# Table of Contents

Chapter 1: Introduction .......................................................... 1

Chapter 2: Project Background ................................................ 2
  Motivation ........................................................................... 2
  High content screening ...................................................... 2
  State of the art ................................................................. 2

Chapter 3: Neural Networks .................................................... 3
  What are neural networks? .................................................. 3
  Supervised learning ......................................................... 4
  Unsupervised learning ...................................................... 4

Chapter 4: Image Segmentation ................................................ 5
  Aims of image segmentation .............................................. 5
  Histogram based methods ................................................. 5
    Thresholding .................................................................... 5
  Region Growing Methods .................................................. 6
    Watershed algorithm ...................................................... 7
  Boundary based methods .................................................. 7
    Snake algorithm ................................................................ 8
  Pixel based methods ....................................................... 9
  Neural network based methods ......................................... 9
    Hierarchical self organising maps .................................... 10

Chapter 5: Working with CITU ............................................... 11
  What is CITU? ................................................................. 11
  Ensuring Portability ........................................................ 11
  Issues to be resolved ....................................................... 11
    Database connections ..................................................... 11
    Missing data .................................................................... 12
    Memory permissions ...................................................... 12
    Training of the neural network ....................................... 12
    Performing segmentation .............................................. 13

Chapter 6: Conclusions .......................................................... 15
  Research conclusions ....................................................... 15
  CITU conclusions ............................................................ 16

Acknowledgments and Bibliography .......................................... 17
Chapter 1: Introduction
The project, “Analysis of Cell Images”, is broken into 2 parts. The first part is research and learning based. The aim is to gain a greater understanding of the fields of image segmentation and neural networks. Once this has been accomplished the objective is to understand how these fields overlap and interact and what uses this interaction has.

The second part of the project is to become familiar with CITU, a Trinity College developed tool for segmentation and annotation of cell images. Once this has been achieved the aim was to continue the testing and development of the software.

Chapter 2 will discuss the motivation behind the project and the state of the art of cell image analysis.

Chapter 3 will cover the research of neural networks and both supervised and unsupervised learning systems.

Chapter 4 will look at Image segmentation and how it interacts with neural networks.

Chapter 5 will examine the second stage of the project, including a description of CITU and the changes made during the project.
Chapter 2: Project Background

Motivation:
The main motivation behind this project is to aid the development of high content screening and through that the testing and development in new drug treatments.

High Content Screening:
High Content screening is an automated cell biology method drawing on optics, chemistry, biology and image analysis to allow for rapid, biological research and drug discovery. Advances in microscopy now allow cell biologists to view events at sub-cellular levels. This includes cell count, cell mitosis and the effects of treatments on both healthy and diseased cells. The images are generated by running a set of well samples through an automated microscopy reader. According to BioMarket it is valued at between 5-10% of the $14 billion Biochemistry industry [1].

Until recently a small group of highly skilled and trained individuals would have to look over these images by hand and come to a conclusion themselves about the success of the treatment being monitored. Each session of HCS can generate up to 1200 cell images. Over the course of a year this can add up to as many as 13 million images [2]. This is too many images to reasonably examine manually which is why image segmentation and recognition software is so useful to the field. Unfortunately HCS images, due to the imaging method involved, contain large amounts of background noise. This makes the segmentation of cell images non trivial. How this is addressed will be discussed further in chapter 4.

State of the art:
Currently there are four main solutions in the HCS market, GE Health Care and Cellomics’ IN-Cell Analyser [3], Molecular Devices and Dannaher’s MetaXpress [4], Perkin Elmer’s Opera [5] and Media Cybernetics’ ImagePro [6].

With each system the user must perform 3 tasks. First the user must define the objects or features they are looking to extract from the images. Then they must set and validate a series of parameters for the for the image segmentation, object identification and classification, and feature classification. These include but are not limited to: the type of lens used to take the image, the viscosity of the immersion medium and setting the correct wavelength. After all this the user still has to visually mind the newly acquired data. This means they still have not removed the need for the expert analysis, only made their job easier.

