Label-free cell tracking
Markeringsfri cellspårning

Jonathan Dahlberg (BME–19), Robin Öhrnberg (BME–19)

***Abstract*—Migrating cells under various force fields in microfluidic devices has been used as a tool to study the cell properties. Cells respond to the same force field in different ways depending on their properties, resulting in different migration velocities and trajectories which are obtained by cell tracking. The cells that are to be tracked are typically labelled with fluorescent dyes to enhance the contrast between cells and the medium. However,the staining process may change the cell properties to be measured. This paper describes the development of a cell tracking algorithm in MATLAB, which is able to track cells in a label-free manner with the images obtained using phase-contrast microscopy Dr. Wei Qiu. The algorithm operates on a set of images where the cells enter from the left and migrate along the microchannel until they exit the field of view on the right. The code is divided into two parts. An image processing part wherein the images are processed through thresholding, dilating and eroding, which is able to locate the cells. Then a data analysis part. By applying this part the cells on each image are connected and coherent paths for the cells are created. The algorithm has proven to be functional and manages to detect a vast majority of the cells while filtering out noise and irrelevant particles.**

I. Introduction

A. Cell tracking

In the study of cell behaviour, fluorescence microscopy is a very useful method for obtaining visual information about the cells. In fluorescence microscopy, cells are stained with a fluorophore which can be excited by electromagnetic radiation of a specific wavelength [1]. Once excited, the fluorophores emit electromagnetic radiation with a typically longer wavelength that can be detected by the camera [2] However, fluorescence microscopy has some notable drawbacks. In some cases the dyes and the staining process may alter the properties of the object that is being studied [3] [4]. Changes in properties might become an issue during studies of cell functions, either using living cell imaging or taking measurements of the cell properties based on the cell response in certain force fields. In the latter case migrating cells are exposed to a force field, e.g., an acoustic, an electric, or a magnetic field, etc. This affects the velocity and the trajectory of the cells depending on their acoustic, electric and magnetic properties. By tracking the velocities and trajectories of the cells indications of the respective properties in each cell can thus be obtained. The refractive indices of the cells and the surrounding cell medium are similar, leading to the use of fluorescence microscopy for enhanced cell contrast in studies of cell functions like those mentioned above. Changes to cell properties caused by the staining process in fluorescence microscopy remain a concern. Continuous progress of phase-contrast microscopy [5] and differential interference contrast microscopy [6] leads to significant enhancements of contrast in cell images without the need for staining. Making label-free cell tracking possible. In this project, a cell tracking algorithm is developed based on MATLAB, which enables automated label-free cell tracking with the images obtained using phase-contrast microscopy.

Handed in June 4, 2022

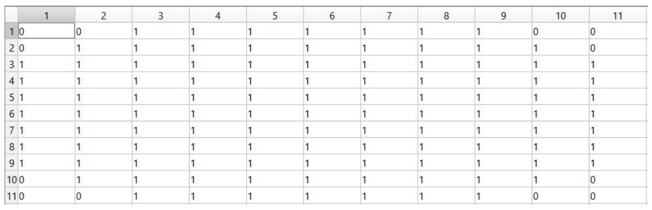
Email:{jonathan.dahlberg.4237@student.lu.se, robin.ohrnberg.433@student.lu.se}

Technical supervisor: Dr. Wei Qiu, Department of Biomedical Engineering

B. Image analysis

To date, there are many existing image analysis algorithms that enable label-free tracking. Image analysis is a very wide field with many applications. Since digital images are simply sets of information, this information can be manipulated in order to change the characteristics of the images. This can be used to for example heighten the contrast, change the colour scales, adjust the brightness of different areas, smooth out sharp edges and reducing noise. Thresholding is a useful method for segmenting the image and separating useful information from the background by introducing a certain limit value and treating pixels differently depending on their relation to this value. You could for example set all pixels that have value of less than the threshold to zero to highlight the rest of the pixels in the image. Besides intensity based methods such as thresholding some methods are morphology based. This means that the image is processed by applying some operation within a structural element traversing the image. This could for example be giving the pixel or the pixels at the center of the element, called the origin of the structural element, the highest or lowest value found in the pixels within structural elements neighbourhood. The neighbourhood is the area defined by the geometrical shape of the element. The neighbourhood is populated by binary values, if the maximum or minimum falls on a 0, no action is taken. The size and shape of the structural element can be chosen in accordance with what works best in a given situation [7]. Figure 1 shows an example of a structural element.

