete assignment of an image to a particular class is required, then

this can be obtained by assigning it to the same class as the class used to train the blueprint, $B_{MAP}$ with the highest maximum a posteriori probability:

$$B_{MAP} = arg \max P(B_i|I) \tag{4.42}$$

# 5. CLASSIFICATION USING GENERATIVE MODELS

In this chapter I demonstrate how to use the generative models developed in the previous chapter to perform recognition and classification. The performance is compared with that of a benchmark in classification literature: Support Vector Machines (SVMs) [73]. Here the comparison will be fair, not encompassing those situations where generative models naturally excel, such as cases with missing or partial data, or the combination of multiple generative models to reach decisions.

The basic hypothesis is that blueprints can be used to classify cells. This will be tested by generating two different blueprints from two different sets of training cells: normal cells and round cells. The likelihood of a blueprint for a given test cell is found, using equation 5.1 (see section 4.4.3). Equation 5.2 (see section 4.4.3) is used to compare the performance of two blueprints.

$$P(I|B) = \prod_i \sum_{m \in M_i} P(A_i|M_i, B) \tag{5.1}$$

$$P(B_k|I) = \frac{P(I|B_k)P(B_k)}{\sum_i P(I|B_i)P(B_i)} \tag{5.2}$$

## *5.1 Results on hand segmented data.*

To demonstrate the validity of the classification method we first tested it using hand segmented cells. Two images from the dataset were identified for this purpose, one which contained wildtype cells, and another which contained a cell phenotype which resulted in cells appearing more rounded than wildtype cells. For convenience we will refer to these cells of these two phenotypes as "normal" and "round" cells. thirty normal and thirty round cells were hand segmented using photoshop (see Figure 5.1). These cell cutouts are used in Experiments 1 - 3. Fifteen normal cells were used to train a normal blueprint, and fifteen round cells were used to train a round blueprint. The thirty remaining cells, fifteen normal and fifteen round, were used for testing. In each case a blueprint of size 20x20 was learnt. Throughout, 5 iterations of the EM algorithm are performed in the learning process.

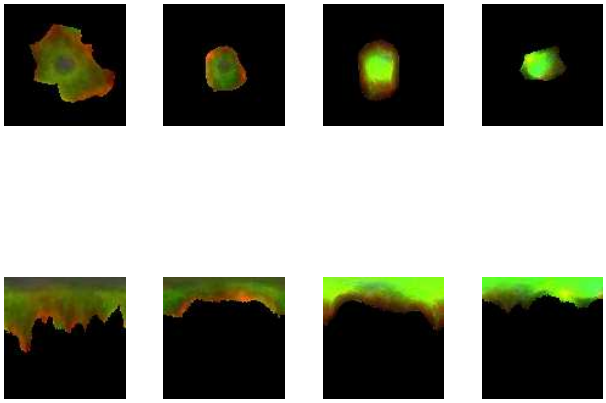

*Fig. 5.1:* **Example hand segmented cells:** The top row shows the cells in cartesian coordinates. The bottom row shows the cells in polar coordinates. The two cells on the left are normal cells. The two cells on the right are round cells

### *5.1.1 Experiment 1 - Calculating the posterior*

We seek to evaluate the posterior probability that a cell belongs to the normal class, using equation 4.41. A patch size of 5x5 was used to learn each blueprint. The posteriors are given in the table below.

| Cell ID | Posterior Probability | Actual Cell Type | Predicted Cell Type |
|---|---|---|---|
| 1 | 1 | Normal | Normal |
| 2 | 4.03E-74 | Normal | Round |
| 3 | 1 | Normal | Normal |
| 4 | 1.37E-28 | Normal | Round |
| 5 | 1.41E-07 | Normal | Round |
| 6 | 0.99 | Normal | Normal |
| 7 | 1 | Normal | Normal |
| 8 | 1 | Normal | Normal |
| 9 | 4.75E-27 | Normal | Round |
| 10 | 1 | Normal | Normal |
| 11 | 1 | Normal | Normal |
| 12 | 1 | Normal | Normal |
| 13 | 1 | Normal | Normal |
| 14 | 1 | Normal | Normal |
| 15 | 1 | Normal | Normal |
| 16 | 8.75E-229 | Round | Round |
| 17 | 4.37E-171 | Round | Round |
| 18 | 6.99E-218 | Round | Round |
| 19 | 4.58E-244 | Round | Round |
| 20 | 4.34E-239 | Round | Round |
| 21 | 2.20E-225 | Round | Round |
| 22 | 6.23E-147 | Round | Round |
| 23 | 1.04E-221 | Round | Round |
| 24 | 1.53E-214 | Round | Round |
| 25 | 1.06E-207 | Round | Round |
| 26 | 7.14E-135 | Round | Round |
| 27 | 1.50E-111 | Round | Round |
| 28 | 7.93E-206 | Round | Round |
| 29 | 1.16E-240 | Round | Round |
| 30 | 2.85E-201 | Round | Round |

Equation 5.3 (see section 4.4.3 is used to classify cells as normal or round.

$$B_{MAP} = arg\max P(B_i|I) \tag{5.3}$$

Classifying according to equation 5.3, four errors are made. Cell IDs 2,4,5 and 9 have a higher probability of being round cells than normal cells. The classifier is very certain of the result even in cases when it is wrong: the misclassified normal cells are assigned a *very* small probability of being normal. There are two ways this result could be improved. One is to use more training data. The other is to represent the uncertainty in the model parameters, which

is currently completely unrepresented. However, looking closely at the data, then it can be seen that the cells are correctly *ordered* by probability. Each of the incorrectly classified cells has a higher probability of being a normal cell than any of the round cells. The posterior probability defines two distinct classes. Hence, we try to improve on classification by using a threshold different from 0.5.

### *5.1.2 Experiment 2 - Classification via a learnt threshold*

Experiment 1 was repeated, except the goal is optimal classification rather than calculation of the posterior. To do this, a normal distribution of the class posterior of the normal cell training data, and a normal distribution of the class posterior of the round cell training data are calculated. Test cells are then assigned to classes using the E step of the EM algorithm for a mixture of two Gaussians. The results are given in the table below:

| Cell ID | Actual Cell Type | Classification |
|---|---|---|
| 1 | Normal | Normal |
| 2 | Normal | Round |
| 3 | Normal | Normal |
| 4 | Normal | Normal |
| 5 | Normal | Normal |
| 6 | Normal | Normal |
| 7 | Normal | Normal |
| 8 | Normal | Normal |
| 9 | Normal | Normal |
| 10 | Normal | Normal |
| 11 | Normal | Normal |
| 12 | Normal | Normal |
| 13 | Normal | Normal |
| 14 | Normal | Normal |
| 15 | Normal | Normal |
| 16 | Round | Round |
| 17 | Round | Round |
| 18 | Round | Round |
| 19 | Round | Round |
| 20 | Round | Round |
| 21 | Round | Round |
| 22 | Round | Round |
| 23 | Round | Round |
| 24 | Round | Round |
| 25 | Round | Round |
| 26 | Round | Round |
| 27 | Round | Round |
| 28 | Round | Round |
| 29 | Round | Round |
| 30 | Round | Round |

*Fig. 5.2:* **Classification**

One error is made. Cell 2, shown in Figure 5.3, is misclassified as being round. 96.7 % of the cells were correctly classified.

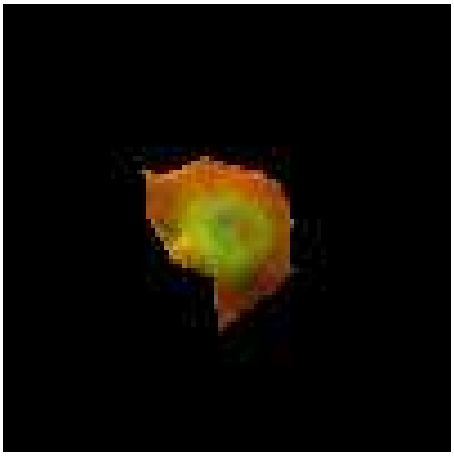

*Fig. 5.3:* **Misclassified Cell** The misclassified cell is shown.

## 5.2 Parameter Selection

There a number of parameters that are used in training a model. In this section the values for these parameters are optimised for the task of recognition. Note there are other ways that could be used to find values for these parameters, such as the probability of the training or test data under the learnt models. However, keeping our end goal in mind, the model parameters have been optimised to perform binary classification. To achieve this the following four distinct sets of cells were chosen by eye to provide both "easy" and "hard" comparisons:

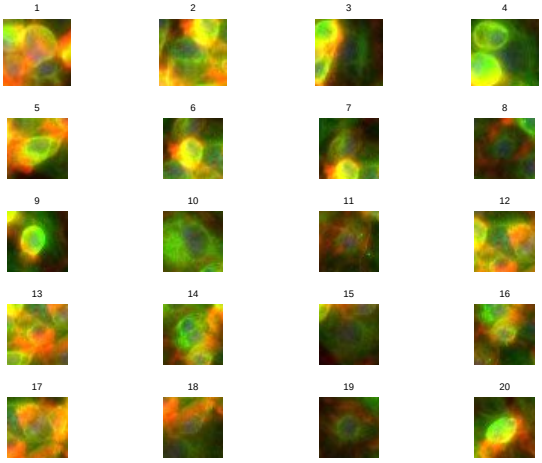

*Fig. 5.4:* **Parameter optimisation** Example cells "Set1"

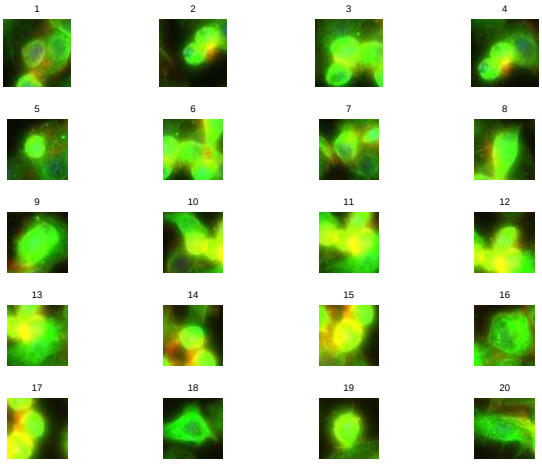

*Fig. 5.5:* **Parameter optimisation** Example cells, "Set2"

