Tracking DNA Damage Repair in Cancerous Cells

A Major Qualifying Project
Submitted to the faculty of
Worcester Polytechnic Institute
in partial fulfillment of the requirements for the
Degree in Bachelor of Science in
Bioinformatics and Computational Biology
By

Michael Corace

Alyssa Nicolella

Date: 5/18/2020

Project Advisors:

Professor Amity Manning, Advisor

Professor Dmitry Korkin, Advisor

Abstract

This project seeks to understand the dynamics of DNA damage repair in cancerous cells utilizing preexisting software and novel analysis approaches. Current analysis tools allow biologists to independently track cell movement and count damage sites, but not simultaneously. Our methods identified a slight increase in damage frequency over time in motile cell populations. We suggest utilizing cell morphology rather than coordinate tracking for further advancements in this area.

1. Introduction & Background

1.1 DNA damage and analysis

The DNA damage response involves the initiation of a chain of events that begins with the identification of problematic sites by kinases such as ATM and leads to recruitment of several proteins to the site of DNA damage (Khoronenkova et al., 2015). Proteins that are recruited to sites of DNA damage include CHK2 and 53BP1. The recruitment of some Damage response proteins is robust and leads to the formation of defined foci that can be detected using fluorescent tags or antibody-based detection of DNA damage response proteins such as 53BP1 (Cohn et a., 2008). The formation of such foci can be used as a surrogate to identify individual sites of DNA damage within cells. As DNA damage is repaired, DNA damage response and repair proteins are removed from DNA.

Historically, fixed cell imaging of successive time points following acquisition of DNA damage has been used to investigate the capacity of individual cellular conditions of DNA damage repair where the dissipation of DNA damage associated foci is an indication of the DNA damage repair process and differences in the time for foci to resolve indicates differences in damage repair capacity. A significant limitation of this approach is the inability to determine the metrics of DNA damage occurring and being repaired. Analyzing fixed time images only gives information at that specific time point. However, it is important to obtain data concerning the timeline of DNA damage to repair, as well as information regarding the percentage of repair.

To address this limitation, live cell imaging can be employed to follow single cells over time. These approaches make use of DNA damage proteins tagged with Green-Fluorescent Protein (GFP) or Red-Fluorescent Protein (RFP), like fixed cell imaging, monitor foci formation and dissipation as an indication of DNA damage acquisition and repair, respectively. While this approach has the ability to analyze the full timeframe of damage instead of fixed cell imaging, its application is limited because of the issue of tracking both individual cells over time and the foci within them simultaneously. The cells move throughout the field of view while their internal components show mobility as well, requiring the tracking of motile objects within larger motile objects. The current method involves having an observer review the videos created and record the various metrics manually. Manual observation is time consuming and inconsistent, so our goal was to develop a software pipeline to record these metrics automatically. A computerized process would speed up analysis, have a higher degree of consistency, and be able to collect more types of data than a human observer.

In this project, we attempt to address this limitation by altering and combining different pipelines supplied by the CellProfiler program that are used to identify different aspects of cells from still images. We also expand upon the CellProfiler pipeline to construct our own program that utilizes distance matrices to link foci across timepoints.

1.2 CellProfiler

CellProfiler is an open-source software developed by the Broad Institute of MIT and Harvard. This tool is utilized by biologists to quantify and analyze data from biological images in a highly-flexible, modular process. Users interact with a Graphic User Interface (GUI) to produce customizable analysis pipelines. These pipelines are composed of sequences of premade modules that handle various image processing functions. CellProfiler is specifically designed to allow users to fine tune their pipelines to specific phenotypes and conditions of individual datasets. This software allows users to extract more information more efficiently than can be collected by eye (McQuin C et al., 2018).

There are other softwares currently available for biological image analysis, such as ImageJ, Fiji, and Icy. CellProfiler stands out from these competing softwares due to its flexibility in analysis pipeline customization and history of successful analyses. This platform was developed with an emphasis on automating analysis of many thousands of high-throughput images, while some other similar softwares, such as Fiji, are better designed for analyzing large, multi-dimensional images. CellProfiler was chosen as the analysis software for this project due to these strengths, in addition to its current use in the Manning Lab (Eliceiri, K et al., 2012).

CellProfiler distributes premade analysis pipelines and example datasets that provide a baseline for standard analyses. Researchers may expand upon and adjust these pipelines as they see fit. Example pipelines include processes such as cell identification, cell tracking, and speckle identification. While these pipelines are often very comprehensive for the provided data, adjusting the pipelines to one's own, often messy, data can be confusing.

For the purpose of this study, the team primarily focused on the cell tracking and speckle identification pipelines. The cell tracking pipeline provides capability to track individual cell movement over time. The speckle identification pipeline identifies and counts all foci present in individual images. The parameters of both pipelines are highly dependent on the dataset at hand. The user must be especially careful when tuning parameters in the speckle identification pipeline to avoid the segmentation of nuclei or identified speckles. Improper assignment of these values may lead to the over or under identification of foci and cells within the dataset. While both pipelines are standardly distributed, there is currently no pipeline capable of running the two processes in parallel. Thus, there is no capability to identify foci and track their movement and lifespan over time. This study aims to produce methodologies to address this current lack in functionality.

2. Methods

2.1 Pipeline overview

Figure 1 provides an overview of our data generation and analysis process. We first collected data generated in Professor Manning's lab. A further description of this data may be found in section 2.2. Next, we identified damage foci within individual dataset images using an optimized speckle identification CellProfiler pipeline, as described in section 2.3. We then performed preliminary data exploration on the data files outputted by CellProfiler. The methodology and results of this analysis may be found in section 3.1. Finally, in section 2.4 we discuss our attempted methods for implementing cell tracking on the CellProfiler output files using a Euclidean distance matrix.