Image analysis is a field of study that has seen rapid progress in later years. Despite this it is difficult to develop an image analysis algorithm that is applicable for every given situation. Visual data is situation specific, it depends not only on what information you are searching for but also on what kind of image you are looking at [8]. This means that, at least for the moment, a tracking algorithm must be somewhat specialized to best fit the experimental data in question in order to yield the most reliable end results. Developing such an algorithm was the purpose of this project.



	1	2	3	4	5	6	7	8	9	10	11
1	0	0	1	1	1	1	1	1	1	0	0
2	0	1	1	1	1	1	1	1	1	1	0
3	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1
10	0	1	1	1	1	1	1	1	1	1	0
11	0	0	1	1	1	1	1	1	1	0	0

Figure 1. A structure element in the shape of a disc with the radius 5 pixels. The pixels within the radius is called the neighbourhood, the pixel in the middle i called the origin. If a maximum or minimum pixel value falls on a binary 0 no action is taken and the origin keeps its current pixel value.

C. *Aim and limitations*

The goal was to write an algorithm in MATLAB that could accurately track cells in the data sets that resulted from Dr. Wei Qiu's experiments in differential interference contrast imaging and phase-contrast imaging at Lunds Tekniska Högskola. The accuracy with which this algorithm can identify and track cells is dependant on the time spent refining the algorithm, but there is a point of diminishing returns. Diminishing returns on the time spent optimizing the code comes when large amounts of time is spent making the algorithm work in situations which rarely occur. Because of this the point at which the project can be considered definitively finished thus becomes somewhat arbitrary and we decided to combat this by working on optimizing the algorithm until a set deadline was reached and then showcasing our results. Shortly summarized, the objective was to: Develop a cell tracking algorithm within the given time frame. This paper displays the algorithm in full in the form of MATLAB code, as well as the resulting identification and tracking of the given cells. A review of potential further steps of optimization is also given.

D. *Outset*

MATLAB was used in this project since it contains many useful image analysis tools that would aid the identification of the cells. The original pictures from Dr. Wei Qiu's experiment were monochrome with the cells seen as white circles with black interiors against a grey background. The cells migrate from left to right in a microchannel with a width of 760 m and a height of 370 m, and the pictures were taken with 0.007 seconds delay between each. The cells depicted in the images are K562 cells, which are human myelogenous leukemia cell lines. In the experiments, the majority of the cells sedimented to the bottom of the microchannel and only the cells in focus are of interest for the tracking.

II. Method

The majority of the project consisted of self-educating in image processing in MATLAB. Theoretical knowledge was gathered through various means of self study while practical experience was gathered through MATLAB Image Processing On-ramp. The On-ramp in particular was helpful in acquiring a library of methods used for image analysis and processing. Through trial and error the final code was produced by testing the methods needed to extract specific information form the images in different orders of application. The algorithm is divided into two main parts. The first section focuses on processing the images and identifying the cells. The second focuses on sorting and combining the data that is produced by the first part to track the cells from image to image.

Dr. Wei Qiu's experiment resulted in four sets of TIF images. Three of the sets contained 300 pictures and one contained 301.

A. *Image processing*

```
ds=imageDatastore("/directory");
ds.Files=natsort(ds.Files);
I=readimage(ds,1);
imhist(I)
imtool
pause
```

The files are loaded using imageDatastore. imageDatastore does not store the files according to alphabetical or numerical order but instead by the value of the characters in the image file name. Writing an algorithm to sort the files proved to be time consuming and the pre-constructed algorithm natsort was used in order to remedy this issue. Since the images are very similar the first one is used as a template for gathering parameters such as pixel values and spatial measurements. The method imhist plots an intensity distribution of the pixels in an image in the form of a histogram. This histogram is used in conjunction with the function imtool, which is a function that contains multiple useful tools when handling images. Here imtool is firstly used to manually extract the pixel values inside the cells, which together with the histogram gives a good indication for possible thresholding values. Secondly imtool is used to extract the approximate radii of the cells. Pause causes MATLAB to pause the compilation of the code until the user presses enter, this is done so that the user can extract the pixel values and the radii.

```
SE=strel("disk",m);
cm={};
```

A structural element in the form of a disc with radius m is created with the method strel. In this case the radius of the disc should be smaller than the cells in the image for optimal results. The optimum value of m was found to be 5 pixels. A cell array is created for future use.

```
for i=1:numel(ds.Files);
I=readimage(ds,i);
I=I(:,:,i);
I=im2uint8(I);
I=imadjust(I);
Idialate=imdilate(I,SE);
I(I<n)=255;
Idialate(I>250)=255;
Iclose=imerode(Idialate,SE);
[centre, ~]=imfindcircles(Iclose,...
[a b]);
cm{i}=round(centre);
i
end
```