Chapter 5 will discuss how CITU over comes the need for this large amount of user input and removes the need for the visual mining stage.
Chapter 3 Neural Networks:
What are neural networks?
Artificial Neural networks are computational models that attempt to simulate biological neural networks. They try to replicate the structure the human brain is believed to have. The network is made up of “neurons” or nodes, each one having a number of inputs and a single output. In general they are adaptive systems and are capable of changing structure depending on the information flowing through the nodes. This is what makes neural networks capable of “learning”. Learning in this case refers to the networks ability to find a solution to a task that is in some way optimal given a set of functions. The network determines which solution is optimal through the use of a cost function. A cost function is provided by the user on setup. The network will find a number of solutions to the task, then the cost of each solution is calculated and the solution with the lowest cost is deemed optimal.

Neural networks are split into layers. Usually the first layer is an input layer; each of these neurons will have a single input and a single output, although the output may be connected to multiple neurons. There may be any number of intermediate layers, each of which can have any number of inputs and a single output. Finally there is an output layer, this functions similarly to an intermediate layer but its output connects to outside the network. Neural network learning is broken down into two types; supervised learning and unsupervised learning [7].

Image showing a simple neural network model.
**Supervised learning:**
In supervised learning each input node is assigned a weight and a bias by the user. The weight may change but the bias is fixed. A set of inputs are run through the network, with a “teacher” reviewing the results. If the output is incorrect the weights of the nodes are updated according to a learning algorithm. This process is repeated until the weights do not change. Once this occurs the network has “learned” the function being run through it. [8]

The bias of a supervised learning network is a set of assumptions that the network uses to predict the outputs of the inputs it has not seen yet. Finding the correct balance of bias size is hugely important to a supervised learning network. If the bias is too weak the algorithm will learn different solutions for the same task for different sets of training data. However having a bias that is too strong makes the algorithm rigid and it may not be able to match each input to a correct output with one set of weights.

Occasionally it occurs that a task has excessive inputs. This is when there is a large enough set of inputs that it may stop the network from learning a correct solution. A solution to this is to try and find inputs that do not affect the outcome or only mildly influence it. By reducing its complexity it allows the network to find an optimal solution. Supervised learning is used to learn pattern recognition and regression. It is also usable for learning speech patterns and gestures.

**Unsupervised learning:**
In unsupervised learning the network only receives data inputs. It gets no indication of whether or not its outputs are correct. With no supervision the machines goal is to generate representations of the input that it can use for decision making, communicating its inputs to other networks or predicting future inputs. [9]

Unsupervised learning is a powerful learning tool. It allows networks to make estimations on the distribution of statistics. It can also be used to “teach” a network file compression and image clustering. The most common type of unsupervised learning network is the self organising map. Due to their significance to image segmentation they will be discussed in chapter 4.
Chapter 4 Image Segmentation:
Aims of image segmentation:
In the segmentation of cell images there are 4 main tasks that must be
accomplished.
First the program must “Detect” the cell in the image.

The software must then “Reject” the background of the image and remove any
other unwanted information contained within.

Then it must “Establish” the boundaries of the cells and contours or features it
has.

Finally the program must “Classify” the cell according to the type of cell
contained in the image.

Histogram based methods:
Histogram based methods are performed using the image histogram. This is a
plotting of the occurrence of a particular pixel measurement. This
measurement can be variation in grey levels, spatial properties or colour
changes across the pixels. The image is then divided into object. This is done
by merging each group of neighbouring pixels with the same pixel measurement
into a single object.

Thresholding:
Usually the colour or grey levels in the image must be “binarised”.
This is done using thresholding. This simplifies and compresses an image which
makes image segmentation significantly easier. Pixels are marked as an object
pixel if they exceed a given threshold and background pixels otherwise or vice
versa. Some of the most common techniques for finding the optimum threshold
are variance thresholding, iterative thresholding, peak and valley thresholding
and entropy thresholding.

Variance thresholding attempts to minimize the variance between the
partitions of the histogram. It is mainly used on images where the foreground
and background are similarly sized. Otsu’s method is a variant of this method
of thresholding [10].