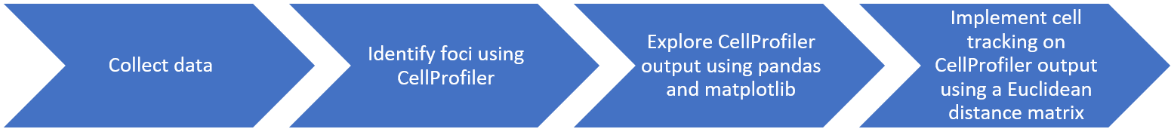
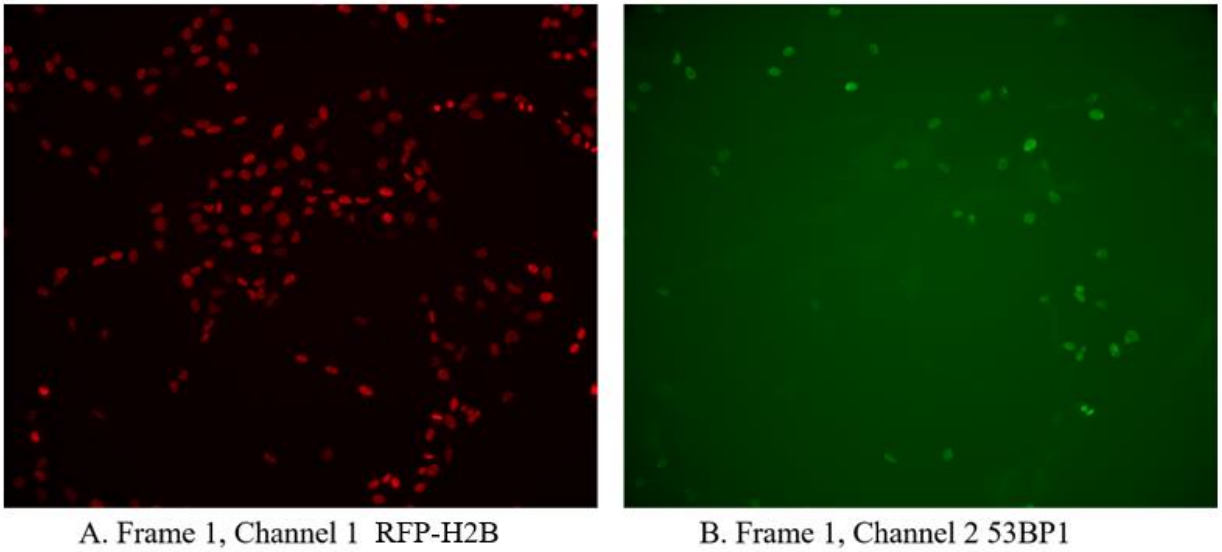

Collect data
Identify foci using CellProfiler
Explore CellProfiler output using pandas and matplotlib
Implement cell tracking on CellProfiler output using a Euclidean distance matrix

Figure 1: Pipeline overview

2.2 Overview of data

The data for this project originates from ND2 video files generated using a “Zyla cMOS camera mounted on a Nikon Ti-E microscope with a 60x plan Apo oil immersion objective, or a 20x CFI Plan Four objective” (Navarro-Serer B et al., 2019). These video files cover one quadrant recorded in the overall sample, which was recorded for 36 hours. TIF files were extracted for each timepoint recorded, resulting in both a “red” and “green” channel image for each timepoint. The first channel identifies the fluorescent protein RFP-H2B, a fusion protein of red-fluorescent protein and the histone protein H2B (Fig 2A). Histones work to package chromatin together within the nucleus, making it an effective marker for identifying morphological properties of nuclei (Kobiyama). The second channel picked up by the microscope identifies GFP-53BP1 (Fig 2B), a fusion protein of 53BP1 and the marker green-fluorescent protein. 53BP1 is a signaling protein used by cells to identify locations of DNA damage by reading a specific histone code signaled by damaged chromatin (Panier et al., 2014). Attaching GFP to 53BP1 is an effective way to take advantage of a functioning cellular mechanism and use it as a reporting method (Miwa et al., 2013). When 53BP1 is drawn to sites of damage, the high concentration of GFP in the area appears in the images as a point of high intensity called a focus. Tracking these foci allow researchers to understand the dynamics of DNA damage and repair within the cancerous cells.



A. Frame 1, Channel 1 RFP-H2B

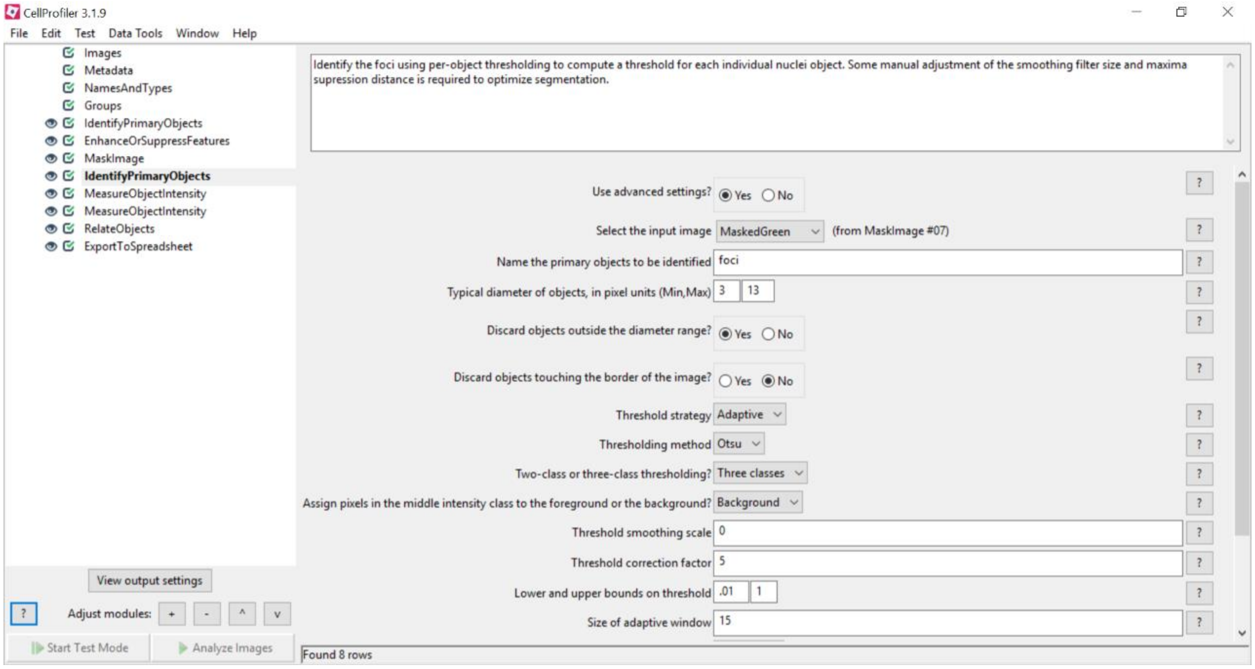
B. Frame 1, Channel 2 53BP1

Figure 2: Channels showing fluorescence of RFP-H2B (A) and GFP-53BP1 (B)

2.3 Foci identification using CellProfiler

The CellProfiler "Speckle Counting" pipeline was used as a starting point for identifying damage foci within the dataset. According to the CellProfiler documentation, "this pipeline shows how to identify smaller objects (foci) within larger objects (nuclei) and how to use the Relate module to establish a relationship between the two as well as perform per-object aggregate measurements (such as number of foci per nucleus)" (McQuin C et al., 2018).

