A PROJECT REPORT ON

EXPLORING CELL SEGMENTATION AND TRACKING IN MICROSCOPIC IMAGES

A Project submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

APPIDI ROHITH REDDY	(17071A0463)
KATTA SHYAM PRASAD	(17071A0490)
UMMAGANI AVINASH	(17071A04B4)

UNDER THE GUIDANCE OF

Dr. L PADMA SREE

Professor

TAMASŌMA JYOTIRGAMAYA

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VIGNANA JYOTHI NAGAR, NIZAMPET, PRAGATHI NAGAR,

HYDERABAD, TELANGANA 50009

2017-2021

VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

VIGNANA JYOTHI NAGAR, NIZAMPET SO, PRAGATHI NAGAR, HYDERABAD, TELANGANA 500090

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

TAMASŌMA JYOTIRGAMAYA

CERTIFICATE

This is to certify that the Industry Oriented Mini Project report entitled **"EXPLORING CELL SEGMENTATION AND TRACKING IN MICROSCOPIC IMAGES"** being submitted by **A. ROHITH REDDY(17071A0463), K. SHYAM PRASAD (17071A0490), U. AVINASH (17071A04B4),** in partial fulfilment of the degree of Bachelor of Technology in Electronics and Communication Engineering during the academic year 2017 – 2021.

Certified further, to the best of our knowledge, the work reported here is not a part of any other project on the basis of which a degree or an award has been given on an earlier occasion to any other candidate. The result has been verified and found to be satisfactory.

Internal Guide **External Examiner**

Dr. L PADMA SREE

Professor

Head of the Department, ECE

DR.Y. PADMA SAI

Professor

CONTENTS

CHAPTER	Page No
1. INTRODUCTION	4
1.1. Overview of the Project	
1.2. Motivation	
1.3. Objectives	
1.4. Tools Used	
2. STUDY OF TECHNOLOGIES USED	7
2.1. Introduction	
2.2. Digital Image Processing	
2.3. Image segmentation	
2.4. Nearest Neighbour Algorithm	
3. STUDY OF TOOLS USED	15
3.1. Matlab	
3.2. Image Processing Toolbox	
3.3. Time-Lapse Imaging	
4. IMPLEMENTATION	18
4.1. Pre-processing	
4.2. Segmentation	
4.3. Post-processing	
4.4. Feature Extraction	
4.5. Cell Tracking	
5. SIMULATION RESULTS	24
5.1. Thresholding Technique	
5.2. Edge Detection	
5.3. Binary and Dilated Gradients	
5.4. Segmented Image (Centroid calculation)	
5.5. Cell Tracking Simulation	
6. CONCLUSION AND FUTURE SCOPE	27
7. REFERENCES	28

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

Identifying cells in an image (cell segmentation) is an essential process for quantitative single-cell biology via optical microscopy. Most current segmentation algorithms rely on a few basic approaches that use the gradient field of the image to detect cell boundaries. However, many microscopy protocols can generate images with characteristic intensity profiles at the cell membrane. This has not yet been algorithmically exploited to establish more general segmentation methods.

Image segmentation is an old topic but in this cell image segmentation area, it is widely used in the medical field, many researchers and scientists have done a lot of frameworks using a cell microscopic image. Cell image segmentation used to detect the number of cells and their position. Cell detection described basic strategies to detect different symptoms and diseases in the human body. The accurate segmentation and tracking of cells in microscopy image sequences is an important task in biomedical research, e.g., for studying the development of tissues, organs or entire organisms. However, the segmentation of touching cells in images with a low signal-to-noise-ratio is still a challenging problem.

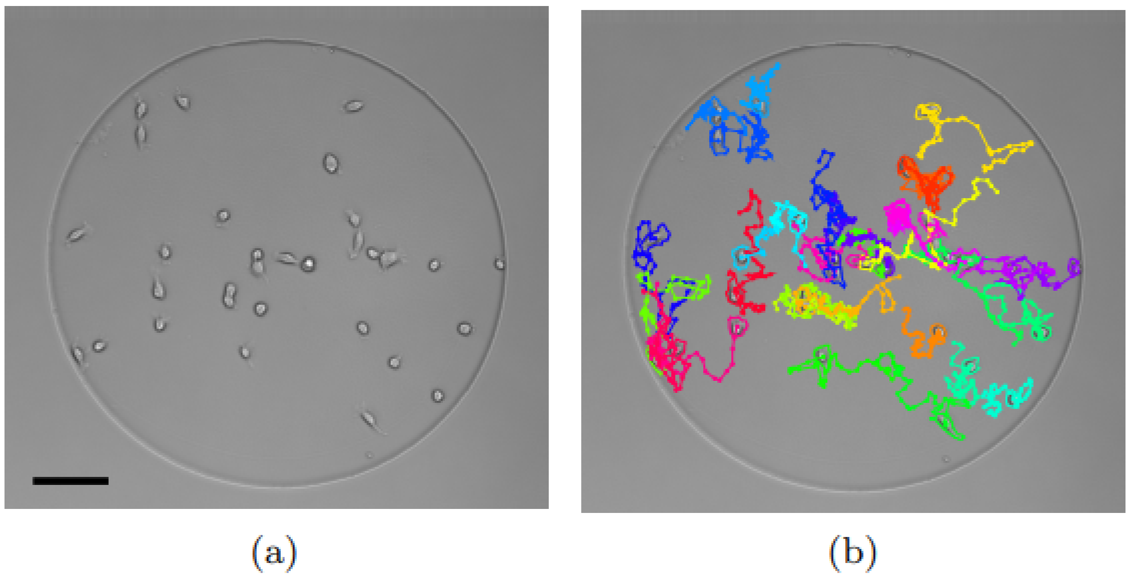


Figure 1.1: Example of input and output of an automated cell tracking system.

In this project, we present a method for the segmentation of cells in microscopy images. Here, we developed an automated cell counting system using segmentation and classification methods. By using a novel representation of cell borders, inspired by edge detection and thresholding techniques, our method is capable of utilizing not only touching cells but also close cells. Furthermore, this representation is notably robust to annotation errors and shows promising results for the segmentation of microscopy images contained in the training data underrepresented or not included cell types. For the prediction of the proposed neighbor distances, an adapted nearest neighbor algorithm was used. We also compared the efficiency of different segmentation and classification algorithms for cell images.