Each image is now run through a number of image processing methods. Firstly the two matrices containing information about the color scales in the image are removed. The remaining matrix is then converted to an eight-bit image since compressing layers of the grey scale makes thresholding easier. The image contrast is enhanced through imadjust. A new image is created by dilating the grey scale image using the previously introduced structure element. Dilating is a morphological operation where in the origin, the pixel in the middle of the structural element, is given the maximum value from the neighbourhood of the structural element. The original image is thresholded at the n-value, which is the pixel value from the cells interior extracted earlier with imtool. This serves as a template when thresholding the new image. The dilated image is then eroded with the structure element to reduce noise in the image. Erosion is a morphological operation where in the origin, is given the minimum value from the structural elements neighbourhood. The image is now ready for cell identification, see Figure 6. imfindcircles scans an image for circular shapes with radii within a given span and returns their center points in a matrix of coordinates. This method is applied to the image with variable a set as the minimum value of the radius and variable b as the maximum. Each cell center of mass is stored in the cell array cm.

B. Data analysis

```
cm1={};
CM=insertSort(cm,cm1);
```

A new cell array is created. The cell array cm now contains the coordinates of all the cells for each image. However, the coordinates are not sorted in any particular order. The self-written algorithm insertSort was implemented to address this.

```
function A=insertSort(cm,cm1)
cm1={length(cm)};
a=1;
C=insertSort2(cm,cm1,a);
A=C;
return
end
function B=insertSort2(cm,cm1,a)
if a>length(cm)
    B=cm1;
    return
end
D=cm{a};
D=sortrows(D,1);
cm1{a}=D;
a=a+1;
B=insertSort2(cm,cm1,a);
end
```

insertSort is a version of a recursive insertion sorting algorithm. Using the empty cell array cm1, insertSort works its way through every image and extracts the coordinates before inserting them in cm1 in order of magnitude based on the x-coordinate. In this way, the coordinates in each image are sorted from left to right in the picture. cm1 is then returned and named CM.

```
same=cell(1,1000);
I=CM{1};
for i = 1:size(I,1)
same{i}=[I(i,:) 1];
end
```

An empty cell array named same is created. The coordinates of the first image are copied and saved in variable I. A for-loop is initiated and the newly created cell array is filled with the coordinates of the cells in I, additionally, each set of coordinates is labelled with 1, representing which image they belong to. Thus the coordinates of each cell in the first image is saved as separate cells in the cell array same.

```
same=connectCells(CM,same);
```

The self-written function connectCells was implemented.

```
function S=connectCells(CM,same)
a=1;
S2=connectCells2(CM,same,a);
S=S2;
return
end

function S2=connectCells2(CM,same,a)
A=[];
B=[];
if a==length(CM)
    same=same(~cellfun(@isempty,...
    same));
    S2=same;
    return
end
x=find(~cellfun(@isempty,same));
x=x(1,end);
for i=1:x
A=same{i};
A=A(end,1:2);
B=[B ;A];
end
    D=CM{a+1};
    for i=1:length(B)
        C=B(i,:);
        E=pdist2(C,D,'euclidean');
        [M,I]=min(E);
        y=D(I,1);
        z=same{i};
        z=z(end,1);
        if isempty(y) ||y<z ||M>20
            continue
        end
        same{i}=[same{i};D(I,:)...
        a+1];
        D(I,:)=[0 0];
        D = D(~all(D == 0, 2),:);
    end

    if ~isempty(D)
    same=wDMHAPCF(same,D,a);
    end
    a=a+1;
    S2=connectCells2(CM,same,a);
    return
    end
```

connectCells is a recursive algorithm that pairs the last known coordinates for a cell with the closest set of coordinates in the following images. Since connectCells pairs up the last known coordinates of each cell with the closest coordinates in subsequent images this pairing can be done even if tracking is lost for a few images. This is the reason each cell has an image number index as well as coordinates, too identify if there was a loss in detection even just for one image. Through some conditions in the second for-loop the algorithm stops the tracking at the edge of the screen an tries to prevent newly detected cells from being confused with already detected ones.

```
function S=wDMHAPCF(same,D,a)
if  isempty(D)
    S=same;
return
end
x=find(~cellfun(@isempty,same));
x=x(1,end);
same{x+1}=[D(1,:) a];
D(1,:)=[0 0];
D = D(~all(D == 0, 2),:);
S=wDMHAPCF(same,D,a);
end
```

wDMHAPCF is a recursive algorithm that adds cell coordinates that did not match in connectCells with any previous ones at the end of the cell array same. Making them newly detected cells.

