1.1 Abstract

The purpose of creating a cell tracking tool is to minimize file size and get accurate and high-resolution analysis from the video of the deformation of cells in a microfluidic channel. There are many conditions that can change the cell’s biological structure and can alter the phenotype. By using this fact to our advantage, many properties of a cell can be discovered. This attribute can yield an interesting development in the medical field, as the affected cells can be tested to see how they deform in comparison to the control cells. Being able to track the deformation of many cells in a small amount of time provides a vast amount of information, such as the cell relaxation time, the elastic modulus and maximum strain of a cell. The speed in which this method can provide diagnosis for diseases and progression is significant in comparison to previous methods. The tracking tool is used to extract the information needed and will output a variety of data. The secondary use for this tool is to refine and trim all unnecessary noise within the results. This is done by applying an array of filters to the video in real time. The application of filters not only reduces noise but also helps with one of the primary goals, file data reduction.

2.1 Introduction

The cell is the basis of life of all living organisms on Earth, complex and singular, that contain important information to carry out selective biological functions. To achieve their purpose, cells use their structure and biochemistry to their advantage to adapt to their surroundings [1]. One of these important properties of cells is their rigidity. Cell deformation is important for the detection and diagnosis of several diseases, including many cancers. The properties for cell deformation arise from the biological structure and mechanisms of the cell, such as cytoskeleton, lipid bilayer, cell membrane and cytoplasm [2]. Some diseases cause cell structure to change, specifically the cytoskeleton. These changes to the cytoskeleton can alter the cellular processes and biomechanical responses. The biophysical changes to the cell can be tested in a microfluidic channel, where cells can be deformed through shear stresses and pressure gradients without the need for contact [3]. This is important as the channel provides low stresses for higher strains. This Label-free marker is much more efficient and cost effective than the current medical fluorescent antibody trackers. For drug creation, measuring the cytoskeleton of the affected cells could provide a simple solution to screen and test cytoskeleton-acting diseases. The speed at which the cells can be analysed in a lab, and therefore a diagnosis would increase significantly [4]. The ability to provide diagnostics on a possible disease so quickly could help to save lives, or minimize damage caused by diseases, by treating them as early as possible. The effect of certain diseases on the rigidity of cells can be extreme and therefore provides solid and repeatable biological markers for finding diseases. A study found that metastatic cancer cells can be up to 70% softer than benign cells under the same conditions [5]. This research has also been suggested to control the quality of stem cells and has been gaining popularity in various flow fields for its approach of video microscopy [6]. To measure this deformation comparatively to unaffected cells, software is needed. The objective of this project is to create software which can track and measure important information from a video of cell deformation that you would otherwise be unable or incredibly difficult and time consuming to obtain using non-computerized methods. The benefit of this system is that it requires almost no human input to analyse the results. This not only reduces the time spent analysing the results, but also increases accuracy compared to human input. The highspeed analysis of cells requires very little complex equipment. All that is needed is a computer and cell tracking software.

Method and Theory

3.1 Background experiment

The method used to test this deformation uses a crossflow microfluidic channel that sends ~2000 cells per second through a high-speed camera at 7,500-260,000 frames per second. The microfluidic channel puts various stresses and strains on the target cell without contact and has a flow rate Q<100µL/min. The microfluidic channel was mounted above an inverted brightfield microscope with exposure time of 0.37–6.67 µs. The microfluidic channel has a height of 25µm and a width of 35µm. Using these dimensions, and by taking information learned from analysis, the scale of the video can be found using pixel measurements and therefore the velocity of the cells can be found [2]. The Microfluidic channel is a narrow channel with viscous liquid entering at both sides of the channel, represented below by the inlets. They meet in the centre and continue their flow to both outlets. The cell has large amounts of shear forces acting upon the cell from the centre to the outlets.

Graphical user interface

Description automatically generated with medium confidence

Figure 1: Diagram of the microfluidic channel, apparatus used for applying shear forces to the cells without contact. Here also shows the forces acting in the microfluidic channel - in this case we have a shear dominant regime, in which shear forces are most of the total force [2].