1.2 MOTIVATION

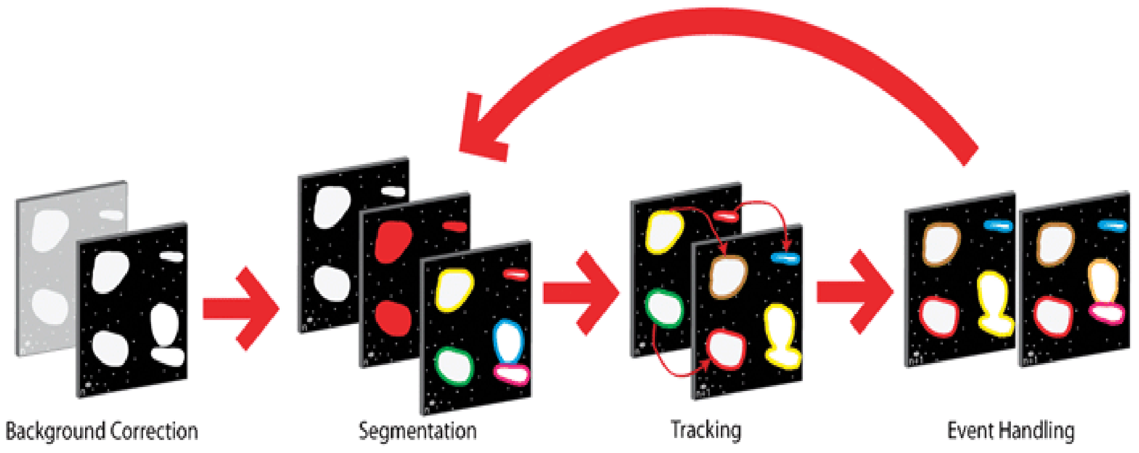
In biology, many different kinds of microscopy are used to study cells. There are many different kinds of transmission microscopy, where light is passed through the cells, that can be used without staining or other treatments that can harm the cells. There is also fluorescence microscopy, where fluorescent proteins or dyes are placed in the cells or in parts of the cells, so that they emit light of a specific wavelength when they are illuminated with light of a different wavelength. Many fluorescence microscopes can take images on many different depths in a sample and thereby build a three-dimensional image of the sample. Fluorescence microscopy can also be used to study particles, for example viruses, inside cells. Modern microscopes often have digital cameras or other equipment to take images or record time-lapse video. When biologists perform experiments on cells, they often record image sequences or sequences of three-dimensional volumes to see how the cells behave when they are subjected to different drugs, culture substrates, or other external factors. Previously, the analysis of recorded data has often been done manually, but that is very time-consuming and the results often become subjective and hard to reproduce. Therefore there is a great need for technology for automated analysis of image sequences with cells and particles inside cells. Such technology is needed especially in biological research and drug development.

1.3 OBJECTIVES

- To develop an automated cell segmentation and tracking system.
- Segment cells using thresholding and edge detection methods.
- Removing noise to improve the efficiency of the proposed system.
- Obtaining centroids of the segmented cells.
- To track the path of cells from time-lapse microscopic images.
- Analysing the obtained results from tracking such that different metabolic activities of cells are studied.

1.4 TOOLS USED

- Matlab
- Image Processing Toolbox
- Time-Lapse Imaging Tools



Source: xlink.rsc.org

Figure 1.2: Process involved in the cell segmentation and tracking system.

CHAPTER 2

STUDY OF TECHNOLOGIES USED

2.1 INTRODUCTION

We have used various technologies in the development process of the proposed model. They are used based on particular functionalities and aspects. The model mainly revolves around the concept of image processing. It includes capturing multiple images and analysing similar data in them. This data is used to analyse particular patterns and also used for feature extraction.

2.2 DIGITAL IMAGE PROCESSING

Signal processing is a discipline in electrical engineering and in mathematics that deals with analysis and processing of analog and digital signals , and deals with storing , filtering , and other operations on signals. These signals include transmission signals , sound or voice signals , image signals , and other signals e.t.c.

Out of all these signals , the field that deals with the type of signals for which the input is an image and the output is also an image is done in image processing. As its name suggests, it deals with the processing of images.

It can be further divided into analog image processing and digital image processing.

Analog image processing

Analog image processing is done on analog signals. It includes processing on two dimensional analog signals. In this type of processing, the images are manipulated by electrical means by varying the electrical signal. The common example included is the television image.

Digital image processing has dominated over analog image processing with the passage of time due its wider range of applications.

Digital image processing

Digital image processing deals with developing a digital system that performs operations on a digital image.

What is an Image

An image is nothing more than a two dimensional signal. It is defined by the mathematical function f(x,y) where x and y are the two coordinates horizontally and vertically.

The value of f(x,y) at any point gives the pixel value at that point of an image.

Figure 2.1: Sample Image

The above figure is an example of a digital image that you are now viewing on your computer screen. But actually , this image is nothing but a two dimensional array of numbers ranging between 0 and 255.

128	30	123
232	123	321
123	77	89

80	255	255

Each number represents the value of the function f(x,y) at any point. In this case the value 128 , 230 ,123 each represents an individual pixel value. The dimensions of the picture is actually the dimensions of this two dimensional array.

2.3 IMAGE SEGMENTATION

Let's understand image segmentation using a simple example. Consider the below image:

Figure 2.2: Image Segmentation

There's only one object here – a dog. We can build a straightforward cat-dog classifier model and predict that there's a dog in the given image. But what if we have both a cat and a dog in a single image?

Figure 2.3: Cat-Dog Classification

We can train a multi-label classifier, in that instance. Now, there’s another caveat – we won’t know the location of either animal/object in the image. That’s where image localization comes into the picture. It helps us to identify the location of a single object in the given image. In case we have multiple objects present, we then rely on the concept of object detection (OD). We can predict the location along with the class for each object using OD.

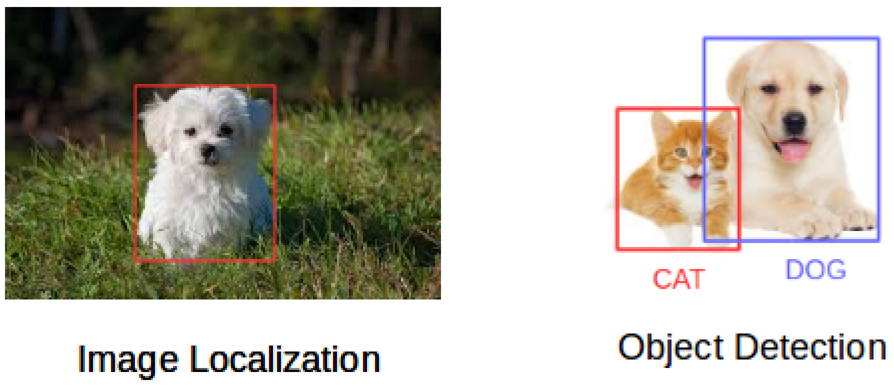


Figure 2.4: Image Localisation and Object Detection

Before detecting the objects and even before classifying the image, we need to understand what the image consists of. We can divide or partition the image into various parts called segments. It’s not a great idea to process the entire image at the same time as there will be regions in the image which do not contain any information. By dividing the image into segments, we can make use of the important segments for processing the image. That, in a nutshell, is how image segmentation works. An image is a collection or set of different pixels. We group together the pixels that have similar attributes using image segmentation. Take a moment to go through the below visual (it’ll give you a practical idea of image segmentation):