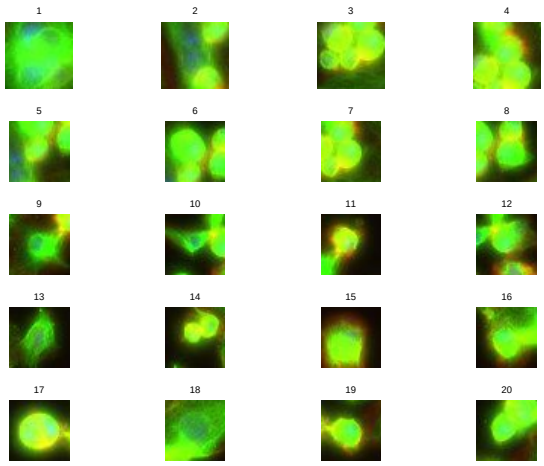

*Fig. 5.6:* **Parameter optimisation** Example cells, "Set3"

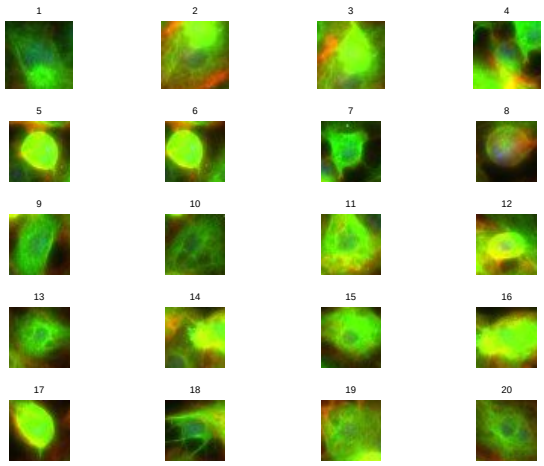

*Fig. 5.7:* **Parameter optimisation** Example cells, "Set4"

The above sets of cells were used in all permutations of binary tests to determine an accuracy for a given parameter set. Twenty example cells are shown in each case, although a total of thirty (inclusive of those shown) were used for training (except for on the training optimisation results) and another thirty were used for testing. The schematic in 5.8 gives the layout of the graphs in each of the parameter optimisation tests, showing which sets of cells are compared in each position.

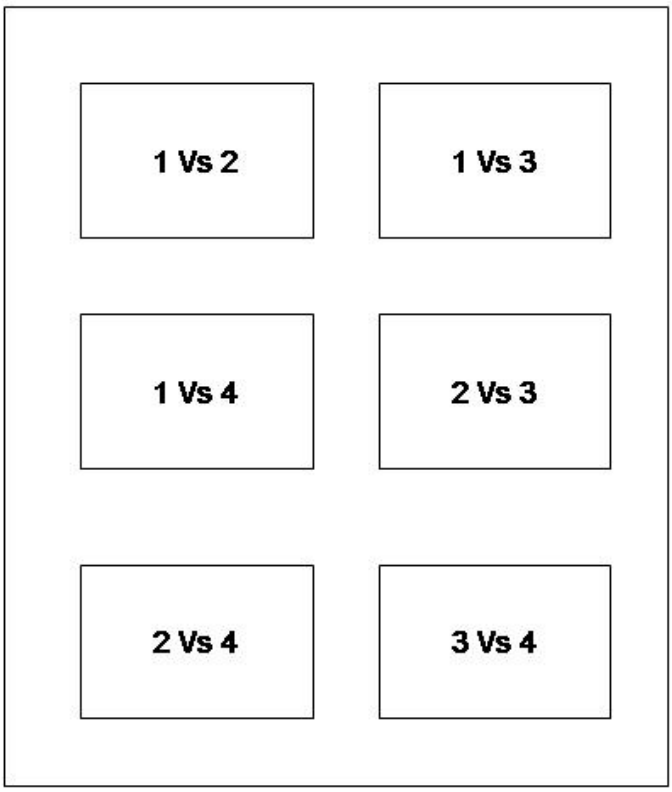


*Fig. 5.8:* **Schematic for graph results below**

Figure 5.8 is a schematic for the parameter optimisation graphs in the following sections, showing which cell set is compared which other cell set in each position.

### 5.2.1 Patch Size

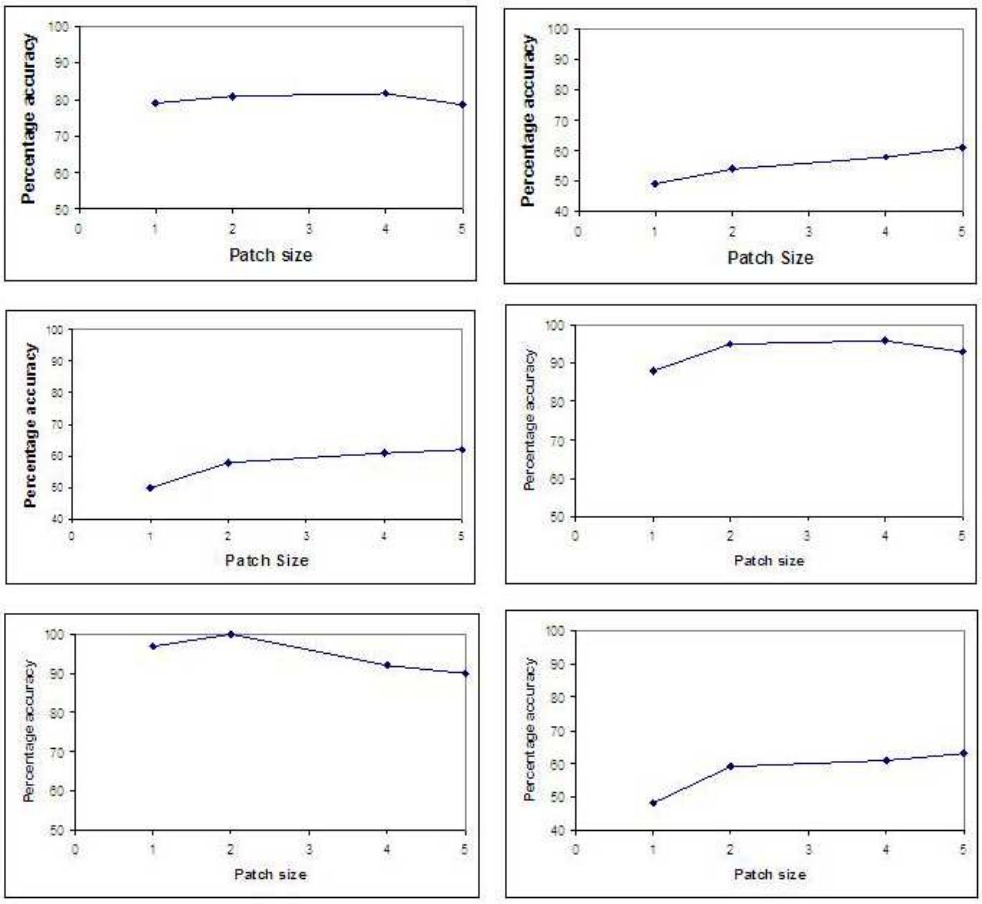


*Fig. 5.9:* **Graphs showing accuracy of classification on the test sets as a function of patch size**

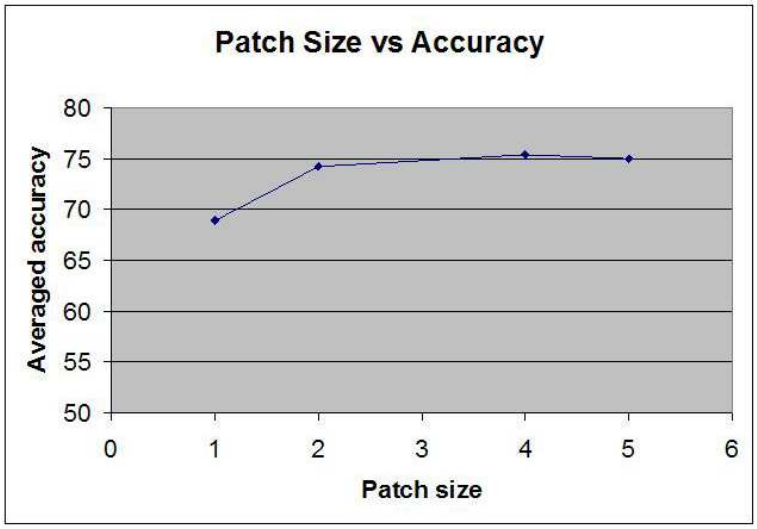


*Fig. 5.10:* **Graph showing the patch size vs accuracy averaged over collected data**

There is a trade off in increasing the patch size. A smaller patch size is able to model the data more accurately, but at the same time becomes better

at modeling arbitrary data. From the trial tests run with varying patch size, the optimal patch size to use is not clear cut. However, from the average classification accuracy graph (Figure 5.10), the trend is for the accuracy to increase as the patch size increases, reaching a plateau by 5x5 patches. Since the aim is to capture locally repeating patterns, larger patch sizes are preferred to smaller patch sizes all else being equal. Hence a square patch size of 5 is used for future models.