3.2 Method and Uses

To create the tracking tool that will analyse the deformation, first several processes must be completed on the video. These processes allow noise from the video to be reduced drastically, which cuts down the amount of data processed. This also reduces file size, which is important for the storage of data files in important databases. This is especially crucial in the medical field, where resources are limited, and massive amounts of data is being stored from a multitude of testing. To apply filters to the video, a package called, Open CV is used. The package is used for video editing and manipulation, as well as, tracking and contouring. The video is separated into individual images and edited to remove empty and unimportant frames. The editing process takes the size of the object in each frame and only saves frames with cells over the limit set. The pictures that pass the conditions set are then saved into a folder, as a jpeg, after being assigned a number, which identifies each frame. The code then loops through this folder, that reads and appends each image to an array. A new loop is started that repeats through the new array frame by frame performing several different operations, to allow for data extraction from the video. These operations include processes, such as thresholding, background subtraction and most importantly ellipse fitting. Each of these operations are performed individually on each image in the folder and then compiled together at the end after each image has been edited. While being compiled, the positions and radii of each cell are added to arrays, where they are stored. The information is then used to calculate the deformation ratio index:

Equation(1)

Where I is the perimeter of the cell, H is the height of the cell and W is the width of the cell [2].

3.3 Method in Depth

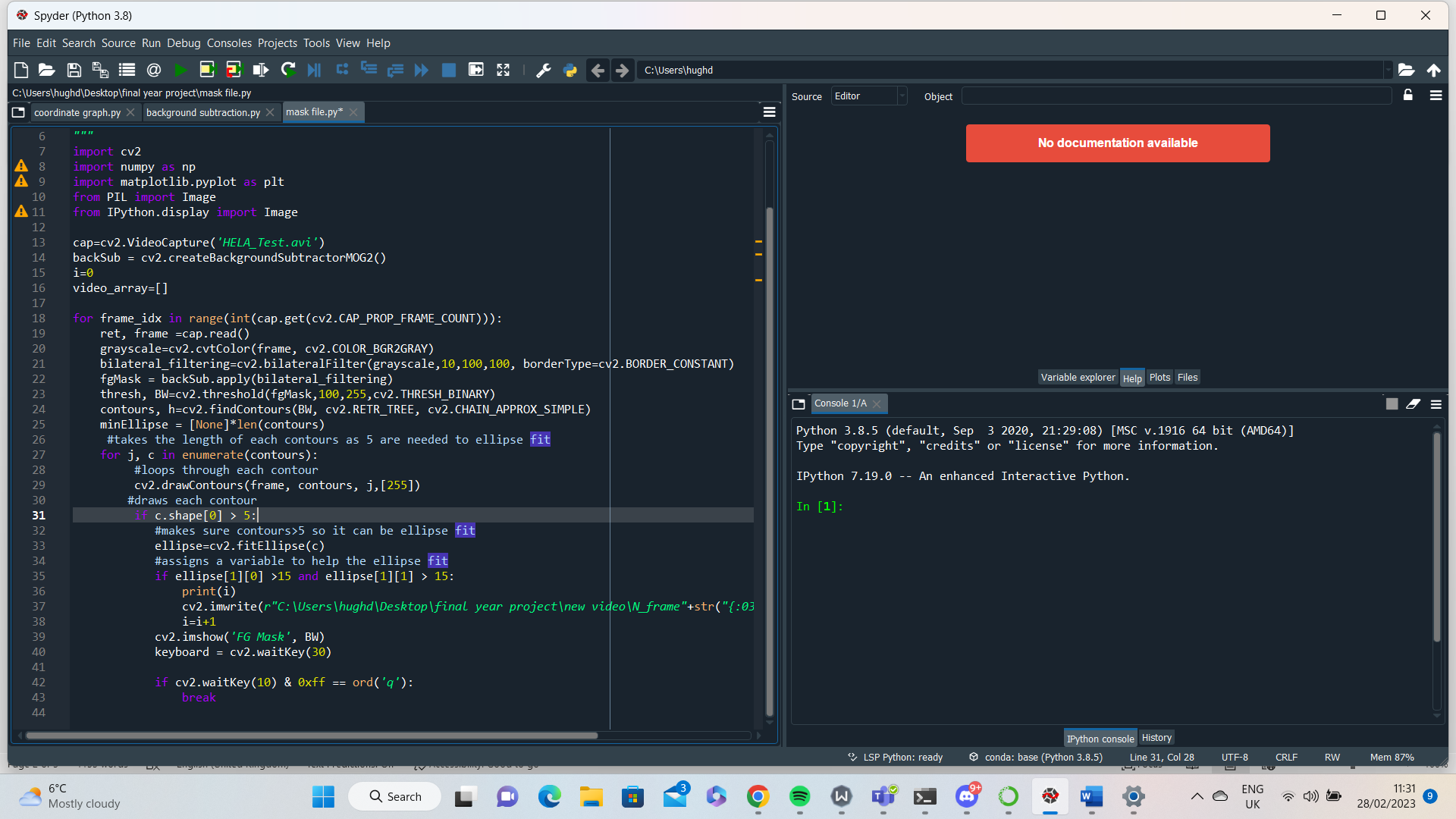


Figure 2: The first section of the video editing tool. This displays the first loop that repeats through every frame in the video and applies a multitude of filters.

The video analysis starts by reading the video into the program; in this case it is assigned the variable ‘cap’. The videos main loop starts, which iterates through each frame in the video. This allows the first filter to be added to the original video. Many operations needed for this tool cannot be applied to a colour video, additionally colour is not needed for the analysis of results, so by removing the colour, the size of the data and its clarity are improved. To achieve this cv2.cvtColour is used. This takes the frame as an argument and converts it to RBG to grayscale. Grayscaling an image is important when performing image recognition software, as changes in lighting and shadows can affect the results of tracking on colour images; grayscaling eliminates this [7].