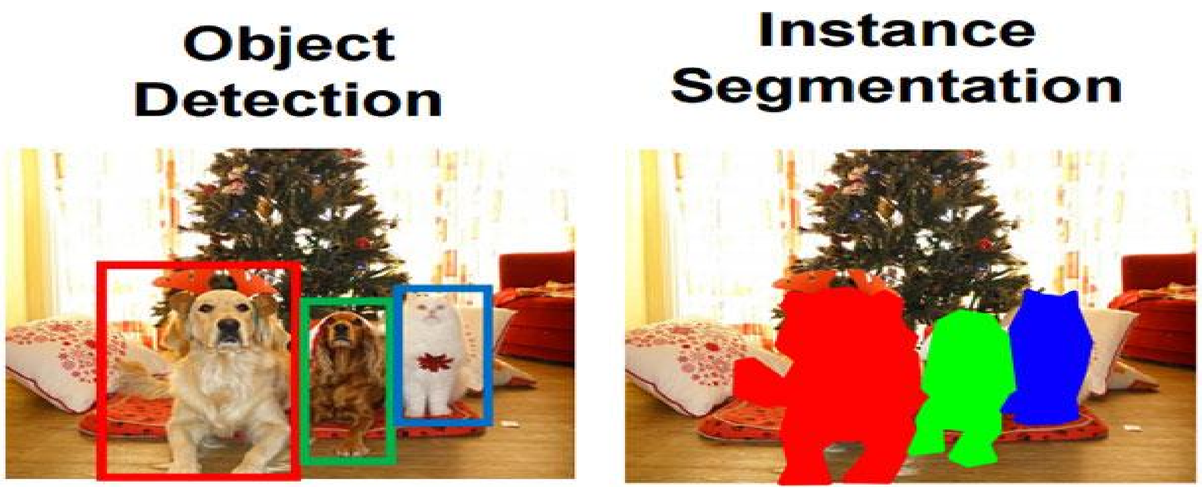


Figure 2.5: Object Detection and Instance Segmentation

Object detection builds a bounding box corresponding to each class in the image. But it tells us nothing about the shape of the object. We only get the set of bounding box coordinates. We want to get more information – this is too vague for our purposes. Image segmentation creates a pixel-wise mask for each object in the image. This technique gives us a far more granular understanding of the object(s) in the image.

Why do we need Image Segmentation?

Cancer has long been a deadly illness. Even in today's age of technological advancements, cancer can be fatal if we don't identify it at an early stage. Detecting cancerous cell(s) as quickly as possible can potentially save millions of lives. The shape of the cancerous cells plays a vital role in determining the severity of the cancer. You might have put the pieces together – object detection will not be very useful here. We will only generate bounding boxes which will not help us in identifying the shape of the cells. Image Segmentation techniques make a MASSIVE impact here. They help us approach this problem in a more granular manner and get more meaningful results. A win-win for everyone in the healthcare industry.

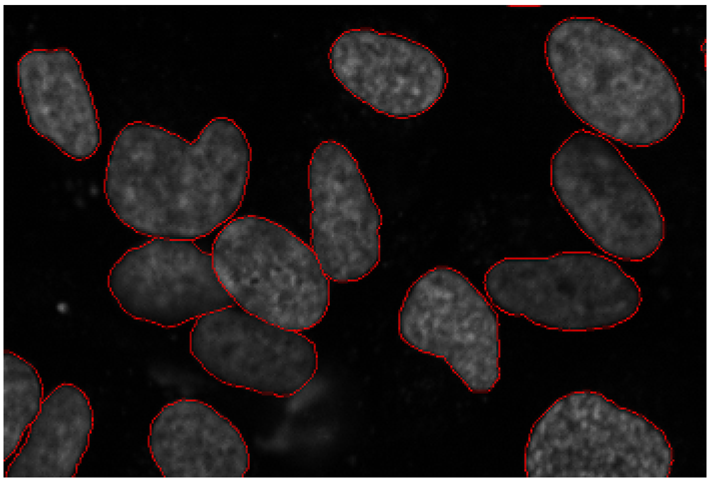


Figure 2.6: Cancer cell segmentation

Here, we can clearly see the shapes of all the cancerous cells. There are many other applications where Image segmentation is transforming industries:

- Traffic Control Systems
- Self Driving Cars
- Locating objects in satellite images

There are even more applications where Image Segmentation is very useful. Feel free to share them with me in the comments section below this article – let's see if we can build something together.

We can broadly divide image segmentation techniques into two types. Consider the below images:

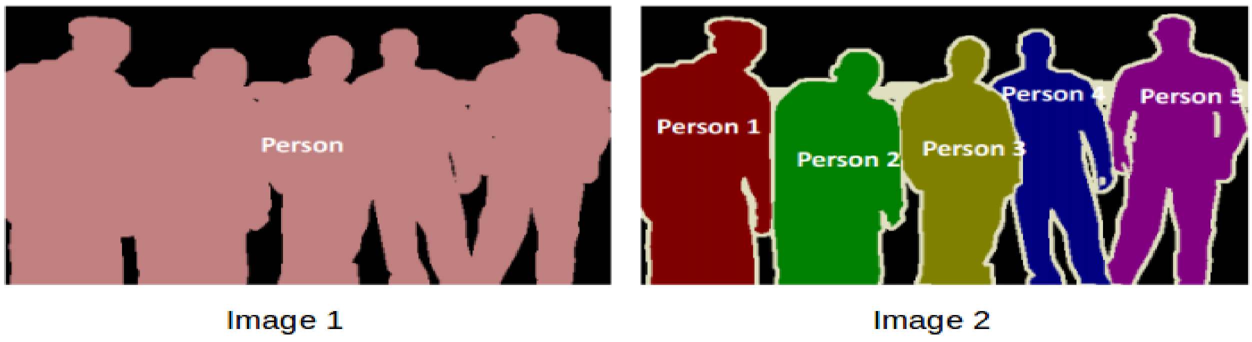


Figure 2.7: Image segmentation variants

Can you identify the difference between these two? Both the images are using image segmentation to identify and locate the people present.

- In image 1, every pixel belongs to a particular class (either background or person). Also, all the pixels belonging to a particular class are represented by the same color (background as black and person as pink). This is an example of semantic segmentation.
- Image 2 has also assigned a particular class to each pixel of the image. However, different objects of the same class have different colors (Person 1 as red, Person 2 as green, background as black, etc.). This is an example of instance segmentation.

If there are 5 people in an image, semantic segmentation will focus on classifying all the people as a single instance. Instance segmentation, on the other hand. will identify each of these people individually. So far, we have delved into the theoretical concepts of image processing and segmentation. Let's mix things up a bit – we'll combine learning concepts with implementing them in Python. I strongly believe that's the best way to learn and remember any topic.

2.4 NEAREST NEIGHBOURHOOD

Nearest neighbor search (NNS), as a form of proximity search, is the optimization problem of finding the point in a given set that is closest (or most similar) to a given point. Closeness is typically expressed in terms of a dissimilarity function: the less similar the objects, the larger the function values.

Formally, the nearest-neighbor (NN) search problem is defined as follows: given a set S of points in a space M and a query point $q \in M$, find the closest point in S to q. Donald Knuth in vol. 3 of *The Art of Computer Programming* (1973) called it the post-office problem, referring to an application of assigning to a residence the nearest post office. A direct generalization of this problem is a k-NN search, where we need to find the k closest points.