Peak and valley thresholding uses peak analysis to detect the first terminating
and second initiating crossing of the “zero” point on the histogram, and uses
these to set the threshold. This method is most useful for images whose
histogram have more than one peak (bimodal or multimodal histograms) [11].

Iterative thresholding iteratively updates the threshold based on the mean grey
level in the partitions image histogram. This is repeated until there is no
change in the mean grey level. The technique is only useable when the object
and background have different grey levels. It is used in text enhancement
where the text will generally have a very different grey level to the
background colour [12].

Entropy thresholding calculates the entropy of the histogram. The threshold
used is either the value that maximises or minimises the entropy. It is used for
images where the foreground and background are reasonably different in size
[13].

When thresholding an image any number of threshold values may be
used. Despite this in most cases only one is chosen. This is due to two reasons. Firstly
some of the techniques, namely variance and iterative thresholding, are not
designed to find multiple thresholds, so multiple runs of the algorithms will
return the same value. The second reason is that, the most common use is
differentiating between the background and objects in the image, and only one
threshold is needed for the process.

An example of histogram based segmentation can be seen in chapter 5, figure
4.

Region growing:
Region growing methods segment images in one of two ways, growing and
merging, and splitting and merging.

Growing and merging techniques begin by taking in a number of points in the
image (seeds) along with the image to be segmented. The pixels neighbouring a
seed are compared to the seed using their intensity, the pixel with the closest
intensity is added to the seed creating a region. The process continues
comparing the mean intensity of the region to its neighbours until each pixel in
the image is a member of a region. Seeds are usually pixels with property that
makes them prominent such as highest or lowest grey level. In general growing
and merging is not suitable for cell image segmentation. This is because of
different sections within the cell, using grow and merge these are often split
into different regions. For accurate analysis the entirety of the cell must be
detected as a single region [14].

Splitting and merging works in the opposite direction. It begins with the entire
image as one region and each region is recursively split in two until no more
splits are possible. If a region is uniform it is bypassed in any further iteration.
If four of the child squares are uniform a merger is performed. This creates
regions that will also be bypassed and helps to prevent over segmentation from
occurring. Splitting and merging techniques are less object driven then growing
and merging, and “tends to produce boundaries consisting of long horizontal
and vertical segments”. Therefore it gives less useful results for the purposes
of cell image segmentation [14].

An example of region growing can be found in chapter 5, figure 4.
Watershed Algorithm:
One method of overcoming the short falls of region based methods is the watershed algorithm. The image is first converted into grey scale. The algorithm then uses the grey values of the pixels to create a topographical relief similar to a map. Each pixel’s value is treated as its elevation in the relief. The relief is then split into regions named “catchment basins” with each region being separated from the others by a watershed. A watershed is defined as a line where a drop of water falling on the line will have an equal chance of falling into any of the catchment basins it separates. The most popular method of watershed calculation is “flooding”. A water source is placed at pixels which are in local minimums of elevation. The relief is then “flooded” from these sources, and watershed lines are drawn where the flood waters from different sources would meet [14-15]. This method generates more natural region edges than the splitting and merging method. It also reduces the over segmentation of cells that occurs with grow and merge techniques although this may still occur with cells with strong boundaries or prominent nuclei.

Fig 1. Example of watershed algorithm on human brain scan (reprinted from [14]).

Boundary based techniques:
Two of the most popular boundary based segmentation techniques are edge detection and the snake algorithm (also known as the active contour model).

Edge Detection:
In an image the edges of an object are generally marked with a change of colour intensity. Intensity changes can also occur within an object but these are usually a smaller change than the ones at object boundaries. These large changes in intensity are found using the first and second derivatives of the intensity changes. The most prominent edges are then found using a threshold, similarly to the histogram methods. This will prevent the smaller changes mentioned above from being used as object edges.
There are a number of issues with this method however. Firstly it is highly sensitive to noise. This is not much of a difficulty for high contrast images, but in low contrast image where the intensity changes will not be as large it can cause serious problems. It can be overcome by using noise reduction techniques such as a median filter however.
Converting the images to grey scale can also result in reductions in the change of intensity between areas; therefore it is advisable to perform this technique on full colour images if they are available [16].