Many object identification criteria were adjusted in order to optimize this pipeline to identify nuclei within the study's dataset. Figure 3 displays an example of the CellProfiler interface where parameters are adjusted. The first parameter adjustment for optimizing nuclei identification was changing the threshold strategy from global to adaptive. This was changed because the intensity of nuclei varied greatly within the study images. Next, the method for separating clumped objects was set to separate by shape, which eliminated the potential for nuclei to be split and counted as unique cells. Finally, the object diameter threshold was adjusted to range from 30 to 80-pixel units to reflect the scaling of images, where a single pixel corresponds to 0.33um and a typical cell has a diameter of about 20um.



Identify the foci using per-object thresholding to compute a threshold for each individual nuclei object. Some manual adjustment of the smoothing filter size and maxima supression distance is required to optimize segmentation.

Use advanced settings? ● Yes ○ No

Select the input image MaskedGreen (from MaskImage #07)

Name the primary objects to be identified foci

Typical diameter of objects, in pixel units (Min,Max) 3 13

Discard objects outside the diameter range? ● Yes ○ No

Discard objects touching the border of the image? ○ Yes ● No

Threshold strategy Adaptive

Thresholding method Otsu

Two-class or three-class thresholding? Three classes

Assign pixels in the middle intensity class to the foreground or the background? Background

Threshold smoothing scale 0

Threshold correction factor 5

Lower and upper bounds on threshold .01 1

Size of adaptive window 15

Found 8 rows

Figure 3: CellProfiler parameter adjustment interface

Adjustments were also made to optimize the identification of foci within the observed nuclei; many of these adjustments were performed due to the disparity in object size between the sample dataset and the study dataset. First, the default feature size parameter for enhancing or suppressing features was decreased from 10 to 4. Next, the diameter size threshold was adjusted to restrict foci size to remain between 3 and 13-pixel units. From a lower limit, this adjustment ensures that detected foci are greater than one pixel in size, which will reduce the detection of random noise and or fluctuations in intensity within the image. From an upper limit, this ensures that cells containing so much damage that they would no longer be alive are excluded from the pipeline output. By default, objects identified along the border of their parent objects were set to be excluded. In this case, the parent object is the observed nucleus and the child object is the foci. This setting was turned off because foci are often brought to the cell boundary for repair. The fluorescence intensity of objects in the green channel of the images falls into three classes: background outside of the nucleus, background inside of the nucleus, and damage foci. Due to this multitude of classes and the variation in foci intensity between cells, adaptive intensity thresholding was also utilized for foci identification. Next, the default pipeline specifies the use of the Otsu method of thresholding. However, this method assumes that at least 50% of the image is covered in objects, which is not true in the study dataset. Thus, the threshold correction factor was adjusted to 5 to ensure that foci are appropriately identified. Finally, the default declumping setting was removed because foci should be a far enough distance from one another to negate the necessity for declumping.

Visualizations of identified nuclei and foci may be seen as the pipeline is running; an example of this may be seen in Figure 4. The final output of the pipeline is composed of three comma separated (csv) files: Nuclei.csv, Foci.csv, and Image.csv. Nuclei.csv contains a row for each nucleus identified in each image. Each row contains data about its respective nucleus, including the positional coordinates of the nucleus and the total number of foci associated with the nucleus. Foci.csv provides more detailed data on the individual foci identified in each frame. Each row corresponds to a unique foci and contains data such as the positional coordinates of the foci and its parent nucleus identity. Finally, Image.csv is a summary document that lists data such as the total number of nuclei and foci in each image. While this information is very helpful, the object labeling of data points within Nucleus.csv and Foci.csv is relative to the parent image. Because of this, the foci labeled as “1” in the first image is likely not the same as foci “1” in the second image. This is a problem that will be addressed in section 2.4.

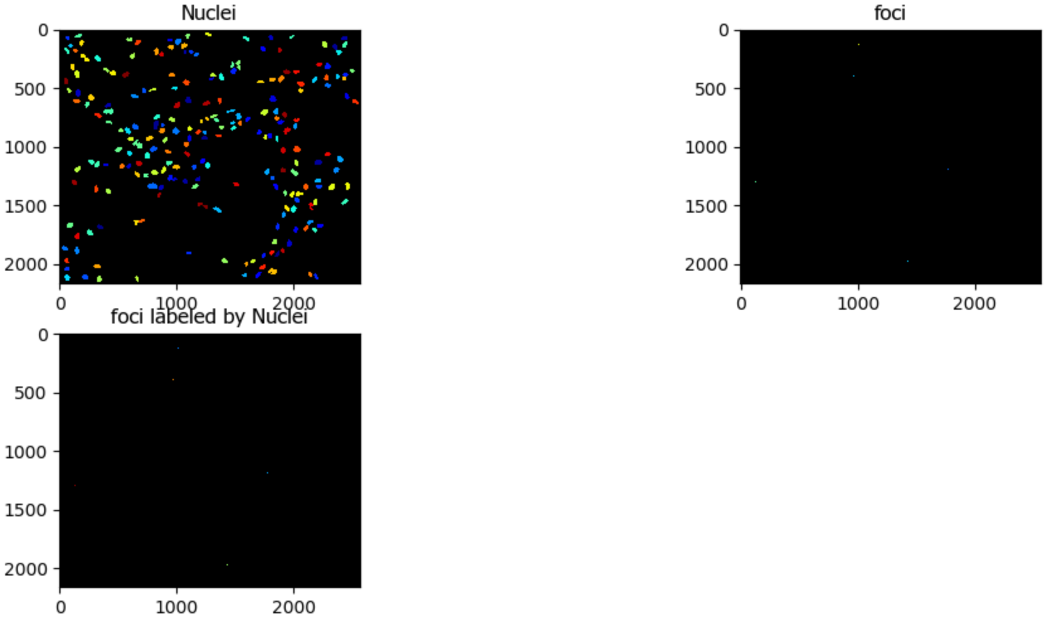


Figure 4: CellProfiler pipeline foci visualization

2.4 Matrix manipulation for cell tracking

In order to track the individual damage foci across each frame, we had to develop a method of matching each foci from one frame to the corresponding foci in the prior frame (Fig 5A).