C. Playback

```
j=1;
for i = 1:numel(CM)
  A=CM{i};
  B=same{cellnbr};
  C=B(:,1);
  D=B(:,2);
 imshow(readimage(ds,i))
 hold on
 plot(A(:,1), A(:,2), 'rx')
 if j==size(B, 1) || j<size(B,1)...
 && i==B(j,3)
 plot(C(1:j), D(1:j), 'r',...
 'LineWidth', 2)
j=j+1
 end
 hold off
 pause(0.05)
i=i
end
```

All unnecessary variables are cleared and j is set to 1. The remaining section of the code shows a playback of every image in the given set while the cell cellnbr is being tracked.

III. Results

The first image of the cells from Dr. Wei Qiu's experiment is shown in Figure 2.

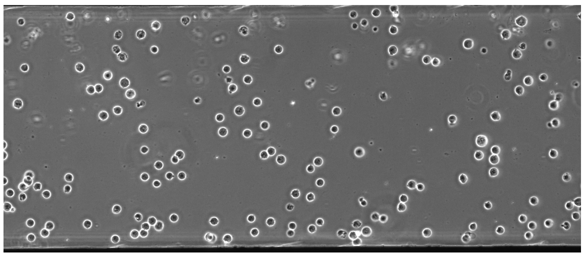


Figure 2. Still image of cell migration from left to right. This is the image that was used as a template for taking measurements. The blurred cells in the background are cells that are out of focus.

All the images were taken against the same background and thus the only thing that varies between them is the locations of the cells as well as some visual noise. The ideal scenario would be perfectly circular cells against a homogeneous background, however, the reality involves smaller particles and cells that are out of focus yet visible enough to interfere with the cells in focus. The histogram of the intensity distribution from this image can be viewed in Figure 3 and the pixel values inside one of the cells can be seen in Figure 5. The resulting thresholding value n and minimum and maximum radii can be viewed in Table 1.

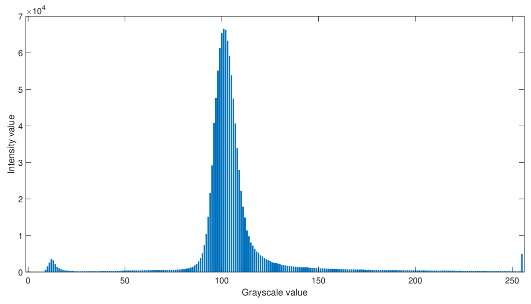


Figure 3. The histogram of the intensity distribution from the image in Figure 2.

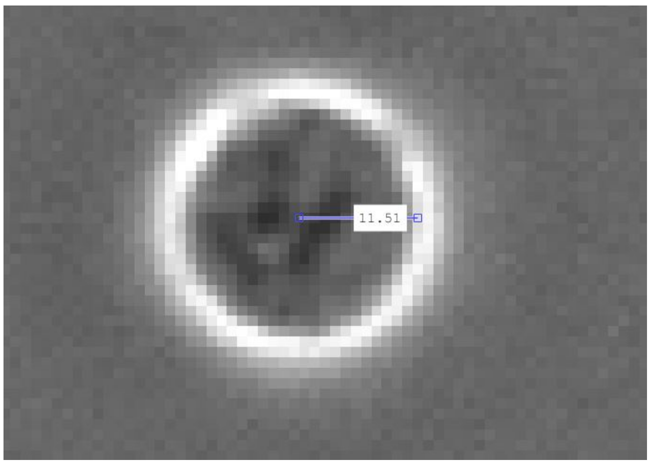
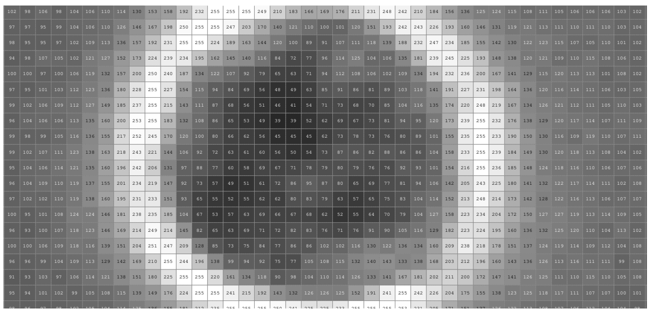


Figure 4. Zoomed in picture of a cell in Figure 2 using imtool. The approximate radius of the cell is measured with the tool ”Measure distance”. Note that this is only an example image and the value 11.51 was not used in the code. It was however using this method that the minimum and maximum cell radii were acquired.