### 5.2.2 Model size

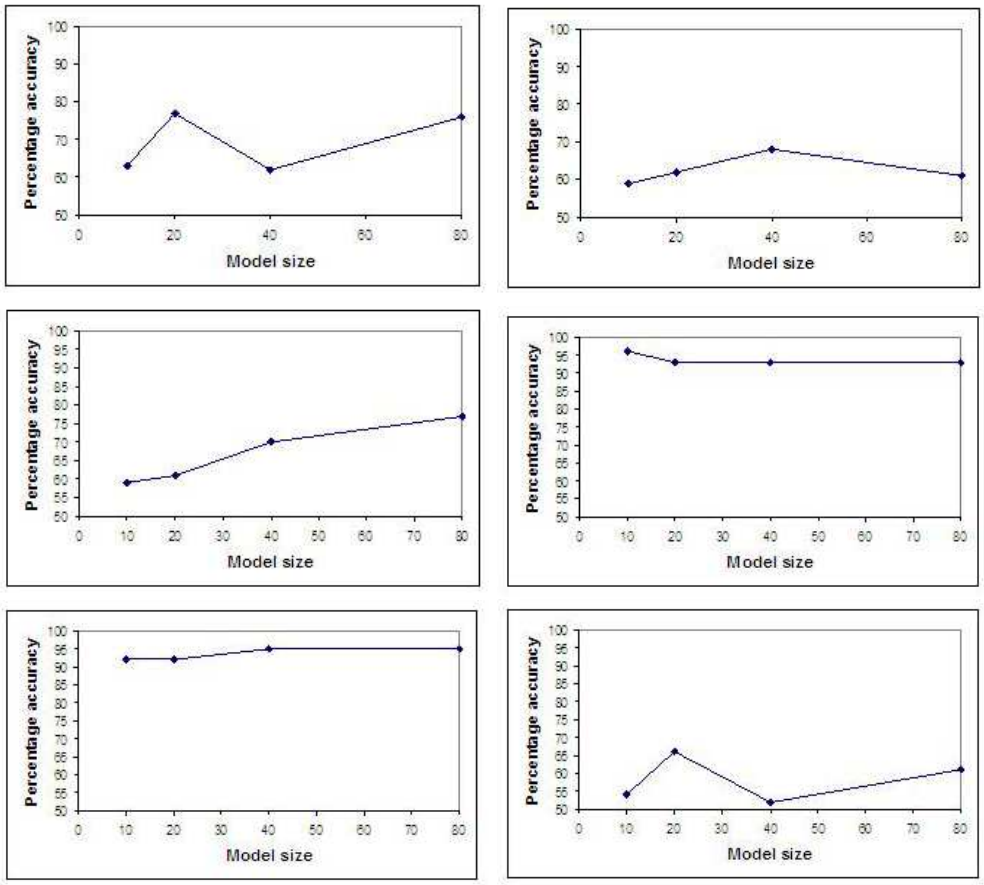


*Fig. 5.11:* **Graphs showing accuracy of classification on the test sets as a function of model size**

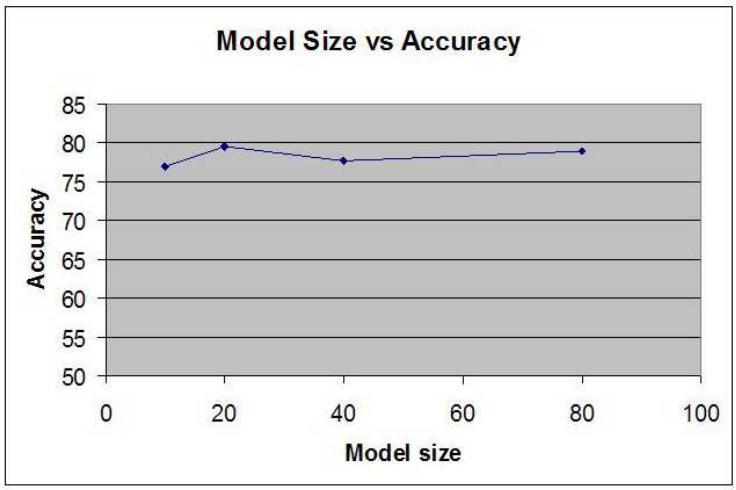


*Fig. 5.12:* **Graph showing the model size vs accuracy averaged over collected data**

The trial data on the four cell test sets suggests that there is not necessarily an optimal model size. Part of this is since different data sets will have differing amounts of data which needs to be "encoded" by the model. Increasing the

model size allows better modeling of the data, but also increases the number of possible alternative data sets that could be modeled equally well by that model. It is important to choose a model size which is large enough that it can capture more than just the broad features of the data. Larger models require more calculations both to learn in the first place, and also when used for recognition tasks. For these reasons a modest model size of 20x20 was chosen for future models, although it is acknowledged that an argument could be made for other model sizes.

### 5.2.3 Number Of Iterations

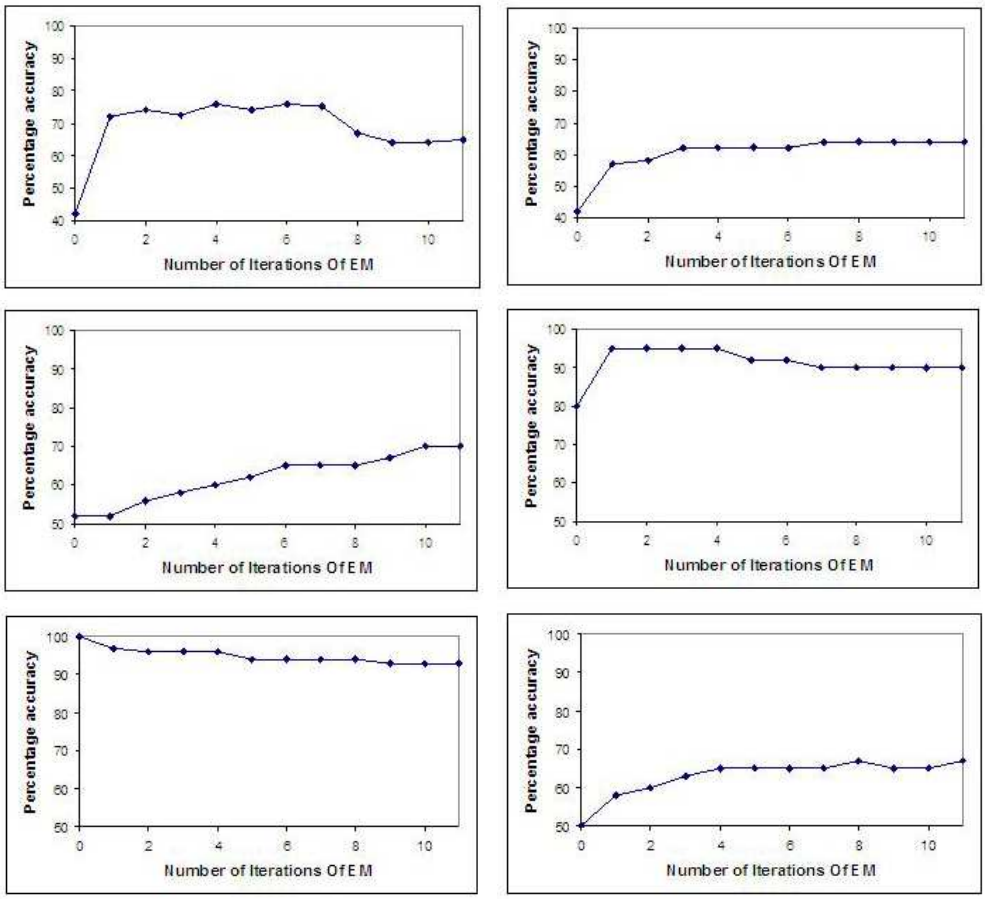


*Fig. 5.13:* **Graphs showing accuracy of classification on the test sets as a function of the number of iterations of the EM algorithm performed in the model learning stage**

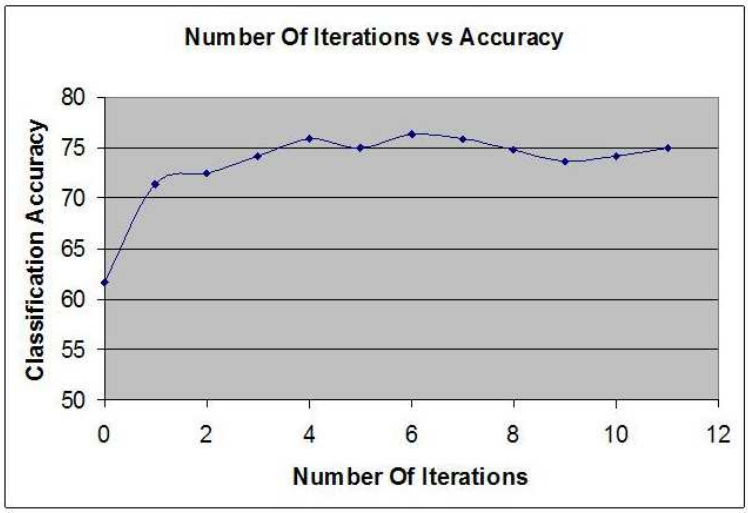


*Fig. 5.14:* **Graph showing the number of iterations of the EM algorithm used in learning vs accuracy averaged over collected data**

As can be seen from the graphs above accuracy reaches a plateau around 5 iterations, and then oscillates slightly. This is in line with what is heuristically chosen in the literature when the EM algorithm is used. Hence we choose a value of 5 iterations for all further uses of the EM algorithm.

It is interesting to note the odd occurrence of 100% accuracy after 0 iterations in the case of set 2 against set 4 (bottom left of Figure 5.13). This example underlines the complex relation between how well a model describes the training data, and how well a model distinguishes members of the training class from other classes. The means of the models are initialised to one of the training images plus random noise, and the variance is initialised to 10,000 (a high value). Due to this initialisation process for the models, fifty percent classification accuracy is not expected after zero iterations of the EM algorithm. Clearly the models after zero iterations are likely to give low probability values for the data they are intended to describe, since the parameter have not been optimised at all. However, the key point is that in this instance they consistently gave even lower probability values to data that were not of the same type. The rest of the data shows that in general scheme this result was a fluke occurrence, but this occurrence nevertheless provides excellent insight into the comparison process.