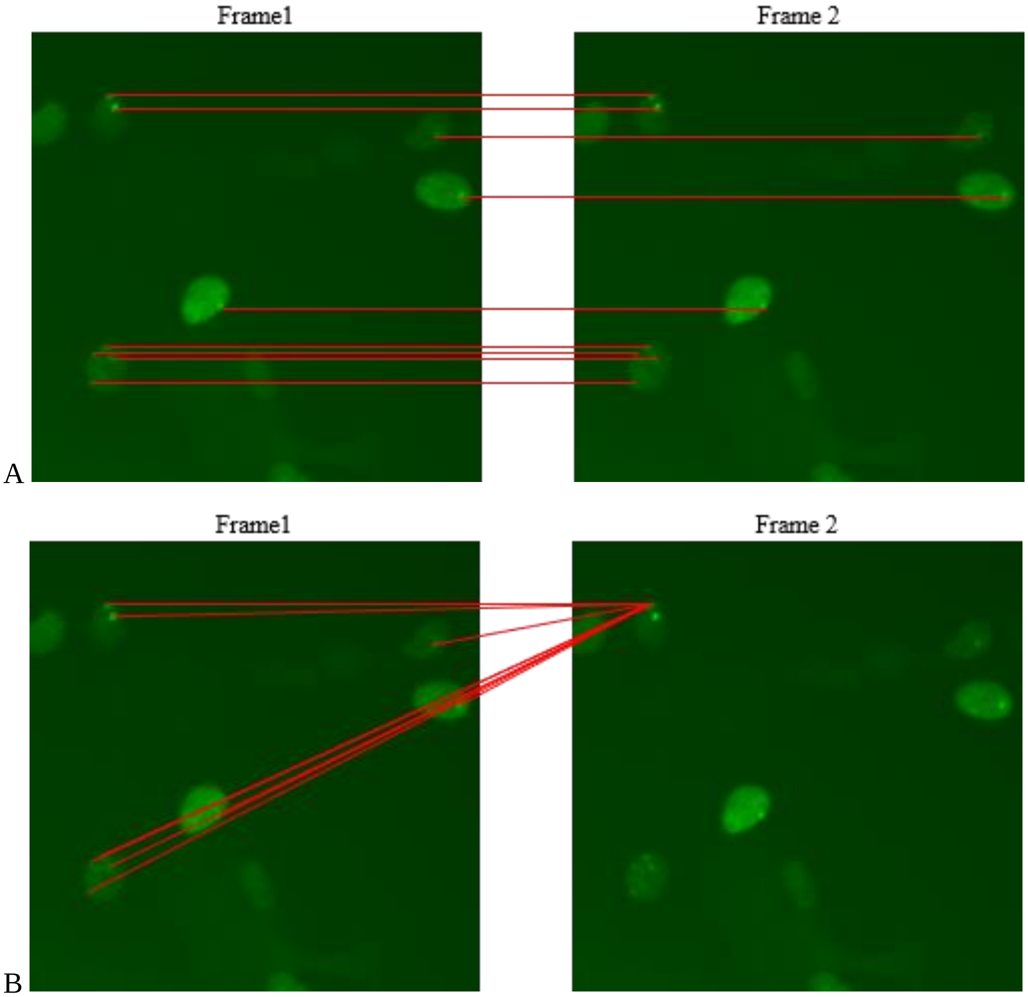
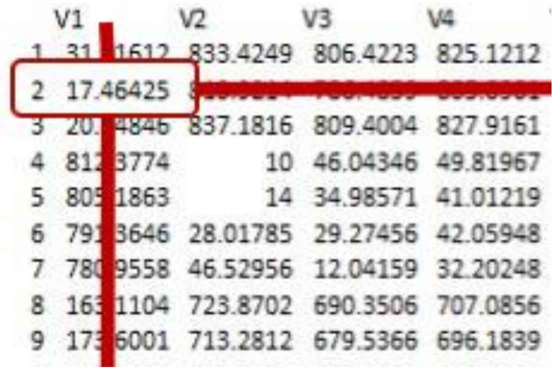


Figure 5: A) Visual mapping of foci from frame 1 to frame 2 B) Comparison of individual foci across frames

To achieve this result, we instituted a Euclidean distance matrix (Albanie S, 2019) to compare the X-Y coordinates of foci in consecutive frames (Fig 5B). The pipeline was built and run in R, using the CSVs generated from Cellprofiler as mentioned earlier. An R script was developed to take the CSVs as input and output a matrix with each unique foci, the first frame they enter on, and the last frame they are identified on. Using the spatial coordinates, we built a matrix that holds the absolute distance between each foci on one frame to each foci on the next frame. This information is stored in a data.frame, allowing for simple search for and elimination of values. The distance between the X-Y coordinates of Foci1 in Frame 2 and the X-Y coordinates of every foci in Frame 1 are listed in one column of a matrix, and the minimum of these distances is identified as the displacement from of Foci1 from Frame 1 to Frame 2 (Fig 6). The displacement is calculated by $\Delta D = \sqrt{[(\Delta X)^{2} + (\Delta Y)^{2}]}$, where ΔX is the difference in X coordinates between frames and ΔY is the difference in Y coordinates between frames.

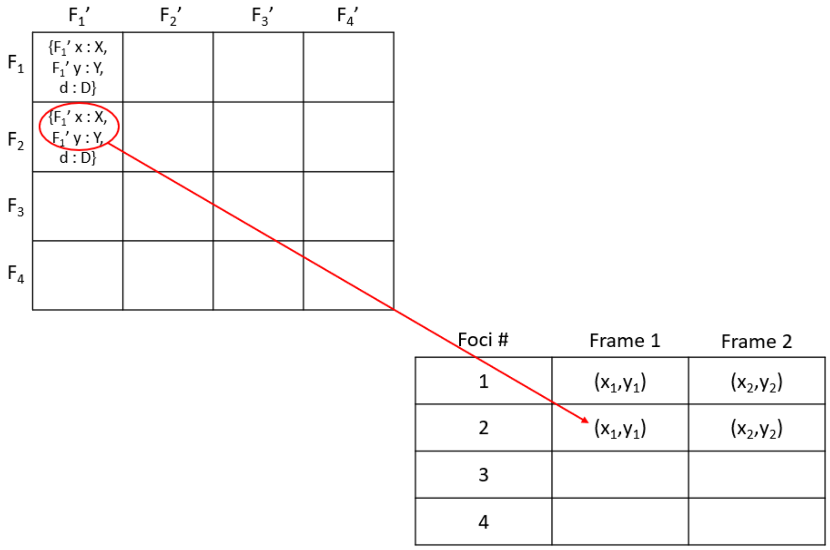
We decided on using a Euclidean distance matrix to compare the foci because the foci tend to appear in clusters within nuclei but can appear or disappear at any point when damage occurs and is repaired. A Euclidean distance matrix works well for this scenario for a few reasons. Firstly, it is simple to eliminate foci options because there would be a large difference in displacements between foci outside the cluster and those within. Additionally, when matching the least-displaced foci, an unmatched foci from a previous frame is an obvious case of repair. Similarly, since DNA damage is unlikely to occur at the exact same spot that was just repaired, an unmatched foci in a later frame can be assumed to be a new point of DNA damage.