**Snake Algorithm:**
The snake algorithm works by adding an active contour to the image, which may be done manually. The contour is then deformed either by internal force pushing out or external forces pushing in. The deforming of a point on the contour stops when it contacts the boundary of an object, although the rest of the contour may continue to be deformed. The boundaries are recognised as the minimum of a function which is generated by the formulas for both the internal and external forces.
There are 2 common variants of the algorithm, the geographic model and the parametric model. The parametric model increases the speed of the algorithm by adding additional parameters such as extra functions for calculating external force or the addition of boundary maps.
Geometric models are capable of dealing with changes in the topology in the image more easily than parametric models. They can also deform multiple contours at once. However despite the potential to be parallelised the geometry models are generally much slower [17].

![Fig 2. Snake algorithm first (i) and final (ii) iterations on a drill bit gauge (reprinted from [17]).](image)
**Pixel based methods:**
Pixel based methods attempt to sort each pixel in an image into two groups; background and foreground. This is done by comparing the intensity levels of the pixels, either in the red/blue/green channels for colour images or grey level for grey scale images.

There are 3 major drawbacks to pixel based segmentation. Firstly, the algorithms do not take into account the location of the pixel within the image which can lead to pixels being placed incorrectly leading to blurred object boundaries.

The second drawback is that pixel based segmentations are much more sensitive to colour and light changes than humans. So much so that the technique may produce segmentation that is not usable directly for human analysis.

The third drawback is its extreme noise sensitivity. This can lead to over segmentation. It can be overcome by median filters or getting the average of blocks of neighbouring pixels [18].

Fig 3. Example of Pixel based segmentation (reprinted from [18]).

**Neural network segmentation:**
Neural networks, as described in chapter 3, can also be used to segment images. This is done using self learning self organising maps (SOM). An SOM is a type of artificial network, trained using unsupervised learning, which produces discrete representations of its training samples. This is the map the name refers to.

During the training process a vector is chosen at random, the vectors will have 3 values corresponding red/green/blue values of a pixel. An m*n array of neurons will also be placed over the image with a weight vector assigned to them The image is then scanned for the neuron which is closest to the vector, the best matching unit, calculated using the Euclidean distance formula. The
chosen neuron’s and its neighbour’s colour vectors are updated according to the formula

\[ C_v(t + 1) = C_v(t) + F(v, t) \alpha(t)(I(t) - C_v(t)) \]

Where \( C_v(t) \) is the neuron’s colour value

\( F(v, t) \) is the function for determining which pixels are considered neighbors for the purpose of the SOM, usually with a binary result. 1 for a neighbor and 0 otherwise.

\( I(t) \) is the input vector

\( \alpha(t) \) is a learning coefficient which decreases with each iteration [9].

This process is repeated with new sample inputs as necessary. When the process is finished the array of neurons will have dispersed themselves to outline shapes in the image as seen in the image below:

**Hierarchical self-organising maps:**

CITU uses a variant of the SOM called a hierarchical self ordering map or HSOM. Whereas a SOM uses a set size of neuron grid, which maybe too big or too small for the image in question.

HSOM starts at a 2 by 2 grid. As the HSOM continues the size of the grid increases to 4 by 4 then 8 by 8 and so on. HSOM’s are preferred to single-layer SOM’s because the single-layer procedure requires the number of desired segments in order to determine the number of units in the output. This use of multiple layers both removes the need for prior knowledge and results in a much more precise segmentation [19].

Segmentation results from each layer by using grey-scale feature vectors (reprinted from [19]).
Chapter 5: Working with CITU

What is CITU?
CITU stands for Computerised Image and Text Understanding. It is a program currently being developed in Trinity College under the supervision of Prof. Khurshid Ahmad from the school of computer science and Dr Anthony Davies from the institute of molecular medicine in St. James’s Hospital. CITU aims to provide a solution to the disadvantages of the software solutions mentioned in the “state of the art” section in chapter 2. CITU is designed to perform image segmentation and annotation on a large number of images. It uses neural learning methods, which remove the need for a user to define objects and features to extract unlike the existing solutions.