Most commonly M is a metric space and dissimilarity is expressed as a distance metric, which is symmetric and satisfies the triangle inequality. Even more common, M is taken to be the *d*-dimensional vector space where dissimilarity is measured using the Euclidean distance, Manhattan distance or other distance metric. However, the dissimilarity function can be arbitrary. One example is asymmetric Bregman divergence, for which the triangle inequality does not hold.

The nearest neighbour search problem arises in numerous fields of application, including:

- Pattern recognition – in particular for optical character recognition
- Statistical classification – see k-nearest neighbor algorithm
- Computer vision
- Computational geometry – see Closest pair of points problem
- Databases – e.g. content-based image retrieval
- Coding theory – see maximum likelihood decoding
- Data compression – see MPEG-2 standard
- Robotic sensing[2]
- Recommendation systems, e.g. see Collaborative filtering
- Internet marketing – see contextual advertising and behavioral targeting
- DNA sequencing
- Spell checking – suggesting correct spelling
- Plagiarism detection
- Similarity scores for predicting career paths of professional athletes.
- Cluster analysis – assignment of a set of observations into subsets (called clusters) so that observations in the same cluster are similar in some sense, usually based on Euclidean distance
- Chemical similarity
- Sampling-based motion planning

CHAPTER 3

STUDY OF TOOLS USED

3.1 MATLAB

MATLAB (an abbreviation of "matrix laboratory") is a proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

Although MATLAB is intended primarily for numeric computing, an optional toolbox uses the MuPAD symbolic engine allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

As of 2020, MATLAB has more than 4 million users worldwide. MATLAB users come from various backgrounds of engineering, science, and economics.

3.2 IMAGE PROCESSING TOOL KIT

Image Processing Toolbox provides a comprehensive set of reference-standard algorithms and workflow apps for image processing, analysis, visualization, and algorithm development. You can perform image segmentation, image enhancement, noise reduction, geometric transformations, image registration, and 3D image processing.

Image Processing Toolbox apps let you automate common image processing workflows. You can interactively segment image data, compare image registration techniques, and batch-process large data sets. Visualization functions and apps let you explore images, 3D volumes, and videos; adjust contrast; create histograms; and manipulate regions of interest (ROIs). Descriptions of a few functions that we have used in our project were described here under.

imhist(___) displays a plot of the histogram. If the input image is an indexed image, then the histogram shows the distribution of pixel values above a color bar of the color map map.

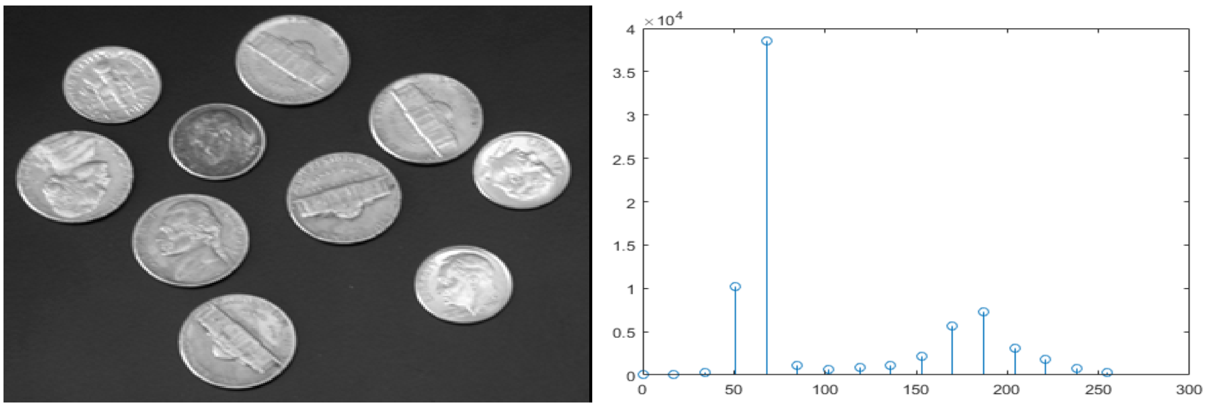
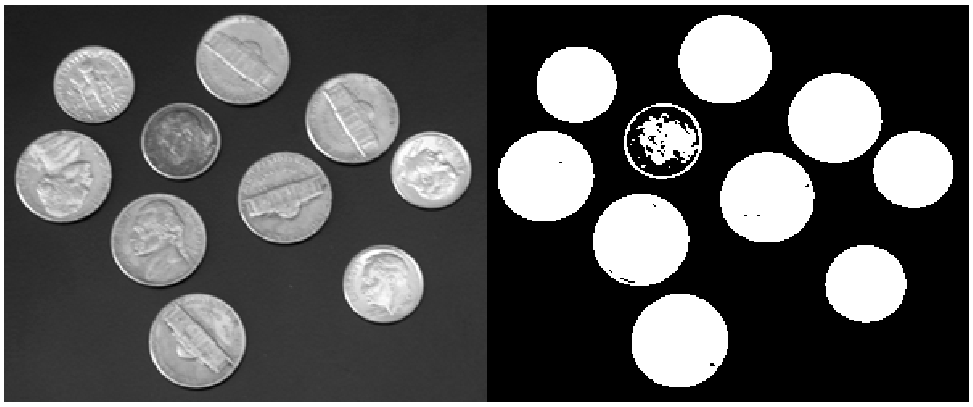


Figure 3.1

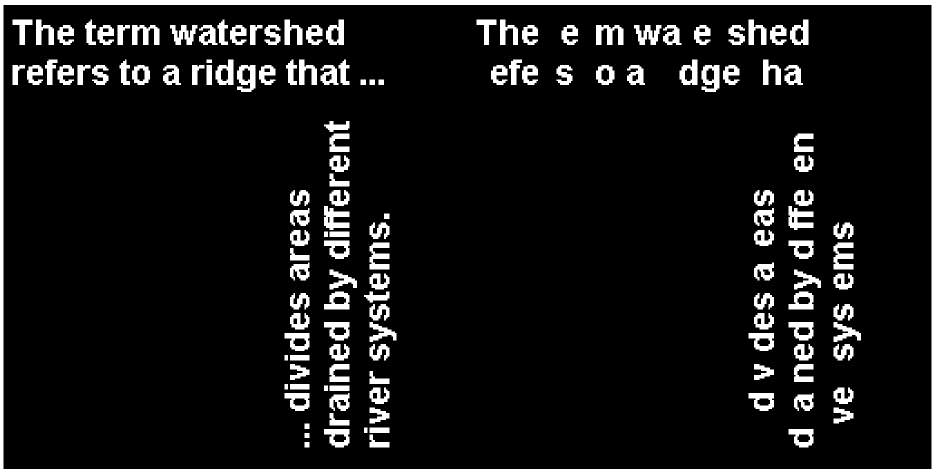
BW = im2bw(I,level) converts the grayscale image I to binary image BW, by replacing all pixels in the input image with luminance greater than level with the value 1 (white) and replacing all other pixels with the value 0 (black).

This range is relative to the signal levels possible for the image's class. Therefore, a level value of 0.5 corresponds to an intensity value halfway between the minimum and maximum value of the class.