	V1	V2	V3	V4
1	31[illegible]1612	833.4249	806.4223	825.1212
2	17.46425	[illegible]	[illegible]	[illegible]
3	20.[illegible]4846	837.1816	809.4004	827.9161
4	81[illegible]3774	10	46.04346	49.81967
5	80[illegible]1863	14	34.98571	41.01219
6	79[illegible]3646	28.01785	29.27456	42.05948
7	78[illegible]9558	46.52956	12.04159	32.20248
8	16[illegible]1104	723.8702	690.3506	707.0856
9	17[illegible]6001	713.2812	679.5366	696.1839

Figure 6: Example of a Euclidean distance matrix between two frames

To track the location of each foci more robustly and efficiently, the matrix cell containing the distance between two foci also contains the X-Y coordinates of the foci in the newer image. This is achieved by storing all three data points in a dictionary within the matrix cell (Fig 7). When the distance matrix has been fully constructed, the coordinates of the foci with the lowest distance score are extracted from the distance matrix and placed in a satellite matrix containing each foci's frame-to-frame coordinates. This allows for later reconstruction of each foci's movement path overtime. The final data.frame contains every unique foci as the first dimension, with the X-Y coordinate pair of that foci in every frame as the second dimension. Frames that do not contain a given foci are assigned a NULL value for that dimension. A summation of values in the first dimension can determine how many consecutive frames a foci is identified on, and data about the amount of foci per frame can be found with a summation of values in the second dimension.



	F_1'	F_2'	F_3'	F_4'
F_1	{F_1' x : X, F_1' y : Y, d : D}			
F_2	{F_1' x : X, F_1' y : Y, d : D}			
F_3				
F_4				

Foci #	Frame 1	Frame 2
1	(x_1,y_1)	(x_2,y_2)
2	(x_1,y_1)	(x_2,y_2)
3		
4		

Figure 7: Matrix mapping method

3. Results

3.1 Preliminary data exploration

Before performing Euclidean distance matrix mapping, the team explored the raw output from the CellProfiler pipeline using the python libraries: pandas and matplotlib. The data used in these analyses consist of the first 216 timepoints (approximately 18 hours) of the full dataset. Figure 8 displays the average number of foci per cell displaying damage over time. This graph indicates that the average number of foci within each damaged cell spans a range between one and three foci per cell. There appears to be a slight increase in the average number of foci within damaged cells over time. The average number of foci per cell generally remains between one and two foci per cell within the first 100 timepoints of the dataset, however this range increases to approximately two to three foci per damaged cell in the last 100 timepoints. Additionally, there are two timepoints, frames 97 and 207, where there is no damage or very little damage present. However, the frames proceeding and following these images appear to display levels of damage that are consistent with the overall population damage. This is potentially due to the images being captured at a time when the microscope was not properly focused, preventing foci from being identified by CellProfiler. These data points may therefore be considered dropouts.

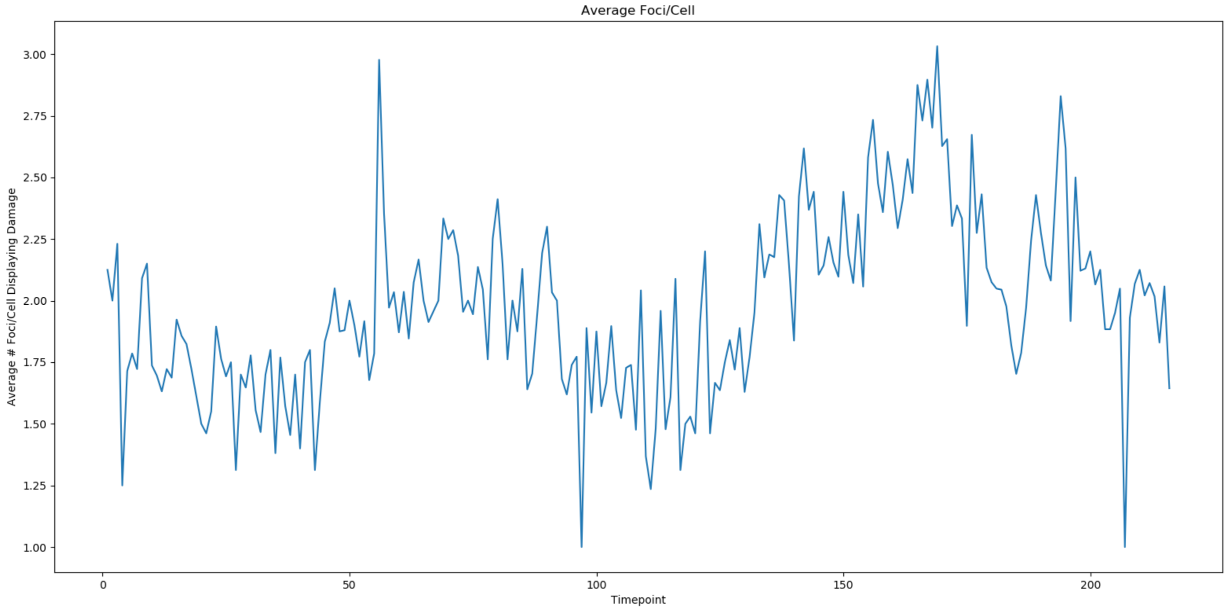


Figure 8: Average number of foci per cell displaying damage

Figure 9 displays the behavior of the total number of cells identified within each timepoint. This appears to be a linear relationship with an equation of linear fit was y = 0.86x + 173.92 and an R-squared (R^2) value of 0.99.