The image in Figure 4 was acquired with imtool and resulted in the measurements of minimum and maximum cell radii a and b that can be viewed in Table 1.



102	98	106	98	104	106	110	114	130	153	158	192	232	255	255	255	249	210	183	166	169	176	211	231	248	242	210	184	156	136	125	124	115	108	111	105	106	106	106	103	102
97	97	95	99	104	106	110	126	146	167	198	250	255	255	247	203	170	140	121	110	100	101	120	151	193	242	243	226	193	160	146	131	119	121	113	111	110	111	110	103	104
98	95	95	97	102	109	113	136	157	192	231	255	255	224	189	163	144	120	100	89	91	107	111	118	139	188	232	247	234	185	155	142	130	122	123	115	107	105	110	101	102
94	98	107	105	102	121	127	152	173	224	239	234	195	162	145	140	116	84	72	77	96	114	125	104	106	135	181	239	245	225	193	148	138	120	121	109	110	115	108	106	102
100	100	97	100	106	119	132	157	200	250	240	187	134	122	107	92	79	65	63	71	94	112	108	106	102	109	134	194	232	236	200	167	141	129	115	120	113	113	101	108	102
97	95	101	103	112	123	136	180	228	255	227	154	115	94	84	69	56	48	49	63	85	91	86	81	89	103	118	141	191	227	231	198	164	136	120	116	114	111	106	103	105
99	102	106	109	112	127	149	185	237	255	215	143	111	87	68	56	51	46	41	54	71	73	68	70	85	104	116	135	174	220	248	219	167	134	126	121	112	111	105	110	103
96	104	106	106	113	135	160	200	253	255	183	132	108	86	65	53	49	39	39	52	62	69	67	73	81	94	95	120	173	239	255	232	176	138	129	120	117	114	107	111	109
99	98	99	105	116	136	155	217	252	245	170	120	100	80	66	62	56	45	45	45	62	73	78	73	76	80	89	101	155	235	255	233	190	150	130	116	109	119	110	107	111
99	102	107	111	123	138	163	218	243	221	144	106	92	72	63	61	60	56	50	54	73	87	86	82	88	86	86	104	158	233	255	239	184	149	130	120	118	113	108	104	102
95	104	106	114	121	135	160	196	242	206	131	97	88	77	60	58	69	67	71	78	79	80	79	76	76	92	93	101	154	216	255	236	185	148	124	118	116	110	106	107	106
96	104	109	110	119	137	155	201	234	219	147	92	73	57	49	51	61	72	86	95	87	80	65	69	77	81	94	106	142	205	243	225	180	141	132	122	117	114	111	102	108
97	102	102	110	119	138	160	195	231	233	151	93	65	55	52	55	62	62	80	83	79	63	57	65	75	83	104	114	152	213	248	214	173	142	128	122	116	113	106	107	107
100	95	101	108	124	124	146	181	238	235	185	104	67	53	57	63	69	66	67	68	62	52	55	64	70	79	104	127	158	223	234	204	172	150	127	127	119	113	114	109	105
96	93	100	107	118	123	146	169	214	249	214	145	82	65	63	69	71	72	82	83	76	71	76	91	90	105	116	129	182	223	224	195	160	136	132	125	120	110	104	113	102
100	100	106	109	118	116	139	151	204	251	247	209	128	85	73	75	84	77	86	86	102	102	116	130	122	136	134	160	209	238	218	178	151	137	124	119	114	109	112	104	108
96	96	99	104	109	113	129	142	169	210	255	244	196	138	99	94	92	75	77	105	108	115	132	140	143	133	138	168	203	212	196	160	143	136	126	113	116	111	111	99	108
91	93	103	97	106	114	121	138	151	180	225	255	255	220	161	134	118	90	98	104	110	114	126	133	141	167	181	202	211	200	172	147	141	126	125	111	110	115	110	105	108
95	94	101	102	99	105	108	115	139	149	176	224	255	255	241	215	192	143	132	126	126	125	152	191	241	255	242	226	204	175	155	138	123	125	118	117	111	107	107	100	101

Figure 5. Zoomed in picture of a cell in Figure 2 using imtool, showing the pixel values for each pixel. Most of the values in the cell interior are below 90.

Figure 5 showcases a further zoomed in version of the image in Figure 2. This time the tool ”Inspect pixel values” was used.