The bilateral filter is performed next, which takes grayscale as an argument. The Bilateral filter is applied to grayscale instead of frame, because all filters are applied together, rather than individually. This is due to selected filters, which would require the image to be in a particular format. Bilateral filtering works by smoothing the image, while still retaining the edges. Pixels are compared by their colour value and geometric closeness. The program is biased to values that are closer, both spatially (its physical location) and its colour value [8]. This furthers the importance of the grayscale, as bilateral filtering works differently for coloured images than grayscale. The purpose of this filter is to try to eliminate small noise, called peppering by smoothing over it. Small noise, like in this case, dead cells and debris, is reduced during this step. One of the most important filters is the background subtraction. Each image is composed of a 2D array of pixel values, which indicate their colour, or in this case, grey level. The background subtraction takes the current image and compares it to the previous image. The values are then compared for their value and position. Open CV’s background subtraction uses an adaptive Gaussian mixture model, which benefits from the video’s simplicity. The model is made to adapt to changes in scenery, however, backgrounds including a variety of small details, for example, leaves and rain struggle to adapt [9]. Using the benefits of the model, in the case of the microfluidic channel, the background is stationary. When performing subtraction, the background will return a black screen, as there is no change between the current and compared frame. This is incredibly useful when trying to focus on a moving object in a video, as only the movement is highlighted. This also completely removes all noise that does not move, such as the peppering mentioned earlier. As shown in figure 2, two steps are needed to create a background subtraction. First is to create a variable that holds a function from open CV library that performs background subtraction, then the second step is to apply the background subtraction to the previous filter; in this instance, the bilateral filter.