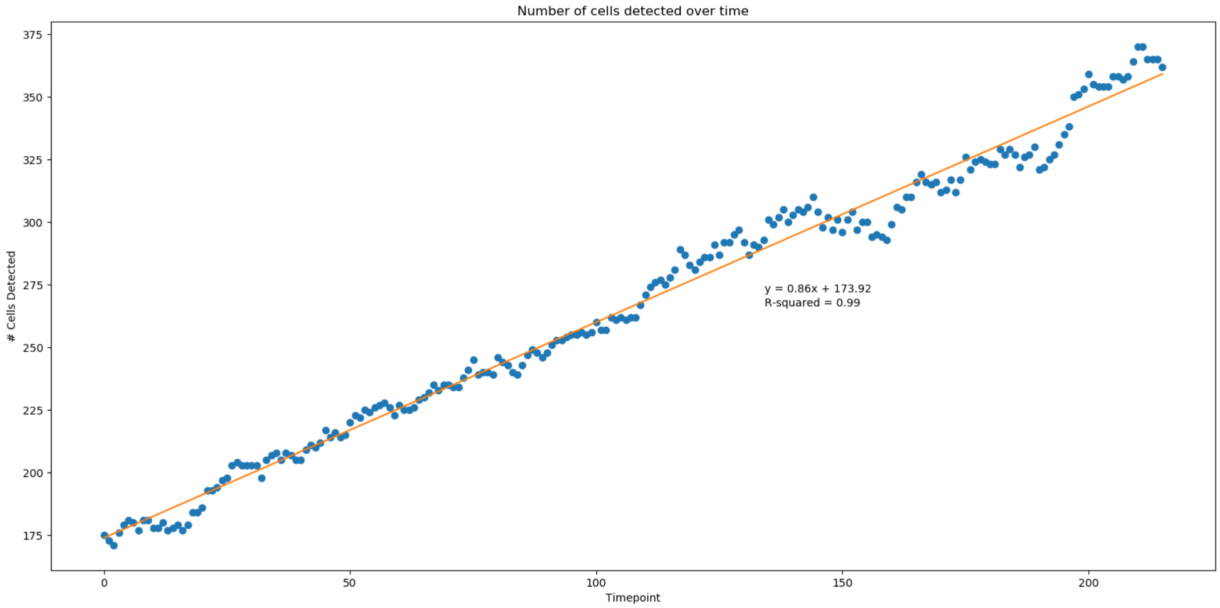


Figure 9: Number of cells detected over time

Next, the percentage of cells containing damage, identified as cells containing one or more foci, was compared to the overall number of cells detected. The results of this query may be seen in Figure 10. The percentage of cells containing damage spanned a 20.0% window, ranging between 0.0% damaged and 20.0% damaged. As addressed previously, the frames containing 0.0% damage may be considered dropouts. Disregarding potential dropouts, most frames spanned a range between 5.0% and 17.5% damaged. As mentioned in the discussion of figure 9, the number of cells identified at each timepoint appears to increase at a steady rate, with slight fluctuations at points of local slope. Based on the information gathered in Figures 8 and 10, it appears that there is a slight increase in the percentage of total cells containing damage over time.

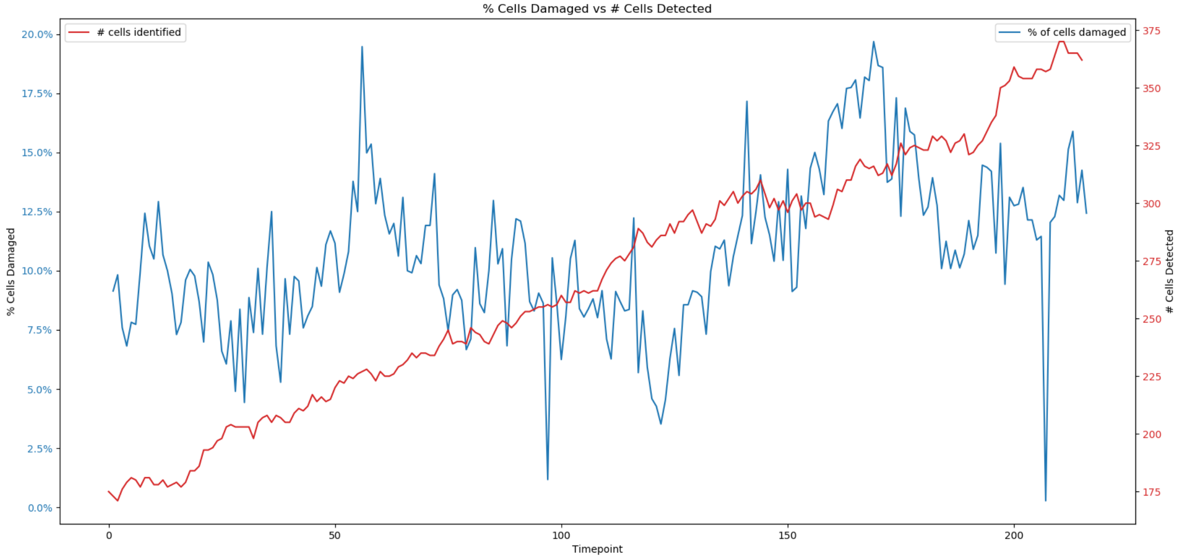


Figure 10: Percent Cells Damaged vs Number of Cells Detected

The team next sought to understand how the number of foci identified changed over time. Figure 11 displays the total number of foci identified over time, with the y-axis indicating the number of total foci and the x-axis corresponding to the timepoints of the dataset. The figure was also colored based on the binning of the total number of damaged cells, containing one or more foci, at that timepoint. This figure shows that the total foci number and the number of damaged cells generally increases over time.

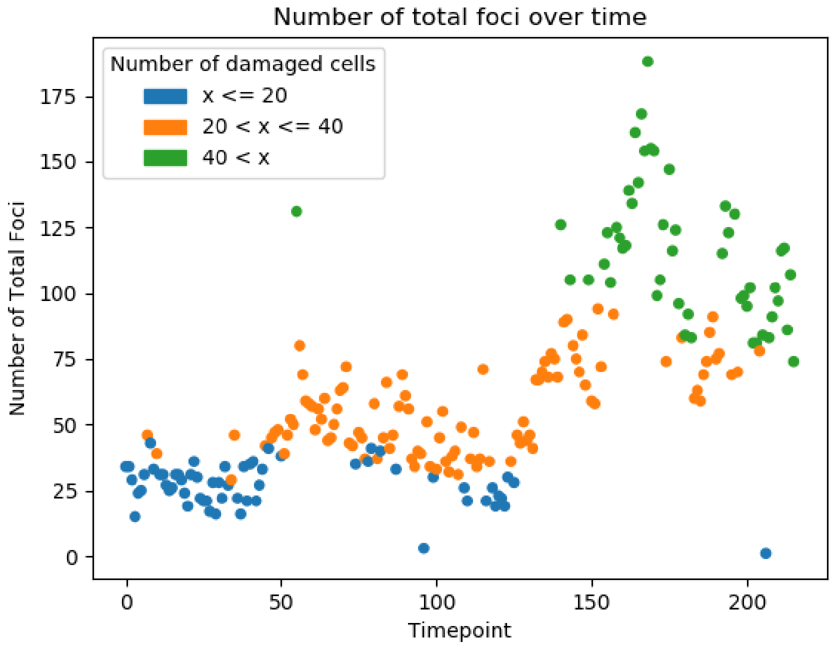


Figure 11: Number of total foci over time

In Figure 12, the team investigated the dynamics of foci number change (delta) between consecutive frames. This was achieved by first normalizing the total number of foci identified at each timepoint. Normalization for each timepoint was achieved by dividing the foci number by the number of damaged cells (cells containing one or more foci). Next, we calculated the difference in normalized foci count between consecutive frames and plotted the results. The normalized delta typically remains within a window between -0.5 and +0.5, with a few outliers spiking near or over +/- 1.0. These delta values are expected to be small because it is unlikely for foci count to vary greatly between consecutive timepoints. The timepoints with large delta values may correspond to timepoints where the CellProfiler pipeline failed to properly identify foci.