Table I
THE MANUALLY MEASURED VALUES OF THRESHOLDING VALUE N, MINIMUM CELL RADIUS A AND MAXIMUM CELL RADIUS B

Variable	Value
n	90
a	8
b	25

Application of the various image processing methods transformed the original image (Figure 2) into what is shown in Figure 6. This image was much better suited for cell identification, as can be seen in Figure 7, where imfindcircles has been executed.

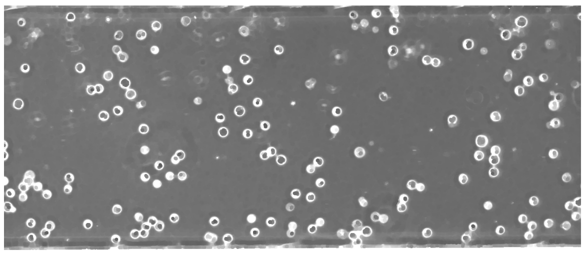


Figure 6. The final version of the image in Figure 2 after every image processing methods has been applied. The image is overall brighter and with the darker parts of the cell interiors removed the circular shapes can more easily be identified by the function imfindcircles.

The algorithm identifies almost all cells that are in focus in the image. There are exceptions though, some three or four cells are not identified, as can be seen in Figure 7.

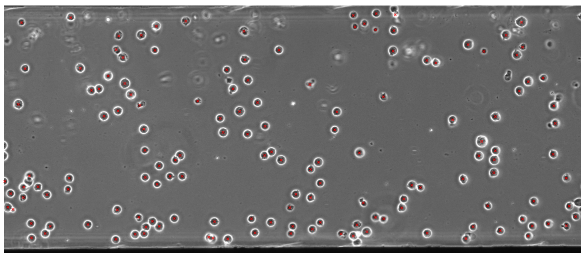


Figure 7. The identified center of masses plotted as red crosses over the image in Figure 2.

In Figure 8 the identification has been applied to the original image shown in Figure 2 for reference.

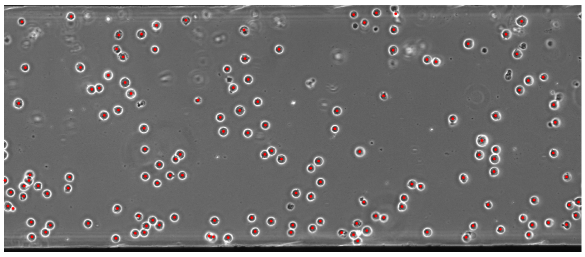


Figure 8. Identification of the cells based on the original image in Figure 2 without any image processing.

Figure 9 shows the tracking in action, in this case it is the first cell in the array that is being tracked across all 300 images of an image set. The red line shows the cell migration path. Although the cell encounters others in very close proximity along the way, the tracking still follows the correct cell for the whole duration.

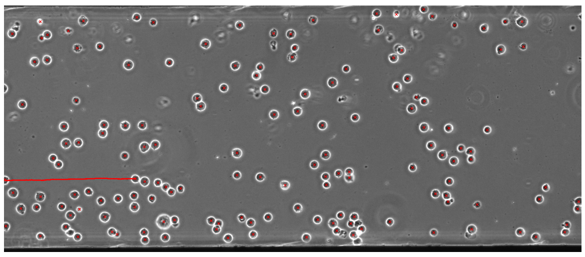


Figure 9. The first cell in the array tracked across all 300 images.

Figure 10 and Figure 11 show one of the cells first at its starting position in Figure 2 and then at its end point as the tracking has followed it for the entire set of images.

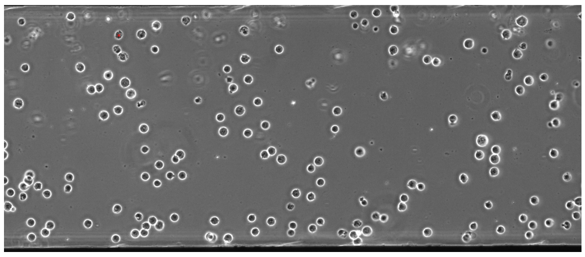


Figure 10. Cell 30 identified and plotted over Figure 2.

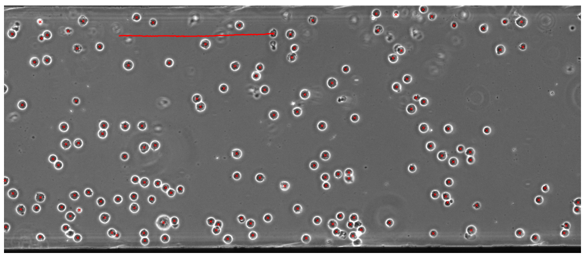


Figure 11. Cell 30 tracked across all 300 images.