### 5.2.4 *Training Set Size*

The training set was split into the smaller subsets. The same test cells as in the above tests were used. The results of these tests are given in Figures 5.15 & 5.16.

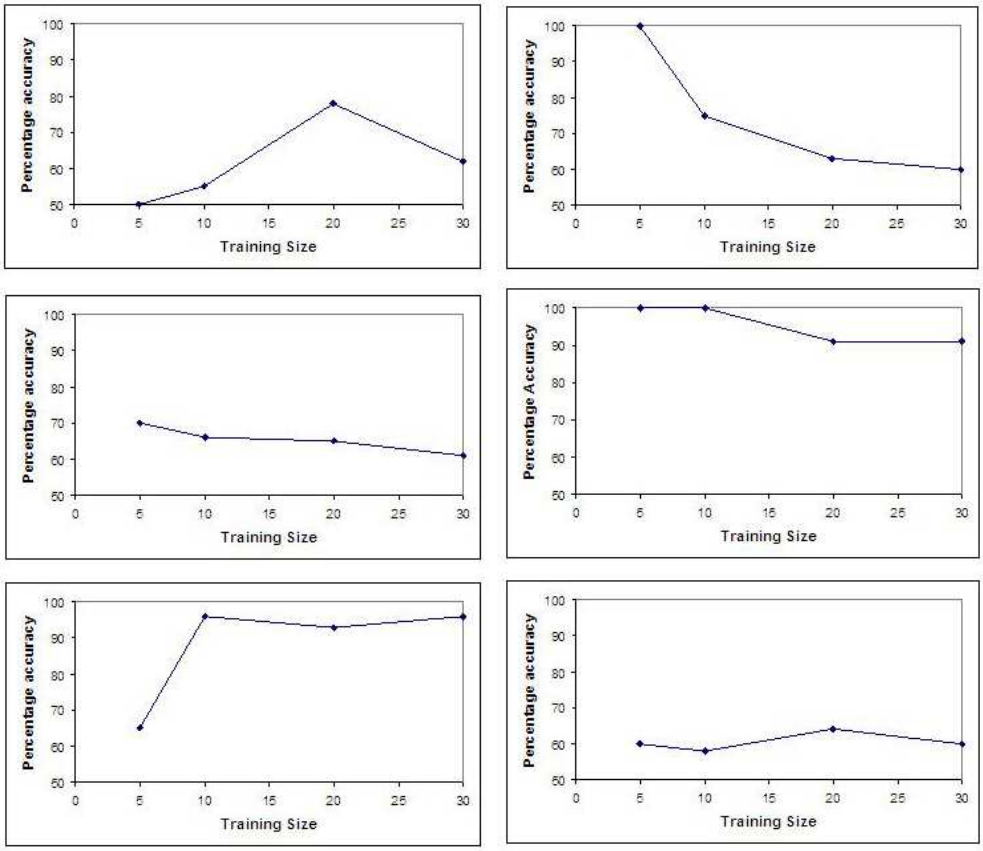


*Fig. 5.15:* **Graphs showing accuracy of classification on the test sets as a function of the training set size**

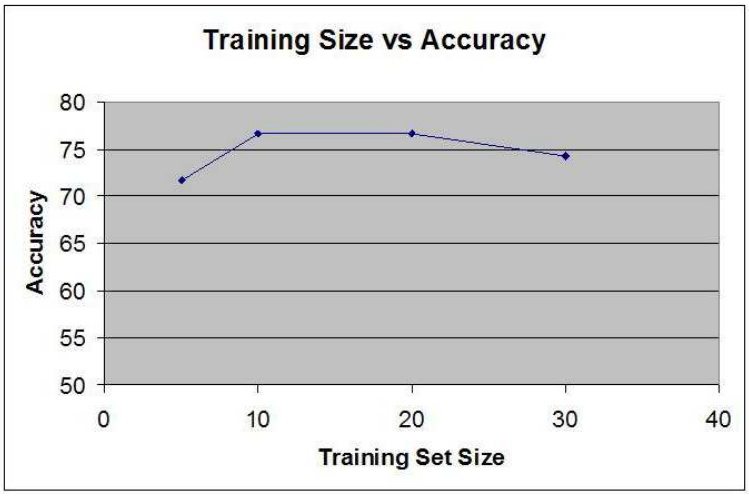


*Fig. 5.16:* **Graph showing the size of the training set used in learning vs accuracy averaged over collected data**

The training set size results provide the most difficult results to interpret out of all of the parameters we seek to optimise. In general when learning parameters of models more training data is considered better. It is difficult to determine from the results the optimal training set size. Hence in the absence of specific information to the contrary we default to the "common sense" approach of using as much data as possible for learning parameter values. Hence the largest training set available was used for all further models trained.

### *5.2.5 Cell set selection*

In order to use this system on a large scale, it is necessary to choose the cells in an automated fashion. I exclude cells which touch the edge of the image, since only part of these cells will typically be visible. There is also an issue of whether to exclude cells which have overlapping regions of interest. Including cells with overlapping regions could bias the results since this means part of the data in the image is being over-represented. However, given the way the models are built it is a reasonable argument to say that the data is different, since despite being the same pixel data, those pixels are found at a different radial distance from a nucleus. In an ideal situation we would not need to invoke this argument and could simply exclude overlapping cells, since this ensures that parts of the data are not being reused and unduly emphasised. However, in many instances images have few cells remaining after pre-processing, and so the best option is to retain overlapping cells on the basis that data duplication is minimal given the model construction. This provides more data for parameter estimation, which in general is necessary given the number of parameters to be estimated and the typical number of training images available.

## *5.3 Binary classification results: Comparison with Support Vector Machines*

The binary classification performance of blueprints and Support Vector Machines (SVMs) is described in this section. The aim is to find out how well the two approaches can perform binary classification on the same datasets. The following sets are indicative of the classes of training and test cells were used. In each case the first 20 training cell regions are shown.

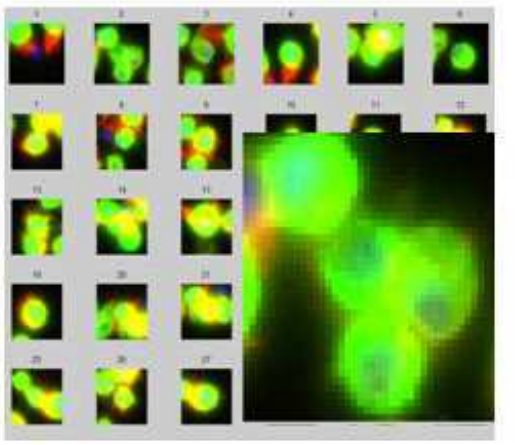

*Fig. 5.17:* **Talin knockout cells:** "Set1". The whole set is larger than shown and was split into disjoint training and test sets. A number of cells are shown and a typical cell region is shown enlarged in the bottom right corner.

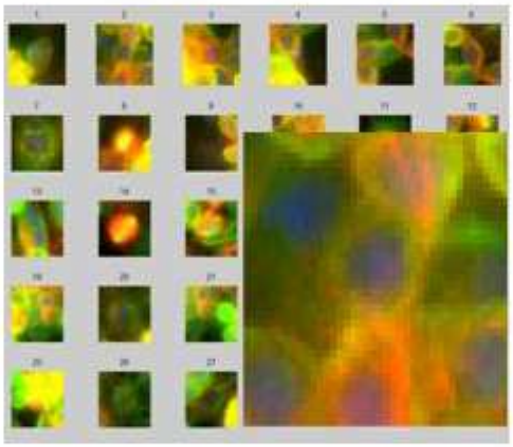

*Fig. 5.18:* **Rac knockout cells:** "Set2". The whole set is larger than shown and was split into disjoint training and test sets. A number of cells are shown and a typical cell region is shown enlarged in the bottom right corner.

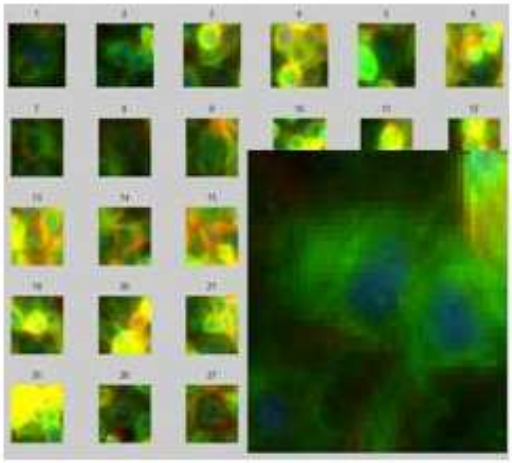

*Fig. 5.19:* **Sos knockout cells:** "Set3". The whole set is larger than shown and was split into disjoint training and test sets. A typical cell region is shown enlarged in the bottom right corner.

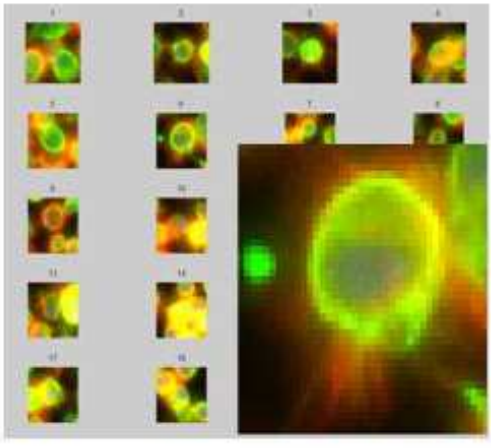

*Fig. 5.20:* **Mys knockout cells:** "Set4". The whole set is larger than shown and was split into disjoint training and test sets. A typical cell region is shown enlarged in the bottom right corner.