As it stands the video produced is a grayscale cell going through a dark background. Gray can be confused and hard to see on the background. To add clarity to the video another filter called thresholding is applied. There are several useful types of thresholding like Global, Adaptive and Inverse. The objective of a threshold is to classify each pixel value as either ‘light’ or ‘dark’. This creates a large contrast between light and dark areas on the video. The cell, which was grey before, transforms into a white cell on a black background. Global thresholding would struggle to perform correctly on the video, as even small areas have been poorly lit, and the threshold would struggle greatly to perform an even thresholding. Due to this fact, the threshold will only work on some parts of the image. On the other hand, the Adaptive threshold is made to adapt to the light levels in the image and perform a smooth threshold over the entire picture. It solves this problem by taking spatial variations in the lighting into account [10]. Both Adaptive and Inverse binary are used in this code (inverse is for helping find the coordinate system for the video). The inverse binary works by flipping the ‘light’ and the ‘dark’ values, giving an inverted image. The Adaptive threshold is performed using a binary threshold argument, this changes all ‘light’ and ‘dark’ values into 0 (black) or 255(white). The Adaptive threshold is added to a background sub variable called fgmask. As shown in figure 2. The threshold code outputs two variables, the first outputs the type of thresholding used and the second is the thresholded image.

The next step is to outline the shape of the cell to fit a shape. Once fitted, the shape will be able to output a variety of information that will be useful for analysis. Contouring takes points that are the same pixel value all the way around the edge of an object and connects them into a line without repetitions. This is a significant reason that a binary threshold was necessary, as only pixel values that are equal can be contoured. Open CV function cv2.findContours takes the binary image from the last steps and outputs the number of contours on the shape contoured [11]. Cv2.RETR\_TREE is the retrieval argument. It specifies the way that the contours are found and adds them to a ‘hierarchy’ which is retrieved by the output variable h. The chain argument in this line of code describes a method, in which the tracing algorithm used to find the contours is described. The chain code is perfect for shape or pattern recognition software, this is due to its ability to find features, such as, perimeter, shape and centres. Finally, before the ellipse fit can start a list is initiated with the size of the total contours found. This list is filled with placeholders, which will be replaced in the following loop.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 3: This is the second part of the editing tool, where the ellipse fit loop is applied to edit empty frames out of the video.

The for loop is initiated by looping through the contours found in the image. Here enumerate is a built-in function that adds a counter to an iterative variable; in these circumstances, this refers to contours, which, through this line of code is given an index represented by j. This means that c refers to each individual contour performed on the cell. On each frame that has had an object contoured, the next line cv2.drawcontours draws an outline, visually showing each contour occurring. While being looped through there are a series of steps that allow all blank or empty frames to be eliminated. To start this procedure, information from fitting a shape to the contoured cells is required. The shape that most accurately describes a cell is an ellipse and therefore is the shape fitted. Ellipse fitting is only possible when 5 or more points on the cell are connected, so the if statement ensures that only shapes with this condition are fitted. The if statement must also be inside the for statement, as c.shape will return a different value for every contour that is looped through. c.shape returns an integer of the quantity of points in contour and once the program has checked the necessary requirements the ellipse is then fitted.

Cv2.fitellipse(c) returns a variable ‘ellipse’, which contains a variety of information. The variable is returned as a five-element tuple: (x , y)- which returns the centre points of the ellipse, (a , b)- that returns the major and minor radii of the ellipse and theta- the angle of rotation of the ellipse [12]. The next step uses this information, specifically (a , b). Inside both loops, another if statement is added that checks if the values of a and b are above a certain threshold. The final loop adds the condition that removes empty frames. There are a few methods that can be used to achieve this goal; one of which determines the area of every white pixel in the frame. The benefit of this method is that it does not require an ellipse fit and instead, would determine a more simplistic code. However, frames where there is a lot of small noise can add up to a large area over the entire frame, leading to frames that has much of the frame as broken up dead cells. Only the full larger cells are important for analysis, adding the condition that the radii of the cell must be over the limit set here as 15. Once all these conditions are met the selected frames are written into a folder called ‘new video’ and are indexed depending on the order each frame was added.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 4: This section has a separate program that recompiles the video that was edited previously.