Figure 12 and Figure 13 show how the tracking works when a cell migrates past the borders of the image. Cell 120 is tracked accurately until its circular shape is broken by the edge.

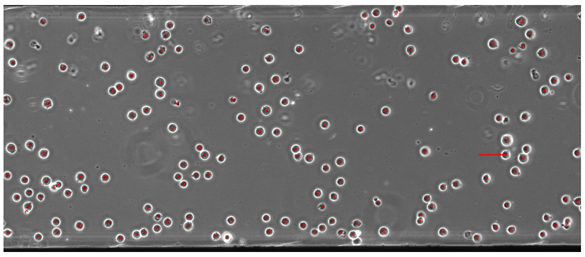


Figure 12. Cell 120 tracked across 50 images.

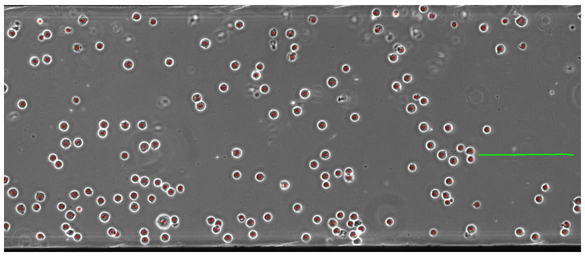


Figure 13. Cell 120 tracked across all 300 images. When the cell reaches the right edge of the screen the tracking stops. For showcasing purposes a green line has been implemented to showing the path of the cell until it exits the frame.

Figure 14 shows the tracked paths of all the cells that are identified across the 300 images.

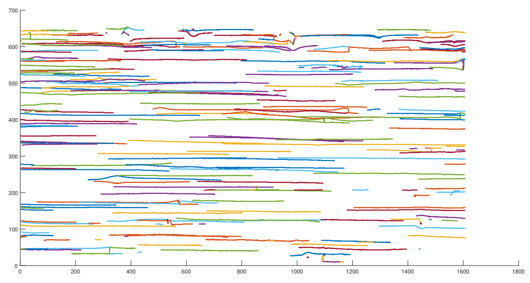


Figure 14. The paths of all the 293 cells detected across the 300 images.

IV. Discussion

The algorithm performs beyond our initial expectations, even if there naturally is room for improvement. imfindcircles finds 141 cells in the first image if it is left unprocessed, and 145 cells is found if the first image is processed. While the difference is small the processing also removes visual noise and out of focus cells that drift by in the image. By carefully reviewing the Figure 7 and Figure 8 it becomes apparent that after processing smaller and more irregularly shaped cells are detected.

Ideally we would like to reduce the amount of user input and completely automate the algorithm as a whole. Also we would like to generalize the algorithm to work on most types of images. For now the image processing works on images which have the approximate appearance of the images above. Gray background and black cells with a thick white brim. We do have a version of the image processing code which can find cells which are completely white. An improvement we would like to implement is a method which can handle both cases. From experience gained during this project we are confident that we could write new code that would display the cells even clearer.

The time required for identifying the cells is relatively short in this case around 3 minutes. The time depends on the amount of images as well as the amount of cells per image. If one or both of these factors are dramatically increased in other experiments, the amount of time needed could potentially become bothersome. The most time consuming part of the code is finding the cells. Most of the processing is fairly quick but finding the cells is time consuming with the current implementation of the code. Since the method for finding the cells is a preexisting method in MATLAB any further optimization would require an entirely new approach. Initially the data analysis part of the code was on par with the image processing for calculation time. But with a combination of iterative and recursive algorithms the computation time is now about 10 seconds, which we are fairly pleased with. While faster computation time is to be preferred the context in which this algorithm is used is not time sensitive. So in the end computation is not a limiting factor unless it becomes extreme which is unlikely in this case.

In the data analysis part of the code we mainly use recursion based solutions supplemented with simpler iterative loops. Everything done with a recursion based solution can be done with an iterative loop. The reason we use recursions so extensively is personal preference. If a task was a bit more complex we found it easier to implement and more perspicuous to read if recursion was used. It also felt more elegant somehow. For simple tasks iterative loops were used.

The identification process relies upon the cells being circular, imfindcircles uses a Circular Hugh Transform based algorithm to find circles. This works well for this experiment but would not be applicable if the cells had any other shape. It does not work perfectly in this case either, since a small number of cells are slightly deformed or not circular enough. Another issue that affects this algorithm is that many cells encounter other cells as they migrate, since they do not all travel with the same velocity. If two cells are in very close proximity of each other or they begin to overlap, their circular shapes become entangled and thus the function imfindcircles starts having difficulties finding the cells. The same problem arises from disturbances in the image such as unwanted background particles. Instances such as these are somewhat remedied by our data analysis part of the code which can compensate for lost or entangled tracking of the cells.