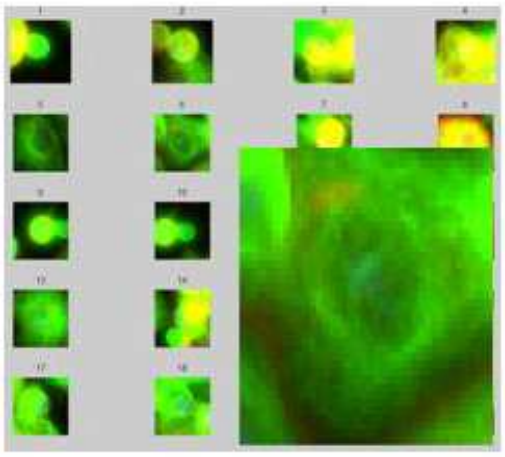

*Fig. 5.21:* **Cdc16 knockout cells:**"Set5". The whole set is larger than shown and was split into disjoint training and test sets. A typical cell region is shown enlarged in the bottom right corner.

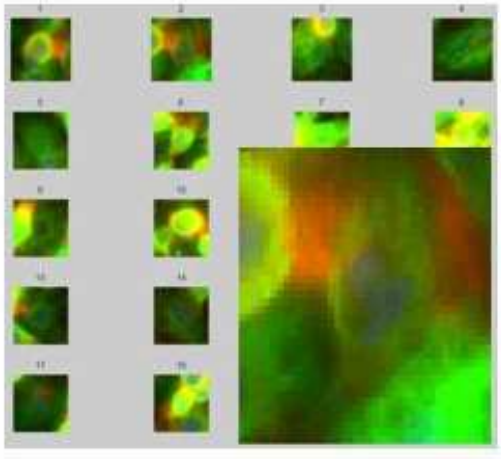

*Fig. 5.22:* **Wildtype cells from plate 1:** "Set6". The whole set is larger than shown and was split into disjoint training and test sets. A typical cell region is shown enlarged in the bottom right corner.

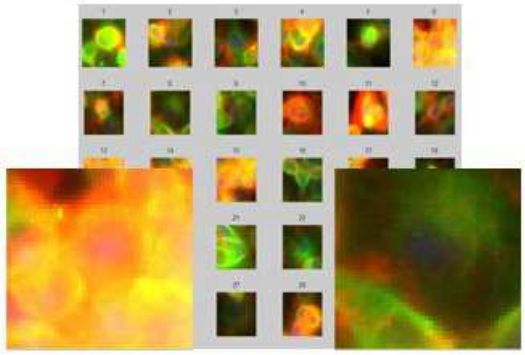

*Fig. 5.23:* **Sample wildtype cells from the control wells across all 58 plates:** "Set7". The whole set is larger than shown and was split into disjoint training and test sets. Sample cell regions are shown enlarged in the bottom right and left corners. Note the high degree of variation in this set

The freely available Support Vector Machine Matlab package osu-svm, which is based on LIBSVM [14] was used, and additional functions were written as necessary. The Support Vector Machines (SVMs) were trained on the vectorised pixel intensity data. The SVMs were tuned in the following manner. Support Vector Machines have been reported to work best for values between 0 and 1, so the pixels values were first re-scaled to lie between 0 and 1, by finding a global minimum and global maximum value over all training examples of both classes, and then subtracting the minimum and dividing by the maximum. A radial basis function was used. The gamma and cost value were chosen using a grid search and cross validation: the training data was split into four parts. The parameters were then learnt using three of these parts and tested on the fourth. Each segment of the training data was excluded in turn. The gamma and cost value that gave the best averaged classification were selected to be used. Once the gamma and cost value were chosen the SVM was trained using all of the training data using these values. Finally the classification of test data was found using the trained SVM. The percentage accuracy could be found since ground truth was known for the selected samples.

Note that the same training data was used for training both blueprint models and Support Vector Machines. The test set, disjoint from the training set, was also the same for both Blueprint and Support Vector Machine classifications.

The tables below show the results of the binary comparison. For Tables 5.24 & 5.25, the entry in the ith row and jth column contains the average binary classification accuracy of class i cells and class j cells averaged across both classes and all test cells. This table is symmetric.

| Phenotype | Talin | Rac | Sos | Mys | Cdc16 | WT1 | WTG |
|---|---|---|---|---|---|---|---|
| Talin | 0 | 92 | 64 | 88 | 78 | 57 | 98 |
| Rac | 92 | 0 | 56 | 73 | 89 | 62 | 98 |
| Sos | 64 | 56 | 0 | 60 | 67 | 56 | 68 |
| Mys | 88 | 73 | 60 | 0 | 85 | 54 | 98.5 |
| Cdc16 | 78 | 89 | 67 | 85 | 0 | 69 | 97 |
| WT1 | 57 | 62 | 56 | 54 | 69 | 0 | 99 |
| WTG | 98 | 98 | 68 | 98.5 | 97 | 99 | 0 |

*Fig. 5.24:* **Blueprint Results Table** Table of averaged binary classification accuracies of row phenotype vs column phenotype

| Phenotype | Talin | Rac | Sos | Mys | Cdc16 | WT1 | WTG |
|---|---|---|---|---|---|---|---|
| Talin | 0 | 89 | 90 | 92 | 83 | 85 | 87 |
| Rac | 89 | 0 | 59 | 69 | 89 | 88 | 80 |
| Sos | 90 | 59 | 0 | 86 | 87 | 88 | 63 |
| Mys | 92 | 69 | 86 | 0 | 96 | 97 | 95 |
| Cdc16 | 83 | 89 | 87 | 96 | 0 | 100 | 80 |
| WT1 | 85 | 88 | 88 | 97 | 100 | 0 | 77 |
| WTG | 87 | 80 | 63 | 95 | 80 | 77 | 0 |

*Fig. 5.25:* **SVM Results Table** Table of averaged binary classification accuracies of row phenotype vs column phenotype

The average binary classification accuracy over the whole data set for Support Vector Machines was 84.8%. The average binary classification accuracy for Blueprints was 76.6 %. These results show that overall the classification accuracy of SVMs is better than that attained using blueprint models. However, there are specific datasets on which SVMs considerably outperformed blueprints, and also on which blueprints outperform SVMs. This becomes clear if we look at the difference table SVM accuracy minus blueprint accuracy, shown below in Table 5.26.

| Phenotype | Talin | Rac | Sos | Mys | Cdc16 | WT1 | WTG |
|---|---|---|---|---|---|---|---|
| Talin | 0 | -3 | 26 | 4 | 5 | 28 | -11 |
| Rac | -3 | 0 | 3 | -4 | 0 | 26 | -18 |
| Sos | 26 | 3 | 0 | 26 | 20 | 32 | -5 |
| Mys | 4 | -4 | 26 | 0 | 11 | 43 | -3.5 |
| Cdc16 | 5 | 0 | 20 | 11 | 0 | 31 | -17 |
| WT1 | 28 | 26 | 32 | 43 | 31 | 0 | -22 |
| WTG | -11 | -18 | -5 | -3.5 | -17 | -22 | 0 |

*Fig. 5.26:* **SVM Blueprint difference table** The difference of SVM results and the blueprint results are shown (SVM - blueprint)

This table shows that SVMs considerably outperformed blueprints on the comparisons involving Sos cells (row/column 3) and wildtype cells from plate 1 (row/column 6). Further insight into the reason for this can be gained by looking at the non-symmetric results table for blueprints, in which in the i-jth entry consists the classification accuracy of class i cells when compared to class i and class j models. These results are shown in Table 5.27

| Phenotype | Talin | Rac | Sos | Mys | Cdc16 | WT1 | WTG |
|---|---|---|---|---|---|---|---|
| Talin | 0 | 92 | 98 | 78 | 94 | 98 | 98 |
| Rac | 92 | 0 | 100 | 52 | 98 | 96 | 98 |
| Sos | 30 | 12 | 0 | 20 | 34 | 36 | 36 |
| Mys | 98 | 94 | 100 | 0 | 100 | 100 | 100 |
| Cdc16 | 62 | 80 | 100 | 70 | 0 | 90 | 100 |
| WT1 | 16 | 28 | 76 | 8 | 48 | 0 | 100 |
| WTG | 98 | 98 | 100 | 97 | 94 | 98 | 0 |

*Fig. 5.27:* **Blueprint Non-symmetric Results Table** Table of binary classification accuracies of row phenotype vs column phenotype (values show the accuracy obtained for row i cells determined using the row i model and the column j model.)

The reason that this table is non-symmetric is that in some instances the model learnt for a particular class is poor at "claiming" cells from its own class. This is notable in rows 3 and 6 in which the models learnt for Sos and wildtype plate 1 do very poorly at describing cells from their own dataset better than other models. It is possible that in these cases the training data was poor at generating models which adequately described the intended class. A possible remedy for this would be to perform threshold tuning as in Section 5.1.2. An alternative would be to explore the use of finding the probability of the training data of each model. This number could be used to offset the effect of models which poorly describe their own data.

Blueprints considerably outperformed SVMs on comparisons involving wildtype cells taken from across the genome wide screen (row/column 7). The set containing wildtype cells from across the genome wide screen is the data set that contains the most variability. The ability of blueprints to perform well on this dataset suggests that the model type is better suited to modeling weakly related data.

# 6. DISCUSSION

## *6.1 Summary*

In this thesis we have examined techniques of interpreting cells images. We have explored the advantages and disadvantages of human analysis, and of existing automated techniques. I have explained the way different models link together in terms of a natural progression from simple models. In Chapter 4 a novel method using a novel generative model by extending the concept of mixture of Gaussian models. I have demonstrated how this can be used for cell recognition, and I have achieved comparable performance on binary classifications as Support Vector Machines.

The specific model described here has potential, and many further developments to it can be envisaged. The major advantage of the model developed here is the flexibility it allows. The major disadvantage is the high dimensionality of the models and the comparative scarcity of data of any given phenotype. Possible extensions to the methods used in this thesis are discussed in the section below.