‘New video’ contains all edited images indexed, however consists of only images. The purpose of the video compiler is to combine all these images and output them into a video file that can be run at any time without the need to run the code. First the directory that holds ‘new video’ is accessed and looped through similarly to contours, and each image is appended to a new array named video array. The dimensions of the outputted video are calculated using .shape, this outputs three variables: the vertical pixel height, the horizontal pixel height and the colour channels of the image. To output a video from the array, the variable ‘out’ is created. This takes in the parameters that will determine the format of the video. Cv2.VideoWriter\_fourcc(\*’DIVX’) is one parameter that informs the program the data blocks to use. Fourcc means four-character codes and contains 2-bit codes for data type and streams. Using these character codes, it specifies the type of video to output, such as compressed or uncompressed videos [13]. The other parameters are the frame rate and the dimensions of the video. Finally the array containing each image is looped through until i ,a counter variable, has reached the end of the video array. Each frame is individually written to the video created by the compiler.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 5: This is the start of the main section of code, in which all filters are reapplied, and the ellipse is fitted.

Now that the groundwork has been laid for the tools used, the next section of code is designed to perform calculations using the information gathered by the ellipse fit and display the results in a clear manner. As shown in figure 5 a large section of code has been reused from both the video compiler and the initial video editor. All the images from the folder are appended to an array and then looped through in the next loop. Within the next loop the same filters applied in figure 2 are performed to change the video into a binary image so an ellipse can be fitted again. Similarly, to Figure 3 the ellipse fit loop stays the same including the conditions for the size of the cells.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 6: This is the most crucial part of the code that outputs all necessary information for the analysis of the cells.

Using the ellipse fit and returning the five-element tuple, there are several different calculations that can be completed. The first step appends the first 4 elements in the tuple to different arrays. If printed these arrays would provide a coordinate or radius for every frame in the compiled video. Using the radii of the cell, the deformation index can be calculated [2]. Once calculated the deformation index is also added to an array to be stored. All arrays were defined at the start of the code. Similarly, to the deformation index the strain of the cell can be found using:

Equation (2)

Where H is the height of the cell and W Is the width of the cell. In the equation used in the code the radii returned by the tuple must be multiplied by two for the height and width. The velocity is determined by comparing the current frame with the previous frame. The x and y coordinate of the cell is determined and subtracted from the previous frames position. The change in position over a single frame gives the velocity in the x direction and the velocity in the y direction. To compute the total velocities the values are squared, added then square rooted, once again, these values are added to an array for storage. The problem with this method occurs when the program is first run. The positions of the cell at the start have nothing to be compared to, so the velocity in each direction would be calculated as just the starting position of the cell. To work around this problem a condition must be set. At the start of the code both delta x and delta y are defined as zero. The condition first asks if the previous frames position is 0, which only occurs at the very start of the program, and then sets the first velocity as 0. After the first frame, the previous frames value is now set as the position of the first frame and the velocity is now calculated correctly. The Deformation Index, the strain and the velocity arrays are all used at the end of the code when they are plotted.

Discussion and results

According to the literature the forces acting upon the cells in the microfluidic channel are heavily dependent on the flow rate of the viscous liquid flowing through the channel [2]. The forces affecting the cell are a compressive force Fc and shear force Fs.

(Equation 3)

Where FT is the total force acting on the cell. In this instance there is a shear dominant regime with high viscosity and low flow rates. This ensures that most of the force acting upon the cells are shear dominant, rather than the compressive force FC. The Deformation Index DI is the method in which the cells are tested for phenotype altering diseases. Knowing how a disease affects the phenotype of the cell can provide valuable information when screening a patient’s cells. Many of the variables returned through the code have non-SI units, either being pixel or frame related. It is simple to transfer these units into SI units, as long as it can be stated with certainty that the dimensions of each pixel position stay constant over the span of the video.

Graphical user interface

Description automatically generated

Figure 7: A graph to show Y-coordinate positions of microfluidic channel.

Firstly, we had to ensure a solid foundation for consistency in our results. To prove that our measurements were within reasonable error, a box was fitted in the four corners of the microfluidic channel and the coordinate of that fixed position was plotted. As shown in figure 7, the coordinates of the four corners of the microfluidic channel had negligible variation over the span of the video. This shows the consistency of the pixel position and can be compared to the real dimensions of the microfluidic channel to return real distances for values like velocity. For the frames to be translated into seconds, the frame rate of the video needs to be known. There are several methods of finding the frame rate in this circumstance, but perhaps the easiest is to time the duration of the video and then divide by the number of frames in the new video folder. Not only can real dimensions be found, but the consistency of the coordinate system can be used to highlight key events by tracking the cell through its entire path. This can be seen below:

Chart

Description automatically generated

Figure 8: Relationship between the y-coordinate and the other parameters over the span of 6 cells.