In spite of the problems, imfindcircles is very effective at identifying the cells. The identification process was initially based on a self-written function that would identify the cells regardless of shape, but imfindcircles worked so well for our purposes that it seemed unnecessary to solve a problem which already had a finished solution. There are nonetheless issues and shortcomings of imfindcircles and our data analysis that we would like to address. In the rare case that two cells overlap imfindcircles only finds one cell, and thus we lose track of one cell. Our data analysis corrects fairly well for this, but we believe that the implementation of a Kalman filter would be even better. By using a state space estimation of cells position and velocity vectors after initial identification tacking could be enhanced. An implementation of this was started but never finished due to time constraints. imfindcircles requires an approximate radius for the cells to work, this is the reason partly the reason for using the imtool function in MATLAB. This is one of the user input cases we would like to automate. While writing this report a potential method for identifying circular cells without user input was thought of, and will be explored in the near future. This would remove the need for inputting an approximate cell radius.

These potential improvements will hopefully be realised in the near future, as we have been fortunate enough to be offered a chance to continue working with Dr. Wei Qiu in a new project, again focusing label free cell tracking.
Being so application specific, it is doubtful that this algorithm could be used effectively for any other purpose than to track circular objects against a low noise background. It is therefore unlikely that the code itself could be used with malicious intent. However, the identification is not perfect, and the resulting data should be analyzed with this in mind. If the proper considerations are not taken, wrongful conclusions could be made which could potentially have harmful effects depending on the experiment. On the other hand, by avoiding the cell staining process the data more accurately represents the true values of the cell properties, from which benefits in development of drugs could hopefully be derived.

V. Conclusions

We did not have a clearly specified thesis for this project as we from the start knew that this problem was solvable. The questions was if we would be able to solve it, and we did. A conclusion we can draw is that it is possible to gain sufficient skills on your own in image processing to solve some fairly complex tasks.

VI. Epilogue

Special thanks to Dr. Wei Qiu from the Department of Biomedical Engineering at Lunds Tekniska Höskola for giving feedback through out the project, reviewing the report and helping us with Introduction A (Cell tracking) and D (Outset), and giving us the tremendous opportunity to continue developing the algorithm for use in a upcoming project.

Credit goes out to the user Stephen23 at MathWorks for the development of the Natural-Order Filename Sort algorithm (natsort) used in the sorting of filenames.

The authors of this report contributed in equal parts to the writing of this report and implementation of the algorithm.

References

[1] Ettinger A, Wittmann T. Fluorescence live cell imaging. *Methods Cell Biol.* 2014;123:77-94. doi:10.1016/B978-0-12-420138-5.00005-7[PubMed]

[2] Sanderson MJ, Smith I, Parker I, Bootman MD. Fluorescence microscopy. *Cold Spring Harb Protoc.* 2014;2014(10):pdb.top071795. Published 2014 Oct 1. doi:10.1101/pdb.top071795.," [PubMed]

[3] Robson AL, Dastoor PC, Flynn J, et al. Advantages and Limitations of Current Imaging Techniques for Characterizing Liposome Morphology. *Front Pharmacol.* 2018;9:80. Published 2018 Feb 6. doi:10.3389/fphar.2018.00080[PubMed]

[4] Pasternak MM, Strohm EM, Berndl ES, Kolios MC. Properties of cells through life and death - an acoustic microscopy investigation. *Cell Cycle.* 2015;14(18):2891-2898. doi:10.1080/15384101.2015.1069925[PubMed]

[5] Nikon's MicroscopyU. Introduction to Phase Contrast Microscopy. https://www.microscopyu.com/techniques/phase-contrast/introduction-to-phase-contrast-microscopy (Accessed 2022-05-29)

[6] Evident, Olympus. DIC Microscope Configuration and Alignment. https://www.olympus-lifescience.com/en/microscope-resource/primer/techniques/dic/dicconfiguration/ (Accessed 2022-05-29)

[7] MathWorks. 2022. Image Processing Onramp. Natick, Massachusetts: MathWorks. https://matlabacademy.mathworks.com/details/image-processing-onramp/imageprocessing (Loaded 2022-05-19).

[8] Chenouard, N., Smal, I., de Chaumont, F. et al. Objective comparison of particle tracking methods. *Nat Methods* 11, 281–289 (2014). https://doi.org/10.1038/nmeth.2808[Nature methods]