Its use of HSOM means that it can learn itself from the input information without the need of a priori knowledge. This has a by-product of removing the need for algorithms which would otherwise limit the types of cells it could identify. It can also learn from the image content to adjust the parameters for segmentation, thereby removing another of the current solution’s downfalls. Finally its current annotation algorithm returned an 85% rate of success in expert testing, which shows it could potentially remove the need for visual mining of data after segmentation.

Ensuring portability:
The original version of CITU I was provided with was heavily tied to the machine it was designed and built on (henceforth known as the parent machine). Because of this my first task with CITU was to port it to my own home machine. The parent machine is running Windows XP and the original database was built using Microsoft SQL 2005, where as my machine runs Windows 7 and Microsoft SQL 2008. By porting CITU to my machine any changes I would make to the program would bring it up to date with the latest versions of these programs. This would ensure that the program could be more easily moved from machine to machine for development and also to minimise the number of portability issues upon eventual release.

Issues to be resolved:
During the transporting of CITU from the parent machine several errors and bugs were discovered which needed to be resolved in order to have a fully functioning copy of CITU for Windows 7.

Database connections:
The copy of CITU on the parent machine had a very specific database connection string. The database required; a username, a password and a workstation ID unique to the computer. It also needed network and IP address information to connect to the online database. To compound the issue the online database could no longer be accessed.
It was decided to change the program to use a database stored locally on the new machine. This removed the need for a unique workstation ID, the network and the IP address. It also allowed the use of the trusted connection flag which removed the need for a password, which was not as necessary now the database was stored locally and inaccessible to other machines.

Another issue with the database was using a newer version of Microsoft SQL to create the database. Certain data types had been removed and replaced with expanded versions of other data types e.g. the data type “text” was removed with users having to change the data type to “varchar(X)”. This meant that the types listed in CITU were not the ones required in the database table set up. A closer look at the SQL documentation gave the types need so the tables could be set up correctly.

**Missing data:**
When the database was connection was functioning, I noticed that the images and image descriptions needed to populate the database were missing. When searching for them, it was discovered that the hard drive containing them had been wiped clean. I had to contact a previous student of Prof. Ahmad’s in England in order to get a copy of the files. Eventually the files were made available to me on the 13th of March but at that stage the project had been significantly delayed.

**Memory permissions:**
Once the database had been populated the next step was to test the image retrieval functionality. When the database search was performed it returned the correct files, however when an image was selected to load, CITU would return a file not found message. It was discovered that this is due to how CITU loads files from the database. Upon selecting a file to load CITU saves a temporary copy directly to the computers C drive. Unfortunately with Windows 7 programs must have administrator permission to save directly to the C drive as a security precaution and must save files into a folder on the drive instead. As the parent machine runs Windows XP this had not been an issue and the problem had not occurred there. Fixing the problem required a change to CITU’s file saving path. By discovering this error other errors were avoided as the source code could be checked for similar occurrences of bad save and load paths. Other examples of this error were found in storage of training results and in local saving of images.

**Training of neural network:**
While attempting to use the training mode another error was encountered. When CITU tried to load the images from the database for training and out of memory exception occurred. In an attempt to remedy the problem, the version of CITU on the parent machine was accessed; unfortunately the data for the original had been corrupted. Due to time constraints this issue was not
resolved. This prevented further testing and evaluation of CITU’s HSOM segmentation and image annotation.

Performing Image Segmentation:
The first piece of main functionality to be tested was the individual image segmentation. The images can be grey scaled based on any of the three colour values.

Fig 1. Cell image (i) grey scaled by red (ii), green (iii), and blue (iv) intensity.

The images may also have the level of noise reduced by a median filter and have the texture smoothed using Law’s filter.

Fig 2. Cell image before (i) and after (ii) median filter.
Segmentation itself can be performed with both region growing methods and with Otsu’s method in both 1D and 2D versions.

However due to the issue with loading training data from the database SOM and HSOM are unusable.
Chapter 6: Conclusions

Research Conclusions:
After discussing the different types of image segmentation, the conclusion is to decide which is the most suitable for the specific purpose of cell image segmentation.