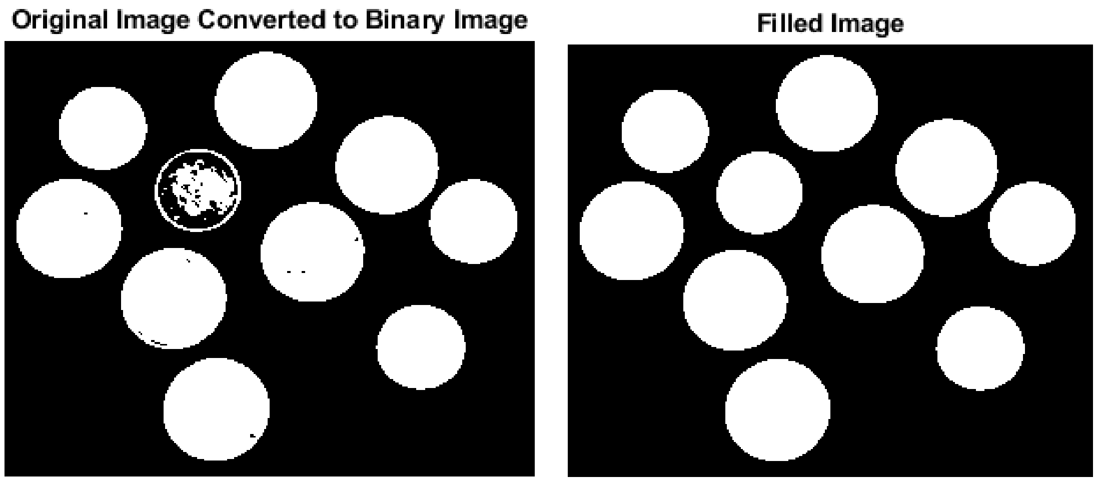


BW2 = bwareaopen(BW,P) removes all connected components (objects) that have fewer than P pixels from the binary image BW, producing another binary image, BW2. This operation is known as an *area opening*.

BW2 = bwareaopen(BW, 50);



BW2 = imfill(BW,'holes') fills holes in the input binary image BW. In this syntax, a hole is a set of background pixels that cannot be reached by filling in the background from the edge of the image.



CHAPTER 4

IMPLEMENTATION

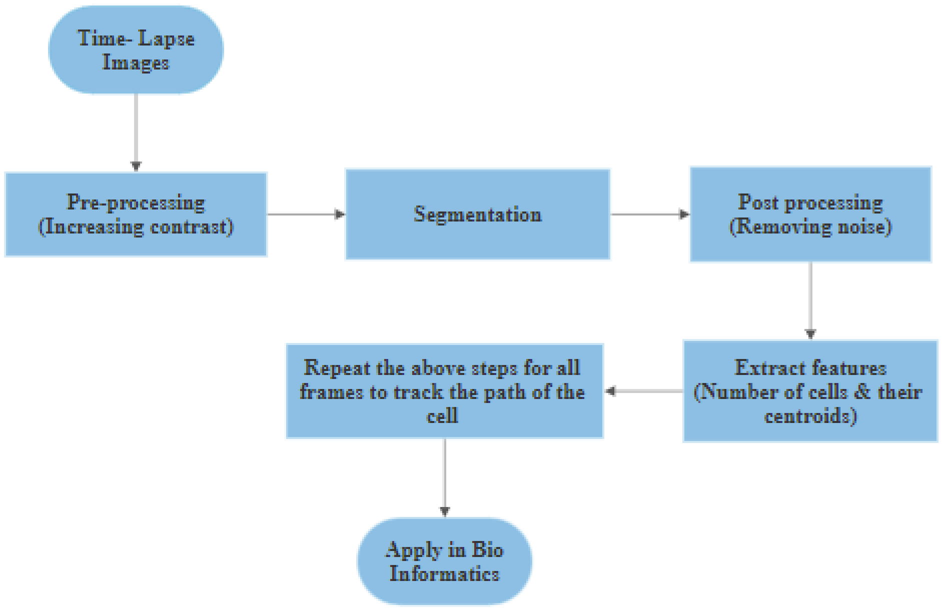

Time- Lapse Images
Pre-processing (Increasing contrast)
Segmentation
Post processing (Removing noise)
Extract features (Number of cells & their centroids)
Repeat the above steps for all frames to track the path of the cell
Apply in Bio Informatics

Figure 4.1 : Flow Chart

Matlab Code:

1)<u>Thresholding Technique</u>

```
clc
clear
I = imread('images2\t006.tif');
imshow(I);

%% Pre-processing
I_eq = adapthisteq(I);
imshow(I_eq);
imhist(I_eq)

%% Segmentation
bw = im2bw(I_eq, .095);
imshow(bw)
bw2=imfill(bw,'holes');
imshow(bw2)
```

```
%% Post-processing
se90 = strel('line',3,90);
se0 = strel('line',3,0);
BWsdil = imdilate(bw2,[se90 se0]);
imshow(BWsdil)
BWsdil=imfill(BWsdil,'holes');
imshow(BWsdil)
bw3 = bwareaopen(BWsdil, 2050);
imshow(bw3);
bw4 = imclearborder(bw3);
imshow(bw4)
bw4=imfill(bw4,'holes');
imshow(bw4)

%% Feature Extraction
s = regionprops(bw4,'centroid');
%imshow(s);
centroids = cat(1,s.Centroid);
%imshow(centroids)
imshow(bw4)
hold on
plot(centroids(:,1), centroids(:,2), 'b*')
hold off
plot(centroids(:,1), centroids(:,2), '*')
set(gca, 'YDir','reverse')
axis([0 1024 0 1024])
```

2)<u>Edge Detection</u>

```
clc
clear
%% Cell Segmentation
n_frames = 5; % Number of frames
points = cell(n_frames, 1);
for i = 1: n_frames
I = imread(sprintf('images/t%d.tif',i-1));
figure(n_frames+1);
imshow(I);
title('Original Image');
text(size(I,1),size(I,2)+25,'Cell Segmentation',
'FontSize',7,'HorizontalAlignment','right');
```

```
  [~,threshold] = edge(I,'sobel');
  BWs = edge(I,'sobel',threshold * 0.5);
  figure(n_frames+2);
  imshow(BWs)
  title('Binary Gradient')

  se90 = strel('line',3,90);
  se0 = strel('line',3,0);
  BWsdil = imdilate(BWs,[se90 se0]);
  figure(n_frames+3);
  imshow(BWsdil)
  title('Dilated Gradient Mask')

  BWdfill = imfill(BWsdil,'holes');
  figure(n_frames+4);
  imshow(BWdfill)
  title('Binary Image with Filled Holes')

  BWnobord = imclearborder(BWdfill,8);
  figure(n_frames+5);
  imshow(BWnobord)
  title('Cleared Border Image')

  % Create the diamond structuring element using the strel function.
  seD = strel('diamond',1);
  BWfinal = imerode(BWnobord,seD);
  BWfinal = imerode(BWfinal,seD);
  figure(n_frames+6);
  imshow(BWfinal)
  title('Segmented Image');

  %Finding Centroids
  BWfinal = bwareaopen(BWfinal, 2050);
  s = regionprops(BWfinal, 'all');
  centroids = cat(1, s.Centroid);
  [~, largestidx] = max([s.Area]); %find index of largest region
  s(largestidx).BoundingBox ; %coordinate of bounding box of largest region
  figure(n_frames+7);
  imshow(BWfinal);
  hold on
  plot(centroids(:,1), centroids(:,2), 'b*')
  for n=1: length(s)
      rectangle('Position', s(n).BoundingBox, 'EdgeColor', 'g',
'LineWidth',1),        title('Detected Cells')
  end
  hold off
  points{i} = centroids;
  end
```

```
% centroids in points
%% Tracking.m
debug = false;

[ tracks, adjacency_tracks ] = Tracking(points,'Debug', debug);

%plot points
clf

hold on
for i_frame = 1 : n_frames
  str = num2str(i_frame);
  for j_point = 1 : size(points{i_frame}, 1)
        pos = points{i_frame}(j_point, :);
        plot(pos(1), pos(2), 'o')
        text('Position', pos, 'String', str)
  end
end

%plot track
n_tracks = numel(tracks);
colors = hsv(n_tracks);

all_points = vertcat(points{:});

for i_track = 1 : n_tracks
% We use the adjacency tracks to retrieve the points coordinates. It
% saves us a loop.
track = adjacency_tracks{i_track};
track_points = all_points(track, :);
plot(track_points(:,1), track_points(:, 2), 'Color', colors(i_track, :))
end
```