In addition to the specific model development, overall the thesis gives a framework of the type of system that can be developed in order to analyse genome wide screens. Chapter 3 shows a number of the problems that frequently need to be addressed in the course of genome wide analysis, and provides a basic solution to each of these problems in order to allow further analysis of the data. The framework presented is modular in nature, so any particular component could be extended or replaced, as appropriate, to improve the results. The most obvious part of the process that would benefit from an extended analysis is the identification of cell regions. Segmentation is a burgeoning topic, but received a cursory treatment here since the focus was the development of classification models. Some possible extensions to the methods used in the thesis are described in the next section.

As well as extensions to the methods, there are also additional stages that can be added to provide genome wide analysis. Suggestions about how this could be approached are given in the following section.

## 6.2 Future Work

There are a number of possible ways of extending and improving this work. They fall broadly into two categories:

- Improving the segmentation or the models
- Extending the existing recognition method to classify large datasets such as our RNAi screen [40].

Each of these is a large body of work in its on right. Improving the segmentation so that it precisely and accurately finds entire cells is a highly desirable goal in its own right, but would also help all downstream components of a classification system. A number of methods could be explored including:

(a) Designing data specific filters which work well on a given data set.

(b) Modifications of the watershed algorithm, for example the propagation method [13]

(c) Modifying the experimental technique to provide images which are more straightforward to segment. A possible way of doing this would be to use stochastic labeling. This results in only a small proportion of cells being fluorescently tagged and hence cells only need to be differentiated from a black background in order to segment them.

(d) There are many possible additions to the model that could be made. These include variable patch shape [35], the addition of a multi-scale approach, and clearly also the data input to the models could be varied to include calculated image features.

(e) As well as different phenotypes, images will, in general, contain cells in different stages of the cell cycle. In the cell line we are examining on average about ninety five percent of cells will be in interphase and can be considered to be in an equivalent stage. Around five percent of cells will be in mitosis. The question of how to handle mitotic cells in the training and test sets depends on whether the task is to classify the phenotype only or whether to also classify the cell cycle of each cell. In the former case, mitotic cells can be left in both the training and test sets. A model learnt from a training set containing mitotic cells will simply have some small region of the model which describes mitotic cells. In the latter case, two models for each phenotype could be learnt: one for interphase cells, and one for mitotic cells. The first stage in learning these models would be to separate cells into mitotic or not mitotic. This could be done by reference to a standard mitotic cell model against a standard cell model. Since mitotic and interphase cells are quiet different, this should still be accurate even in cases where the phenotypes being compared do not appear very similar to the wildtype models. That is to say the difference in

appearance between mitotic cells and non-mitotic cells is usually greater than the difference between phenotypes.

Without any of the above extensions, the existing work could be used to provide a framework for the classification of an entire large dataset. This could be done as follows:

1. Choose ten "bank" phenotypes.
2. Create a training set of cell regions for each bank phenotype (Chapter 3).
3. Learn a blueprint model for each bank phenotype (Chapter 4).
4. For each image in the set extract as many cell regions as possible, and find the likelihood of each bank phenotype given each cell region (Chapters 3 and 4).
5. Define a ten dimensional score vector for each image as the normalised summation of the likelihood scores found in the previous step for each cell region of that image.
6. Cluster the images by their score vector, e.g. by Euclidean distance.

The proposed process is shown in figure 6.1.

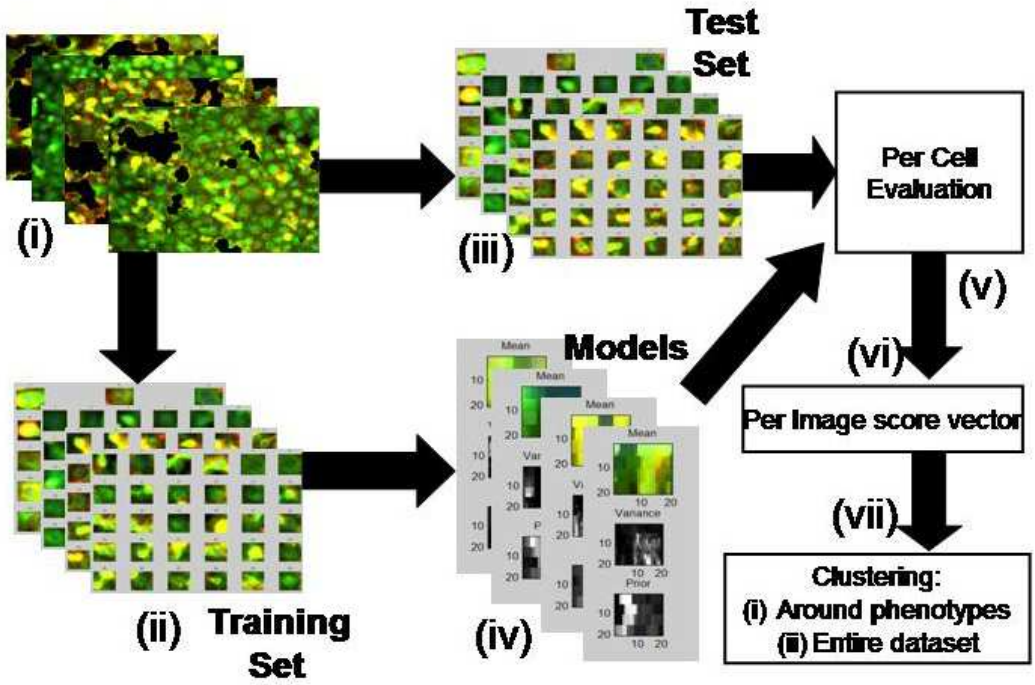


*Fig. 6.1:* **Proposed process for clustering a genome wide image data set on the basis of appearance.** (i) Experimental data from known genes, after normalisation and pre-processing. (ii) & (iii) The data for the chosen genes is split into test and training data. (iv) The training data is used to train some models. (v) The test data is given a score vector using the learnt models. (vi) The score vector for each image in the rest of the set is found. (vii) The images are clustered by clustering their score vectors

The described process could be used to provide an approximate classification for every image in the dataset. This is done by placing each image at some position in the ten dimensional space defined by the ten bank phenotypes. It should be noted that the choice of ten phenotypes is an informed guess as to how many dimensions might be required, but can be considered as an arbitrary number. Consideration of this number yields insight into the nature of the problem that needs to be solved. The classification of images into phenotypes is a problem consisting of two parts. The first is to define the phenotypic space. In particular the dimensionality of the space must be determined. The next part of the problem is to define a "correct" function which maps images into the phenotypic space. Each problem is a difficult problem in its own right. Determination the dimensionality of "phenotypic space" is a non-trivial task. Traditionally it has been approached by defining phenotypes based on characteristics such as cell area, actin content, nuclear area, mitotic index, etc. However, this really is dodging the underlying question. The underlying question is similar to finding the probability of the mappings in the models described in Chapter 4. That is, the variable we would like to know about is a hidden variable, the value of which we must infer via observable data such as nuclear area etc. An analogy is to consider trying to tell the difference between an ambulance and a fire engine. There are many possible parameters that we could measure about each vehicle: colour, size, engine characteristics, number of wheels etc. However, all of these properties flow from a binary categorisation as fire engine or ambulance. This contrived example may seem absurd, but it serves to illustrate the point that a wealth of observable data can sometimes be a hinder rather than help. Classifying data into classes in situations where the number and characteristics of the classes are unknown is an active area of research. The above proposed scheme is a way to use existing models and understanding to give a first pass at obtaining useful biological results.

A useful modification to the above procedure is to replace this with an iterative procedure, whereby instead of defining an initial bank of phenotypes, the bank is grown iteratively. Starting initially with only a wildtype model, all images in the set are scored and then the lowest scoring image is used to define a new model. This procedure can then be repeated, adding a new model at each stage. This has the advantage of not requiring an initial definition of "interesting" phenotypes. This scheme is an indirect way of solving the problem of defining phenotypic space, discussed above. It makes no attempt at defining the dimensions of phenotypic space, but by iteratively adding dimensions it aims to use only as many dimensions as "necessary" to define a space in which to perform the classification task.

Neither of the above procedures in outline need necessarily use generative models. The requirement is that there is some method of comparing the similarity of images. Support vector machines could be used for this purpose, for example. Alternatively the response from specifically chosen filters could also be used in order to "score" images. The question as to which of these is "right" is likely to have to be judged by the classification results that they yield. Any scoring mechanism will necessarily be dependent on the choice of initial segmentation method.

These procedures would produce a classification of the images in the dataset. How to evaluate the quality of such a classification is far from clear. The true clustering of the images is unknown, and arguably, may be necessarily subjective. However, any such classification of a large dataset could yield important information for biologists and give an initial picture of how genes interact and cluster for a given function such as cell morphology. This type of approach and result is typical of systems wide biological studies. Many identified genes in organisms are putative genes that have been predicted by computer simulation, with predicted coding regions. Nonetheless these estimates provide sufficiently good approximations to enable large scale experiments such as a genome wide RNAi screen. If we judge the end goal to be added biological knowledge then RNAi screens, despite being in their infancy, have certainly already produced excellent results [9, 56].

This thesis develops a classification procedure based on patches of images. The classification problem is just one component of the overall analysis problem. This thesis demonstrates how the classification system developed here can be used, by providing a system for processing, identifying and extracting cell regions from cell images obtained in a genome wide screen. Additionally, this thesis gives a sketch map of the overall analysis challenge for interpretation of large scale biological screens. This area of research is likely to become more important and more widely studied in future as the scope of biological experiments and the requirement for computational analysis increases yet further.

# APPENDIX

# A. UNSUPERVISED LEARNING & THE EXPECTATION MAXIMISATION ALGORITHM

Unsupervised learning algorithms are designed to cluster data according to some similarity measure, and do not require labeled data to operate. The Expectation Maximisation [21](EM) algorithm is widely used, and has the desirable property of guaranteed convergence.