Threshold methods work well on sparsely populated images, however they struggle to differentiate between different cells where there is overlap in the image. As this is not something that can be prevented they are not suitable for this particular type of segmentation.

Boundary methods struggle with image cell segmentation. The regions created by split and merge are too rigid to segment cells correctly. Grow and merge tends to over segment cells due to the distinct sections in them. The watershed algorithm reduces that but does not remove it completely. Similar to threshold based methods it can work in some instances but the ones it does not work for cannot be avoided.

The downfall of pixel based methods is there extreme noise sensitivity, as HCS images tend to have a large amount of noise. This makes pixel based segmentation almost useless for this particular branch of segmentation.

One of the most reliable methods of edge detection is the snake algorithm, due to its high levels of reliability regardless of the shape of the object. Its main downfall is sharp corners which are uncommon in cells, leaving it with very few negatives.

HSOM are also incredibly useful as the changing grid size removes some of the limitations of SOM. Also due to the learning aspect of the method, once training data is given, the cell can be segmented regardless of type or shape.

To conclude the most useful image segmentation method would be a hybrid method of the snake algorithm and HSOM. CITU itself has options for threshold (Otsu’s method), region growing, SOM and HSOM.

While this is the most fitting segmentation method for cell images, it may not be the case for other segmentation applications. The HSOM method will still function but other less processor intensive methods may be more suitable. For example, in text recognition, text will generally have very clear boundaries and a large difference in pixel intensity between background and foreground. This means boundary based methods or histogram methods would be more appropriate.
CITU conclusions:
In previous tests CITU’s annotation algorithm was found to have an 86.86% success rate. The tests were performed by giving a trained version of CITU a series of images to segment and annotate. The images were equally split between; star shaped cells, round shaped cells, elongated cells and stumpy cells. An expert was also asked to look at the same photos and classify them into the four categories. The results were compared and compiled into the confusion matrix shown here.

<table>
<thead>
<tr>
<th></th>
<th>Star</th>
<th>Round</th>
<th>Elongated</th>
<th>Stumpy</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star</td>
<td>22.71%</td>
<td>0.83%</td>
<td>1.67%</td>
<td>1.25%</td>
<td>26.46%</td>
</tr>
<tr>
<td>Round</td>
<td>0.42%</td>
<td>22.08%</td>
<td>0.00%</td>
<td>1.88%</td>
<td>24.38%</td>
</tr>
<tr>
<td>Elongated</td>
<td>1.25%</td>
<td>1.46%</td>
<td>22.92%</td>
<td>2.71%</td>
<td>28.33%</td>
</tr>
<tr>
<td>Stumpy</td>
<td>0.63%</td>
<td>0.63%</td>
<td>0.42%</td>
<td>19.17%</td>
<td>20.83%</td>
</tr>
<tr>
<td>Total</td>
<td>25.00%</td>
<td>25.00%</td>
<td>25.00%</td>
<td>25.00%</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

Due to the training functionality not being ported successfully because of time constraints, I was unable to begin the next round of testing in St. James’s Hospital.

The modifications made to CITU to bring it up to date with the latest versions were successful, apart from the previously mentioned training error. The changes made to Windows file storage paths in Windows Vista continued into Windows 7. With this in mind, the hope is that the file path modifications will still work with the next version of Windows, although one cannot be certain. The database tables’ structures are also unlikely to change in the next version of Microsoft SQL. If this is true it means there will be less work for any other individual updating the program to newer versions in the future. Hopefully any other changes that are needed will be simpler than the changes made to essentially skip an edition of the operating system.
Acknowledgements:
I would like to thank Prof. Khurshid Ahmad for all his help and support throughout the project and always being level headed despite the issues.

I would also like to thank Maria Francesca O’Connor for putting up with my questions about CITU and getting those elusive files to me.

Bibliography:
1. BioMarket 2010
13. Zhao Cheng ; Tianxu Zhang ; Luxin Yan ; “Thresholding using two-dimensional histogram based on local entropy,” Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference 6-7 March 2010