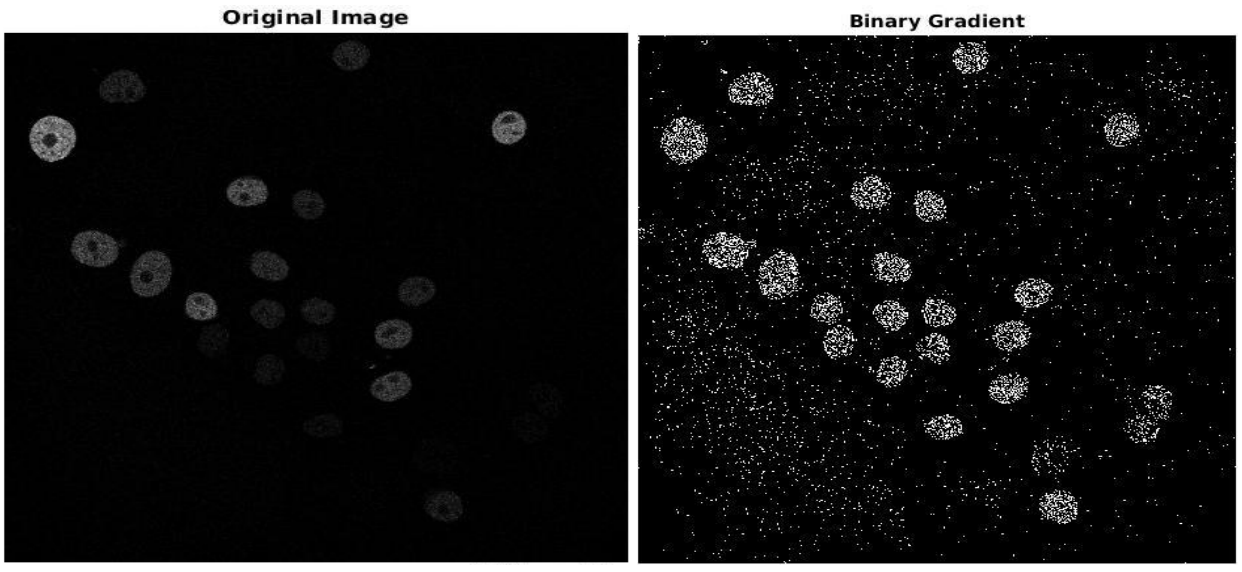
NearestNeighbour.m

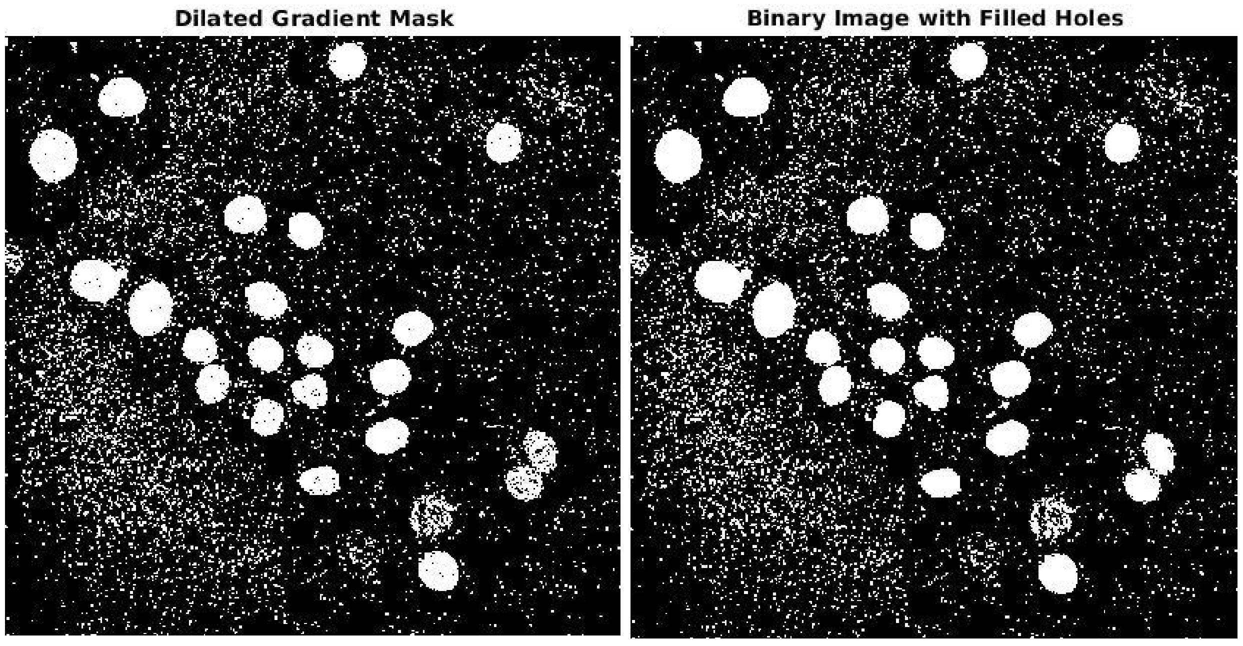
```
function [ target_indices, target_distances, unassigned_targets ] =
nearestneighbor(source, target, max_distance)
   if nargin < 3
       max_distance = Inf;
   end
   n_source_points = size(source, 1);
```