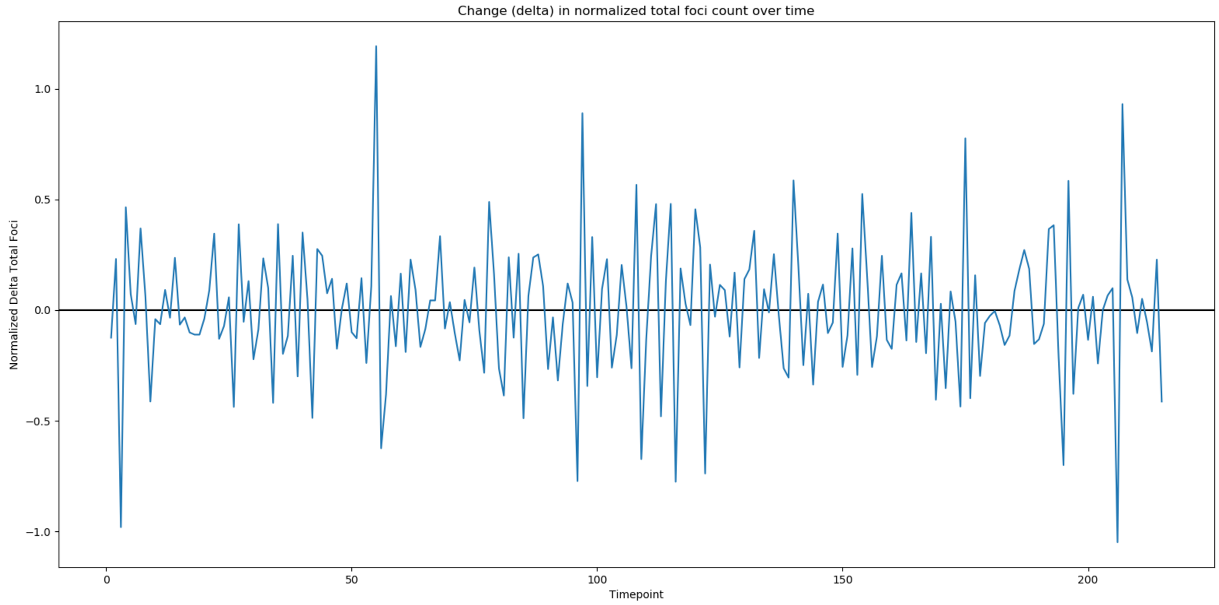


Figure 12: Change (delta) in normalized foci count over time

Next, we identified the percentage of total cells exhibiting damage at each timepoint, binned by the number of damage foci exhibited. An example of an individual timepoint's data distribution may be seen in Figure 13. The x-axis displays the number of foci (greater than or equal to 1) and the y-axis displays the percentage of total cells with "x" foci.

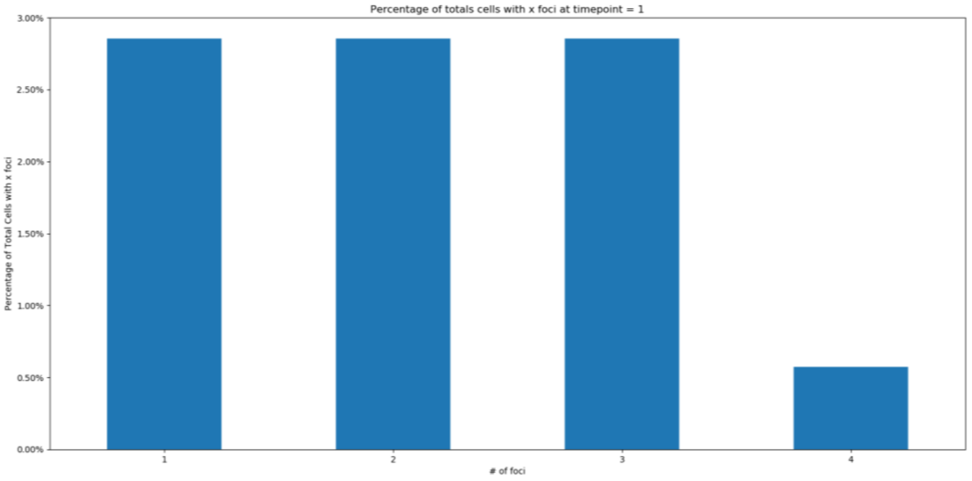


Figure 13: Percentage of total cells with x foci at timepoint 1

Next, we sought to understand the behavior of the percentage of total cells containing damage over time. Figures 14A and 14B display this behavior across the first and middle 20 timepoints in the dataset (images 1-20 and 196-216 in the full movie). The first 20 timepoints approximately correspond to the first hour and forty minutes of the video. The middle 20 timepoints approximately correspond to the middle hour and 40 minutes of the video. Thus, there is about 16 hours and 20 minutes between the data collected at the first and middle 20 timepoints. The x-axis displays the number of foci identified in a cell and the y-axis displays to the percentage of total cells identified with the number of foci at the corresponding x-axis value. Each bar corresponds to a different timepoint of the video, as shown in the figure legend. Figures 13A and 13B both display an overall negative trend, indicating that the frequency of damaged cells with a low number of foci is much higher than those containing many foci. The foci frequency within the middle hour and 40 minutes of the data subset (Fig. 14B), corresponding to the midpoint of the full video, appears to be slightly more variable than that of the foci frequency in the first hour and 40 minutes. Additionally, image 207 in Figure 14B only has one datapoint, indicating that less than 1.0% of the total cells at that timepoint have on foci. This is likely due to CellProfiler being unable to identify foci within the image at that corresponding timepoint, as previously discussed in Figure 8.