The EM algorithm is an unsupervised learning technique for maximising, iteratively, the probability of some given data under a model with respect to the model parameter values. The credit for the EM algorithm as a complete process is usually attributed to Dempster, Laird and Rubin in their seminal 1977 paper [21]. The ideas employed by the algorithm were discussed as early as 1958 by Hartley, and the development of the algorithm can be seen as a gradual process rather than a sudden act of creation. Here, the EM algorithm is used to optimise model parameters given training data.

## *A.1 Formulation of the problem and overview of approach*

The aim of the EM algorithm is to find a maximum likelihood estimate for some model parameters, $\Theta$, given some data, D:

$$\arg\max_{\Theta} P(\Theta|D) \tag{A.1}$$

However, this is done in the presence of hidden or "nuisance" variables so that this probability can not be directly evaluated. The EM algorithm approximates a solution to the maximisation problem. Specifically it finds the parameter vector which locally gives a maximum posterior probability of the parameters. This can be found by marginalising over the hidden variables:

$$\arg\max_{\Theta} \left[ \sum_{H \in \mathbb{H}} P(\Theta, H|D) \right] \tag{A.2}$$

where D, H and $\Theta$ are data, hidden variables and parameters respectively. The algorithm differs from gradient ascent since rather than finding a best H at each iteration, a distribution over the hidden variables is calculated. Assuming the data has been independently sampled the above equation is equivalent to

maximising the joint probability:

$$\arg\max_{\Theta} \sum_{H \in I\!H} P(D, H, \Theta) \tag{A.3}$$

As an example of the distinction between a hidden variable and a parameter, hidden variable optimisation can be thought of as optimally assigning data to a cluster, while parameter optimisation relates to optimally determining the shape of the cluster according to some model.

Since very small numbers are frequently involved and algorithms are intended to be run on computers, it is standard to take the logarithm of probabilities to reduce rounding errors associated with machine representation accuracy. Since logarithm is a monotonic function the argument of the maximum of the logarithm of the probability is identical to the argument of the maximum of the probability. Hence the problem to be solved is:

$$\arg\max_{\Theta} log \sum_{H \in I\!H} P(D, H, \Theta) \tag{A.4}$$

The algorithm proceeds in two steps. The Expectation step (E step) calculates the expected probability distribution of the data and the hidden variables given the current parameters. The maximisation step (M step) maximises this distribution with respect to the parameters. This process is iterated. The E step can also be interpreted as finding a lower bound for the joint probability [20].

In the E step the following distribution is evaluated:

$$P(H|D, \Theta) \tag{A.5}$$

In the M step the probability of the parameters is maximised:

$$\Theta_{new} = arg\max P(\Theta|H, D) \tag{A.6}$$

The term q(H) is frequently written in the literature to mean $< \log P(D, H|\Theta) >$. This is the log likelihood expectation of the parameters.

### *A.1.1 Step 1: Expectation step (Calculating a lower bound)*

We want to calculate:

$$logP(D, \Theta) = log \sum_{H \in I\!H} P(D, H, \Theta) \tag{A.7}$$

The logarithm of a sum is difficult to deal with algebraically, so instead the above is reformulated as a sum of logarithms. This can be done by making

use of Jensen's inequality for convex functions:

**Lemma: Jensen's Inequality**

Given a convex function f, and a set of parameters $\lambda_i$ over some indexing set I, with the property that $\lambda_i \leq 1 \forall i \in I$ & $\sum_{i \in I} = 1$ then:

$$f(\sum_{i \in I} \lambda_i x_i) \leq \sum_{i \in I} f(\lambda_i x_i) \tag{A.8}$$

So we can bound A.7 by:

$$B = \sum_{H \in \mathbb{H}} \lambda_i \log P(D, H, \Theta)/\lambda_i \leq \log \sum_{H \in \mathbb{H}} \lambda_i P(D, H, \Theta)/\lambda_i \tag{A.9}$$

Where the $\lambda_i$ form a probability distribution (the important point being that they sum to unity). This gives *a* bound for the objective function, but ideally we want to find a good bound.

The best lower bound can be shown to be attained if we choose our probability distribution of the $\lambda_i$s to be P(H $|D, \Theta)$. By noting that $\frac{P(D,H,\Theta)}{P(H|D,\Theta)} = P(D, \Theta)$, we can confirm that this choice does in fact touch the objective function:

$$\sum_{H \in \mathbb{H}} P(H|D, \Theta) \log P(D, H, \Theta)/P(H|D, \Theta) = \log P(D, \Theta) \sum_{H \in \mathbb{H}} P(H|D, \Theta) \tag{A.10}$$

Noting that $\sum_{H \in \mathbb{H}} P(H|D, \Theta) = 1$, gives the desired result.

### *A.1.2 Step 2: Maximisation step*

Once the expected distribution has been found, then the joint log likelihood is maximised over the parameters of the model:

$$\Theta_{t+1} = arg \max_{\Theta} B \tag{A.11}$$

The method of finding the parameter values which maximise the bound depends on the problem. For problems which yield a bound that is differentiable with respect to the parameters, then a maximum can be found using ordinary calculus. If it is not then some other method, such as gradient ascent must be employed to estimate the maximum.

# BIBLIOGRAPHY


[1] V Ambros, "The functions of animal microRNAs", *NATURE* Vol 431 September 2004

[2] H Ancin, TE Dufresne, GM Ridder, JN Turner, B Roysam, "An Improved Watershed Algorithm for Counting Objects In Noisy Anisotropic 3D Biological Images" *IEEE International Conference on Image Processing Proceedings* 1995

[3] C Bakal, J Aach, G Church, N Perrimon, "Quantitative Morphological Signatures Define Local Signaling Networks Regulating Cell Morphology" *Science* 316, September 2007

[4] P Bamford, B Lovell, "A Water Immersion Algorithm For Cytological Images Segmentation" *APRS Image Segmentation Workshop* December 1996

[5] B Baum, G Craig, "RNAi in a postmodern, postgenomic era", *Oncogene* 23, 2004

[6] J Besag, "On The Statistical Analysis Of Dirty Pictures" *Journal Of The Royal Statistical Society* B48

[7] CM Bishop, "Pattern Recognition And Machine Learning" *Springer* 2006

[8] DM Blei, AY Ng, MI Jordan "Latent Dirichlet Allocation" *The Journal of Machine Learning Research archive* Volume 3, March 2003

[9] M Boutros, LP Brs, W Huber, "Analysis of cell-based RNAi screens", *Genome Biology* Vol 7:R66 July 2006

[10] J Campbell, J Slater, J Gillespie, I Bendezu, F Murtagh, "Pattern Recognion Methods for Identification of Shellfish Larvae", *Proceedings Of The Irish Machine Vision And Image Processing Conference* August 2005

[11] L Carrivick, S Prabhu, "Deriving a Hierarchical Representation of Lung Disease using Re-Sampling Mixture Models" *Proceedings Of The 9th Annual Conference On Medical Image Understanding and Analysis* July 2005

[12] M Lamprecht, DM Sabatini, AE Carpenter "CellProfiler: free, versatile software for automated biological image analysis" *BioTechniques*42, January 2007

[13] TR Jones A Carpenter and P Golland, "Voronoi-Based Segmentation of Cells on Image Manifolds", *ICCV Workshop on Computer Vision for Biomedical Image Applications*, 2005

[14] CC Chang, CJ Lin, LIBSVM: a library for support vector machines, 2001, software available at *www.csie.ntu.edu.tw/ cjlin/libsvm*

[15] F Cloppet, JM Oliva and G Stamon, "Angular Bisector Network, a Simplified Generalized Voronoi Diagram: Application to Processing Complex Interactions in Biomedical Images" *IEEE Transactions On Pattern Analysis And Machine Intelligence* Vol.22 No.1 Jan 2000

[16] TF Cootes, CJ Taylor "Statistical models of appearance for medical image analysis and computer vision" Proc. SPIE Medical Imaging, 2001, ISSU 4322; Part 1

[17] O. Cuisenaire and B. Macq, "Fast Euclidean Distance Transformation by Propagation Using Multiple Neighborhoods" *Computer Vision and Image Understanding* Vol. 76, No. 2, November, 1999

[18] O Cuisenaire, "Distance Transformations: Fast Algorithms and Applications to Medical Image Processing", *Ph.D thesis*, Oct. 1999

[19] PE Danielsson "Euclidean distance mapping" *Computer Graphics and Image Processing*, 14, 1980

[20] F Dellaert, "The Expectation Maximization Algorithm", *Technical Report number GIT-GVU-02-20*, February 2002

[21] AP Dempster, NM Laird, DB Rubin, "Maximum Likelihood from Incomplete Data via the "EM" Algorithm", *Journal of the Royal Statistical Soceity. Series B(Methodological)*, Vol.39, No.1, 1977

[22] JS Duncan, N Ayache, "Medical Image Analysis: Progress over Two Decades and the Challenges Ahead", *IEEE Transactions On Pattern Analysis And Machine Intelligence* Vol.22 No.1 Jan 2000

[23] B Fang, W Hsu, ML Lee, "On the Accurate Counting of Tumour Cells" *IEEE Transactions of Nanobioscience* Vol 2 No 2 June, 2003

[24] A Fire, S Xu, MK Montgomery, SA Kostas, SE Driver, CC Mello, "Potent and specific genetic interference by double stranded RNA in *Caenorhabditis* elegans" *Nature*, Vol 391, 19 February 1998

[25] W Freeman, EC Pasztor, OT Carmichael, "Learning Low-Level Vision" *International Journal of Computer Vision* 40(1), 2000

[26] BJ Frey, N Jojic, "Transformation invariant clustering and dimensionality reduction" *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2001

[27] Flockhart, M Booker, A Kiger, M Boutros, S Armknecht, N Ramadan, K Richardson, A Xu, N Perrimon, B Mathey-Prevot, "FlyRNAi: the Drosophila RNAi screening center database" *Nucleic Acids Res* 34, 2006

[28] J Goldberger, S Gordon, H Greenspan "An Efficient Image Similarity Measure Based on Approximations of KL-Divergence Between Two Gaussian Mixtures" *Proceedings of the Ninth IEEE International Conference on Computer Vision* 2003

[29] WEL Grimson, DP Huttenlocher, "On The Sensitivity of the Hough Transform for Object Recognition" *IEEE Computer Vision Second International Conference on*, 1998

[30] GE Hinton, Z Gharamani, "Generative models for discovering sparse distributed representations" *Royal Society Phil Trans* B352, 1997

[31] T Hofmann, "Probabilistic Latent Semantic Indexing", *Proceedings of the Twenty-Second Annual International SIGIR Conference* 1999.