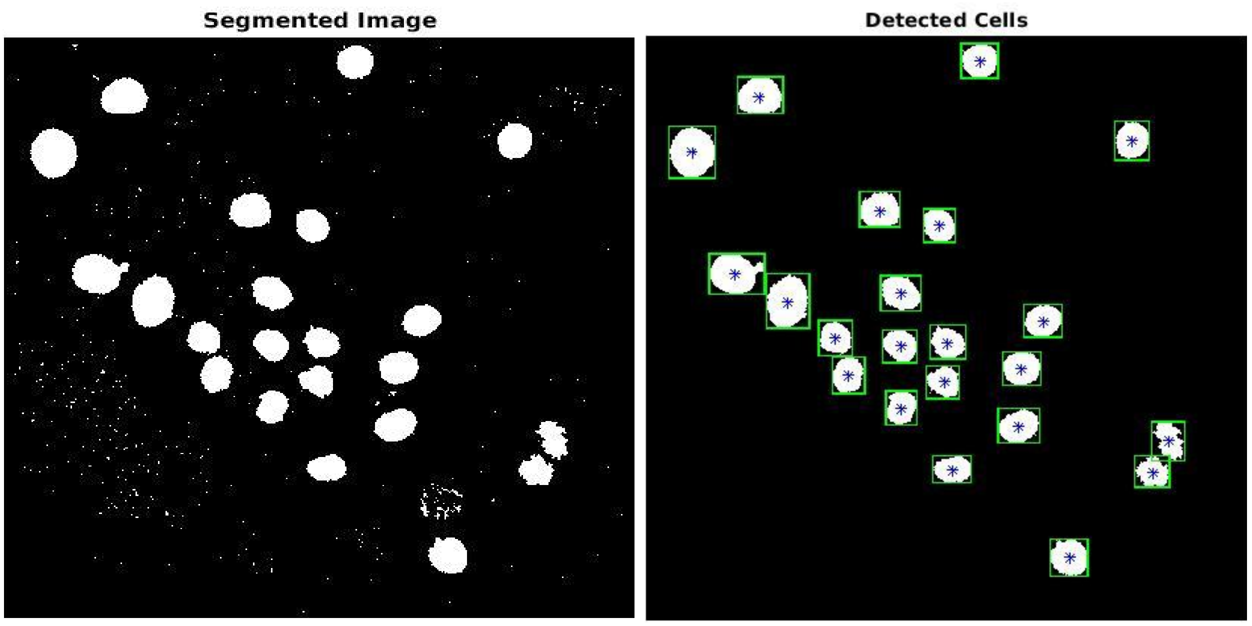
```
    n_target_points = size(target, 1);
    D = NaN(n_source_points, n_target_points);
    % Build distance matrix
    for i = 1 : n_source_points
        % Pick one source point
        current_point = source(i, :);
        % Compute square distance to all target points
        diff_coords = target - repmat(current_point, n_target_points, 1);
        square_dist = sum(diff_coords.^2, 2);
        % Store them
        D(i, :) = square_dist;
    end
    % Deal with maximal linking distance: we simply mark these links as
  already
    % treated, so that they can never generate a link.
    D ( D > max_distance * max_distance ) = Inf;
    target_indices = -1 * ones(n_source_points, 1);
    target_distances = NaN(n_source_points, 1);
    % Parse distance matrix
    while ~all(isinf(D(:)))
        [ min_D, closest_targets ] = min(D, [], 2); % index of the closest
  target for each source points
        [ ~, sorted_index ] = sort(min_D);
        for i = 1 : numel(sorted_index)
            source_index =  sorted_index(i);
            target_index =  closest_targets ( sorted_index(i) );
            % Did we already assign this target to a source?
            if any ( target_index == target_indices )
                % Yes, then exit the loop and change the distance matrix to
                % prevent this assignment
                break
else
% No, then store this assignment
target_indices( source_index ) = target_index;
target_distances ( source_index ) = sqrt ( min_D ( sorted_index(i) ) );
% And make it impossible to find it again by putting the target
% point to infinity in the distance matrix
D(:, target_index) = Inf;
% And the same for the source line
D(source_index, :) = Inf;
if all(isinf(D(:)))
break
end
end
end
end
unassigned_targets = setdiff ( 1 : n_target_points , target_indices );
end
```