Figure 8 describes how each variable changes depending on its position in the microfluidic channel. As the x-axis is identical for each graph, it can be overlapped, but for the sake of clarity here, it is displayed on four separate axes. However, all graphs can still be overlayed to extract results. For the Y-coordinate graph, y stays almost entirely constant (±1) until the cell reaches the centre and starts travelling either up or down the y axis. Viscous liquid enters the channel from both left and right, where it meets in the centre. Due to the conflicting flow of viscous liquid in the channel the cell’s velocity decreases as it approaches the centre. This is shown by the comparison of velocity and y coordinates - the cell slows down as it approaches the centre of the channel, almost completely stopping. The cumulative force of both directional flows is approximately 0 at the centre. This is shown on the velocity graph, as when the cell arrives at the centre, the velocity reaches its lowest value. The average value for the velocity is 73.8 pixels/frames. This value is caused by the nature of the microfluidic channel, due to the viscosity of the liquid used for shear dominant regimes, the cells velocity is very variable, however much slower than inertial regimes where velocity is high, and strain is much lower. When the cell passes the centre, the liquid starts to apply shear force to the cell, which then stretches the cell; this can be evidenced by the strain and deformation index graphs, as the y-coordinate starts to slope, both the strain and the deformation index start to increase.

The average value for the Deformation index is 1.33 and its maximum value was 2.57. DIMax is close to the value found in the literature for shear dominant regime [2]. Shear dominant regimes focus on creating as much strain on the cell as possible to obtain clearer differentials between infected and control cells. The strains average value is 0.130 and has a maximum value of 0.44. Comparably to the Deformation Index, the strain’s value is dependent on the conditions of the liquid in the channel, however the maximum value is nearly double that of the literature. This can be explained by the cell’s properties, possibly having a softer cytoskeleton than those used in the study. Another factor is the conditions of the microfluidic channel - the higher the viscosity and lower the flow rate, the higher the strains achieved should be. The strain can be used to find the elastic modulus of the cell. This is done by fitting the time dependent Kelvin-Voigt model. The model is comprised of a linear spring and a viscous element in parallel. The cells continue to be stretched and the velocity continues to increase as the cells get closer to the outlets of the channel. The velocities highest peaks are when a cell event is at its end. This is due to the fact the force reaches a maximum at the edges of the microfluidic channel and furthest away from the centre. The maximum value for the strain and DI occurs at the edge of the outlets too; this information agrees with the literature [14]. The cells strain and DI are factors that do not include the original size of the cell or the circularity. This could mean, that some cells have a naturally higher DI, if the circularity of the cell is low. Therefore, it’s more important to record the change in DI over the cell event, to output a true representation of the cell deformation. The force applied by the liquid in each direction is constant, however due to the opposing direction of the fluid, as the position approaches the centre, the force applied is decreased. As the cell passes the centre the force applied is increased again at the same rate. In the case that the velocities shape isn’t entirely quadratic, it is likely that the software was analysing two or more cells at once. This is also the explanation for the anomalous results, such as the plots at 150 frames on all graphs and the short cell event, at approximately 275 frames. The Deformation index has an upper limit on cell deformation - any cell with a DI of over 2.84 is past the critical boundary of elastic deformation. If any cell has a value > 2.84, then it should be ignored, however in this instance no cell crosses the boundary. The cells used for analysis were MCF7 metastatic breast cancer cells. These cells are known to be softer than control cells and have high DI values [15].