[32] MB Jeacocke, BC Lovell, "A Multi-Resolution Algorithm For Cytological Image Segmentation", *Intelligent Information Systems,1994. Proceedings of the 1994 Second Australian and New Zealand Conference on*, November 1994

[33] N Jojic, BJ Frey and A Kannan, "Epitomic Analysis of Appearance and Shape," In Proc. *International Conference on Computer Vision*, Vol. 1, 2003

[34] MI Jordan, RA Jacobs, "Hierarchical mixtures of experts and the EM algorithm", *Proceedings of International Joint Conference on Neural Networks* 1993

[35] A Kannan, J Winn, C Rother, "Clustering appearance and shape by learning jigsaws" *Advances In Neural Processing Systems* 2007

[36] A Kapoor, S Basu, "The Audio Epitome: A New Representation For Modeling And Classifying Auditory Phenomena" *Proceedings of IEEE International Conference On Acoustics, Speech and Signal Processing*, Vol 5, March 2005

[37] J Kittler, J Illingworth, "Minimum error thresholding" *Pattern Recognition* Volume 19, Issue 1 1986

[38] AK Jain, RPW Duin, J Mao, "Statistical pattern recognition: a review," *IEEE Transactions On Pattern Analysis And Machine Intelligence* Vol.22, no.1, Jan 2000

[39] V Kolmogorov, R Zabih, "What energy functions can be minimised via graph cuts?" *IEEE Transactions On Pattern Aanalysis And Machine Intelligence* VOL. 26, No. 2, 2004

[40] AA Kiger, B Baum, S Jones, MR Jones, A Coulson, C Echeverri and N Perrimon "A functional genomic analysis of cell morphology using RNA interference" *Journal Of Biology* 2:27, Oct 2003

[41] A Lindbo, WG Dougherty "Plant Pathology And RNAi: A Brief History" *Annual Review of Phytopathology* Vol. 43, September 2005

[42] DG Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision* 2004

[43] S Lucey, T Chen "Learning Patch Dependencies For Improved Pose Mismatched Face Verification" *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol 1, 2006

[44] DJC MacKay, "Information, Inference And Learning Algorithms", *Cambridge University Press* 2003

[45] N Malo, JA Hanley, S Cerquozzi, J Pelletier R Nadon, "Statistical practice in high-throughput screening data analysis", *Nature Biotechnology* Vol 24, No 2 Febraury 2006

[46] N Malpica, CO de Solorzano, JJ Vaquero, A Santos, I Vallcorba, JM Garcia-Sagredo, F del Pozo, "Applying Watershed Algorithms to the Segmentation of Clustered Nuclei" *Cytometry* 28, 1997

[47] N Malpica, A Santos, A Tejedor, A Torres, M Castialla, P Garcia-Barreno, M Desco, "Automatic Quantification Of Viability In Epithelial Cell Cultures by Texture Analysis" *Journal Of Microscopy* Vol 209 Pt 1 January 2003

[48] AM Martinez, AC Kak, "PCA versus LDA" *IEEE Transactions On Pattern Analysis And Machine Intelligence* Vol 23, No 2, February 2001

[49] GJ McLachlan, KE Basford, "Mixture Models Inference And Applications To Clustering" *Marcel Dekker Inc* 1988

[50] K Mikolajczyk, B Leibe, B Schiele, "Local Features for Object Class Recognition" *Proceedings of the Tenth IEEE International Conference on Computer Vision* 2005

[51] TW Nattkemper, HJ Ritter, W Schubert, "A Neural Classifier Enabling High-Throughput Topological Analyis of Lymphocytes in Tissue Sections" *IEEE Transactions On Information Technology In Biomedicine* Vpol5, No2, June 2001

[52] Yoshikazu Ohya, Jun Sese, Masashi Yukawa, Fumi Sano, Yoichiro Nakatani, Taro L. Saito, Ayaka Saka, Tomoyuki Fukuda, Satoru Ishihara, Satomi Oka, Genjiro Suzuki, Machika Watanabe, Aiko Hirata, Miwaka Ohtani, Hiroshi Sawai, Nicolas Fraysse, Jean-Paul Latge, Jean M. Francois, Markus Aebi, Seiji Tanaka, Sachiko Muramatsu, Hiroyuki Araki, Kintake Sonoike, Satoru Nogami, Shinichi Morishita "High-dimensional and large-scale phenotyping of yeast mutants", *PNAS* December 27, 2005 vol. 102 No.52

[53] N. Otsu, "A Threshold Selection Method from Gray-Scale Histogram," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 8, 1978.

[54] NJ Pressman, "Markovian Analysis Of Cervical Cell Images", *The Journal Of Histochemistry And Cytochemistry* Vol 24 No1, 1976

[55] I Ragnemalm, "The Euclidean Distance Transformations In Arbitrary Dimenions", *International Conference on Image Processing and its Applications* 7-9, Apr 1992

[56] M Ramet, P Manfruelli, A Pearson, B Mathey-Prevot, R Alan, B Ezekowitz, "Functional genomic analysis of phagocytosis and identification of a Drosophila receptor for E. coli", *Nature* Vol 416 April 2002

[57] C Restif "Towards Safer, Faster Prenatal Genetic Tests: Novel Unsupervised, Automatic and Robust Methods of Segmentation of Nuclei and Probes" *Lecture notes in computer science*, Oxford Brookes, 2006

[58] JBTM Roerdink, A Meijster, "The Watershed Transform: Definitions, Algorithms and Parallelization Strategies", *Fundamenta Informaticae* 2000

[59] JC Russ, The Image Processing Handbook, Jul 2002, CRC Press Inc

[60] LI Rudin, S Osher, E Fatemi, "Nonlinear total variation based noise removal algorithms" *Physica* D 60, 1992

[61] TL Saito, Jun Sese, Yoichiro Nakatani, Fumi Sano3,4, Masashi Yukawa, Yoshikazu Ohya, Shinichi Morishita, "Data mining tools for the Saccharomyces cerevisiae morphological database" *Nucleic Acids Research*, 2005, Vol. 33

[62] DMU Sabino, LF Costa, EG Rizzatti and MA Zago, "Toward Leukocyte Recognition Using Morphometry, Texture and Color", *IEEE Biological Imaging: macro to nano* April 2004

[63] L Shafarenko, M Petrou, J Kittler "Automatic Watershed Segmentation of Randomly Textured Color Images" *IEEE Transactions On Image Processing* Vol 6 No 11 November 1997.

[64] PA Sharp, "RNAi and double-strand RNA," *Genes & Development*, 1999

[65] R Simpson, R Williams, R Ellis, and PF Culverhouse, "Biological pattern recognition by neural networks" *Mar. Ecol. Prog.* 1992

[66] D Sims, B Bursteinas, Q Gao, M Zvelebil, B Baum "FLIGHT: database and tools for the integration and cross-correlation of large-scale RNAi phenotypic datasets" *Nucleic Acids Research* 34: D479-D483 Jan 2006

[67] J Sivic, BC Russell, AA Efros, A Zisserman, WT Freeman, "Discovering objects and their location in images" *Proceedings of the International Conference on Computer Vision* 2005

[68] I Spadinger, SSS Poon, B Palcic, "Automated Detection and Recogntion of Live Cells in Tissue Using Image Cytometry", *Cytometry* 10, 1989

[69] M Spann, R Wilson, "A Quad Tree Approach To Image Segmentation Which Combines Statistical And Spatial Information" *Pattern Recognition* Vol 18 Nos 3/4, 1985

[70] Genjiro Suzuki Hiroshi Sawai Miwaka Ohtani Satoru Nogami Fumi Sano-Kumagai Ayaka Saka Masashi Yukawa Taro L. Saito Jun Sese Dai Hirata Shinichi Morishita Yoshikazu Ohya "Evaluation of image processing programs for accurate measurement of budding and fission yeast morphology" *Curr Genet* 49, 2006

[71] S Theodoridis, K Koutroumbas, "Pattern Recogntion Third Edition", *Elsevier* 2006

[72] JN Turner, H Ancin, DE Becker, DH Szarowski, M Holmes, N O'Connor, M Wang, T Holmes, B Roysam, "Automated Image Analysis Technologies for Biological 3D Light Microscopy" *Int Journal Syst Technol* 8, 1997

[73] V Vapnik, "Statistical learning theory" *Wiley and Sons Inc* 1998

[74] JJ Weinman, E Learned-Miller, "Improving Recognition of Novel Input with Similarity", *CVPR* 2006

[75] HS Wu, J Barba, J Gil, "Iterative Thresholding For Segmentation of Cells from Noisy Images", *Journal Of Microscopy* Vol 197 Pt3 March 2000

[76] HS Wu, J Gil, J Barba, "Optimal Segmentation of Cell Images" *IEE Proc Vis Image Signal Processing* —Vol 145, No 1, February 1998

[77] K Wu, D Gauthier, MD Levine, "Live Cell Image Segmentation", *IEEE Transcations On Biomedical Engineering* Vol 42 No1 Jan 1995

[78] J Yao, N Kharma, P Grogono, "A multi-population algorihtm for fast and robust ellipse detection" *Pattern Analysis Applications* 8, 2005

[79] M Yoshigi, EB Clark, and HJ Yost, "Quantification of Stretch-Induced Cytoskeletal Remodeling in Vascular Endothelial Cells by Image Processing" *Cytometry Part A* Vol 5, Sept 2003