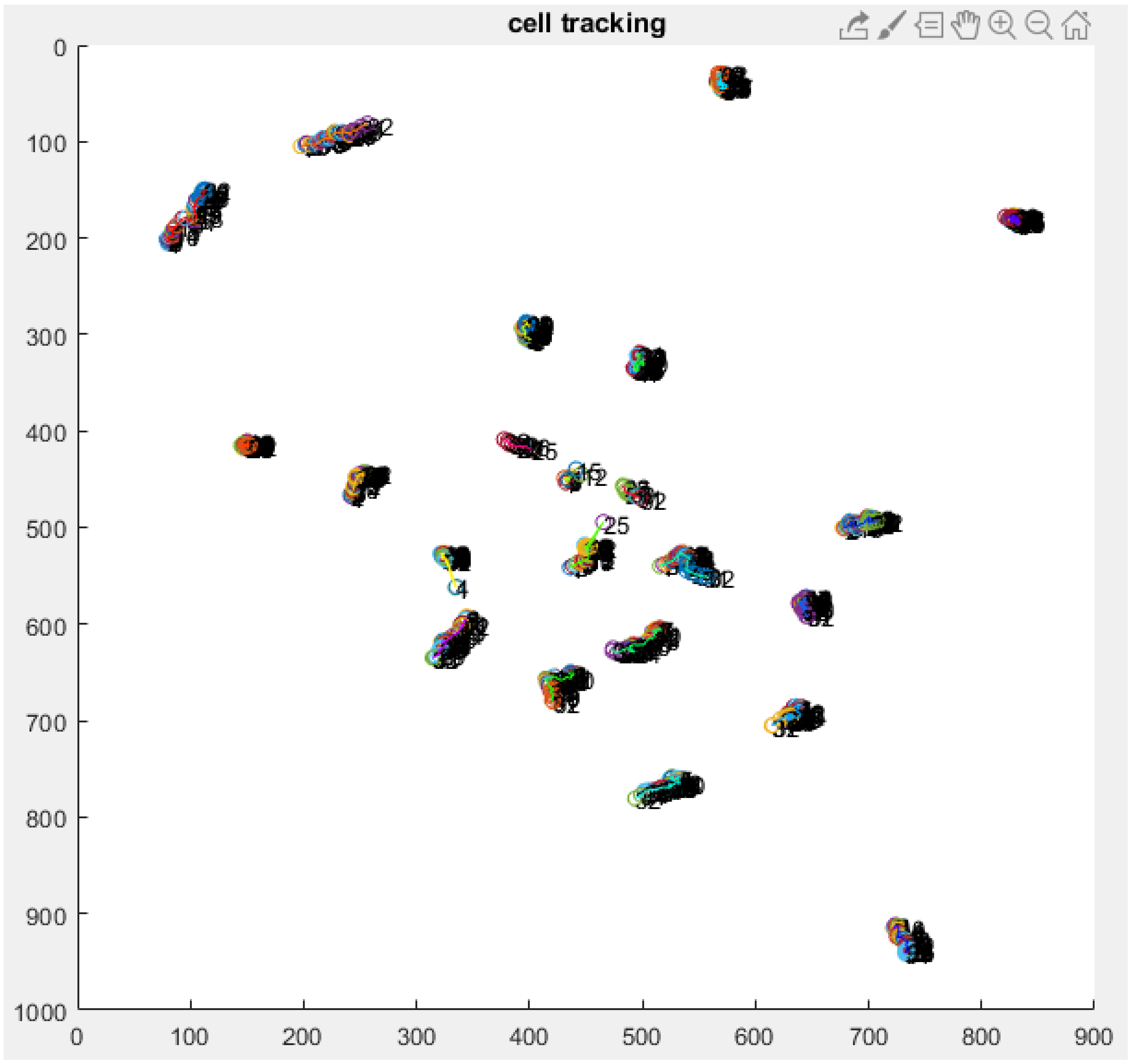
CHAPTER 5

SIMULATION RESULTS

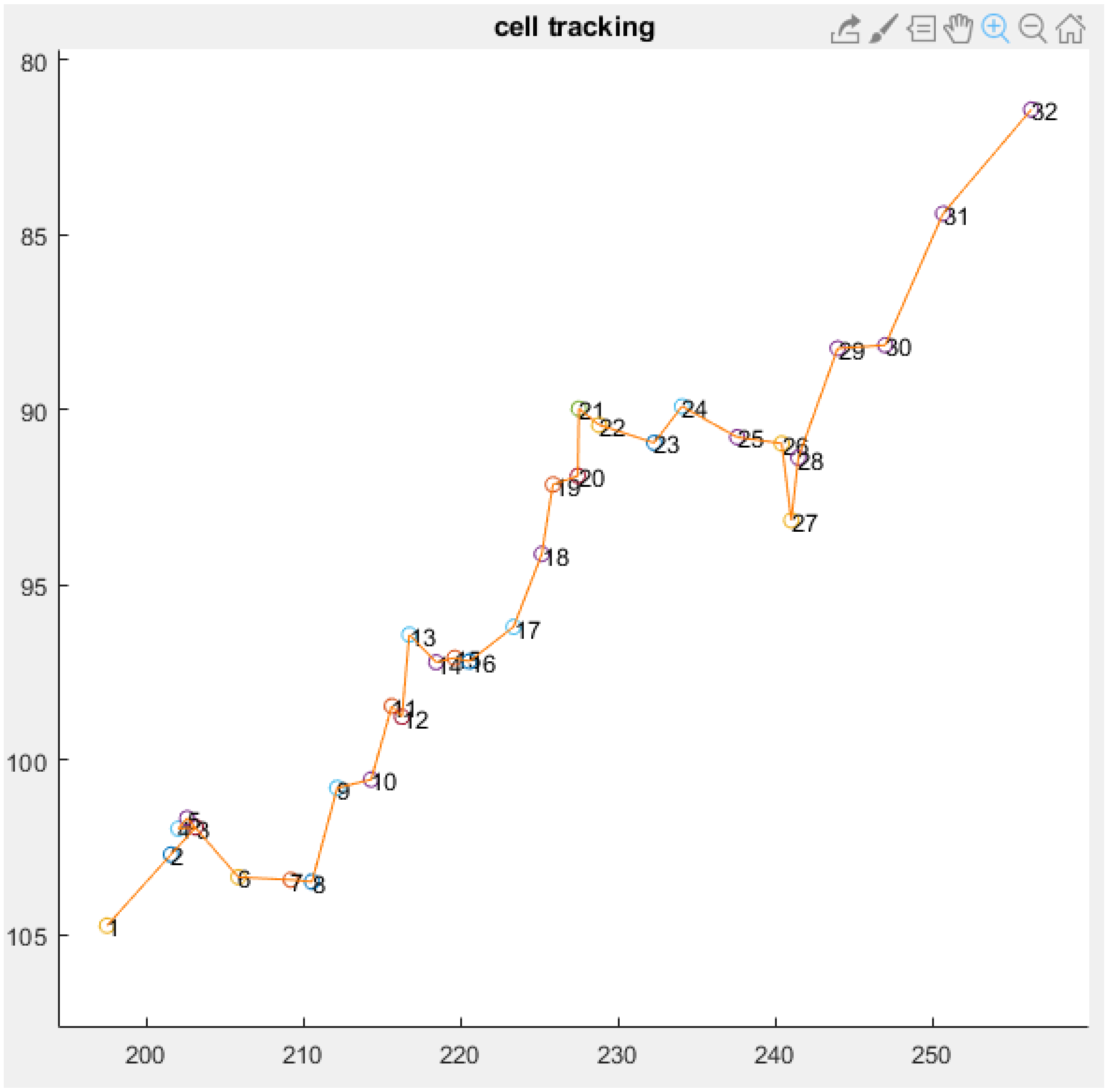





Segmented Image
Detected Cells


cell tracking
0
100
200
300
400
500
600
700
800
900
1000

1


cell tracking
80
85
90
95
100
105
200
210
220
230
240
250
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

Through this project, we have presented a system for cell segmentation and tracking, including algorithms for preprocessing, segmentation, track-linking, and analysis of created tracks. We have presented a global track linking algorithm with low computational complexity and high performance. The algorithm operates directly on the regions produced by the segmentation algorithm and can handle false positives, false negatives, division, death, appearance, disappearance, jointly segmented cells, and migration in and out of the field of view. The algorithm has been shown to give good performance when our complete cell tracking system was compared to other systems. The tracking algorithm itself cannot handle dynamic motion models, but we have shown that dynamic motion models can be incorporated into the tracking by preprocessing the detected object positions (centroids). This preprocessing can be used to improve the tracking performance for subcellular particles, and nuclei in developing biological analytics. Our system has proven to be useful in practical applications, through many successful collaborations with biologists in different fields. These collaborations have resulted in a number of biology publications where our system was used for data analysis. We have also seen that the system can be understood and used without our involvement, with the help of a user guide. We hope that our system will help many more biology labs answer their biological questions.

CHAPTER 7

REFERENCES

- Image Processing Toolbox Documentation (https://in.mathworks.com/help/images/index.html)
- P. Ventura de Oliveira and K. Yamanaka, "Image Segmentation Using Multilevel Thresholding and Genetic Algorithm: An Approach," 2018 2nd International Conference on Data Science and Business Analytics (ICDSBA), Changsha, China, 2018, pp. 380-385, doi: 10.1109/ICDSBA.2018.00078.
- M. Mohamed and B. Far, "An enhanced threshold based technique for white blood cells nuclei automatic segmentation," 2012 IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom), Beijing, China, 2012, pp. 202-207, doi: 10.1109/HealthCom.2012.6379408.
- **Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy** Scherr T, Löffler K, Böhland M, Mikut R (2020) Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy. PLOS ONE 15(12): e0243219. https://doi.org/10.1371/journal.pone.0243219.
- Analytics Vidya (https://www.analyticsvidhya.com/image-processing).
- Tutorialspoint (https://www.tutorialspoint.com/dip/image_processing_introduction.htm)
- Wikipedia (https://en.wikipedia.org/wiki/Nearest_neighbor_search#:~:text=Nearest%20neighbor%20search%20(NNS)%2C,the%20larger%20the%20function%20values)