Conclusion

This research has huge potential in the medical field. Testing for parameters like the deformation Index and deriving other useful parameters, such as elastic modulus and relaxation times, is a fast method for identification and diagnosis of deadly or life altering diseases. This method is also a reliable tester for the quality of stem cells and can be used to provide testing and quality control. The cell tracking tool is capable of reading a video and applying many different filters and shape fits to extract important properties from these cells under shear stress. The tool is adaptable to other videos, as long as the directory, where the video is stored is entered correctly at the top of the code. One of the goals of this project is to reduce the file size of the original video. This was very successful, as the original video, with a size of 813,017 KB was significantly reduced to 5,667 KB, decreasing the file by a factor of 143. This tool created to analyse the cells, works well and outputs the most crucial parameters from the data, however there are improvements that can be made. Firstly, the code should be able to cut or output results correctly from cell events that have more than one cell at a time. This would stop results from overlapping and create more accurate results. Taking circularity into account for each cell would also be an improvement. This would allow the deformation results to be put into context and would create more precise and repeatable results. An additional enhancement is the accessibility of the code. It would be an improvement to the code if you could input a specific cell event and it would output all the graphs and parameters for that specific cell event. This would be extremely useful for analysing certain cell events that require a more in-depth analysis. This would also decrease overload of information, if the video under analysis has hundreds or thousands of cells.

Reference List

[1] Zheng, Yi, et al. "Recent advances in microfluidic techniques for single-cell biophysical characterization." *Lab on a Chip* 13.13 (2013): 2464-2483.

[2] Armistead, Fern, Gala de Pablo, Julia, Gadelha, Hermes Augusto Buarque orcid.org/0000- 0001-8053-9249 et al. (2 more authors) (2019) Cells Under Stress : An Inertial-Shear Microfluidic Determination of Cell Behavior. Biophysical Journal. pp. 1127-1135. ISSN 0006-3495

[3]Otto O., P. Rosendahl, ., J. Guck. 2015. Real-time deformability cytometry: on-the-fly cell mechanical phenotyping. Nat. Methods. 12:199–202, 4, 202.

[4] Gossett, D. R., H. T. Tse, ., D. Di Carlo. 2012. Hydrodynamic stretching of single cells for large population mechanical phenotyping. Proc. Natl. Acad. Sci. USA. 109:7630–7635

[5] Cross, S. E., Jin, Y.-S., Rao, J. & Gimzewski, J. K. Nanomechanical analysis of cells from cancer patients. Nature nanotechnology 2, 780–783 (2007).

[6] Cha, Sukgyun, et al. "Cell stretching measurement utilizing viscoelastic particle focusing." *Analytical chemistry* 84.23 (2012): 10471-10477.

[7] Kanan, Christopher, and Garrison W. Cottrell. "Color-to-grayscale: does the method matter in image recognition?." *PloS one* 7.1 (2012): e29740.

[8] Tomasi, Carlo, and Roberto Manduchi. "Bilateral filtering for gray and color images." *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE, 1998.

[9] Piccardi, Massimo. "Background subtraction techniques: a review." *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*. Vol. 4. IEEE, 2004.

[10] Bradley, D., and G. Roth. "Adaptive thresholding using the integral image. J Graph Tools 12 (2): 13–21." (2007).

[11] Xie, Guobo, and Wen Lu. "Image edge detection based on opencv." *International Journal of Electronics and Electrical Engineering* 1.2 (2013): 104-106.

[12] Rosin, Paul L. "Further five-point fit ellipse fitting." *Graphical models and image processing* 61.5 (1999): 245-259.

[13] Chang, Xu, Jian Wu, and Shanshan Pei. "Research on Recognition and Carving Techniques of Monitoring Video Fragments based on Structural Features." *2018 3rd International Conference on Automation, Mechanical Control and Computational Engineering (AMCCE 2018)*. Atlantis Press, 2018.

[14] Rodrigues, Raquel O., et al. "A simple microfluidic device for the deformability assessment of blood cells in a continuous flow." *Biomedical microdevices* 17 (2015): 1-9.

[15] Landwehr, Grant M., et al. "Biophysical analysis of fluid shear stress induced cellular deformation in a microfluidic device." *Biomicrofluidics* 12.5 (2018): 054109.