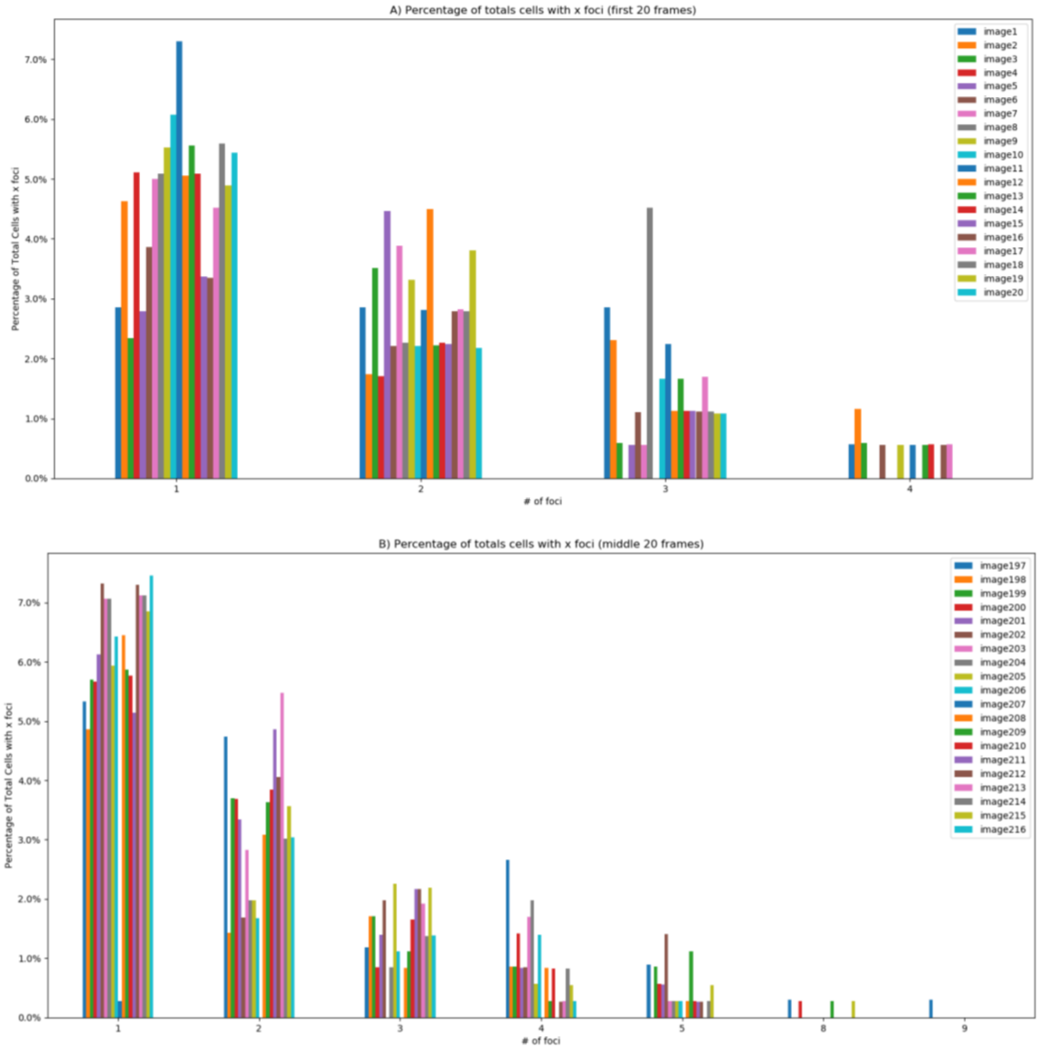


Figure 14: A) Percentage of total cells with x foci (first 20 images)
B) Percentage of total cells with x foci (middle 20 images)

Figure 15 compares the percentage of total cells containing x foci between the first and middle 20 timepoints (hour and 40 minutes) of the video. The axes of this graph are the same as in Figures 13 and 14. The blue bars correspond to the first hour and 40 minutes of the dataset and the orange bars correspond to the middle hour and 40 minutes of the dataset. While both subsets of the dataset display the same negative trend, the cells in the middle 20 timepoints of the dataset appear to have slightly more damage, between 1.0% and 2.0%, per foci number than the cells in the first 20 timepoints of the dataset. This increase in overall damaged cell presence, coupled with the increase in average number of foci per damaged cell, may indicate that cells are either becoming more prone to damage over time or are not healing their damage properly.

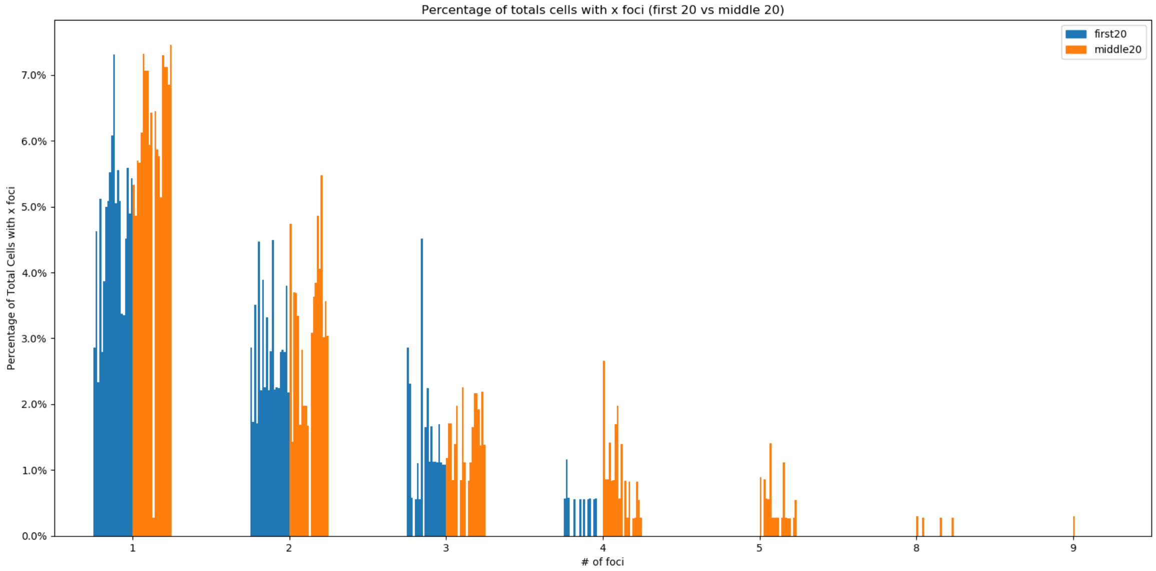


Figure 15: Percentage of total cells with x foci (first 20 images vs middle 20 images)

4. Discussion

CellProfiler was a good starting point for understanding the dynamics of DNA damage repair in cancerous cells. This software is relatively easy to use and customize once you become familiar with it and its nuances. Our utilization of this software allowed us to begin understanding our dataset without writing image analysis algorithms from scratch. Passing our CellProfiler pipeline to the Manning lab will likely cause less confusion than providing raw code due to the lab's familiarity with CellProfiler.

The CellProfiler speckle identification pipeline appears to do a relatively good job of identifying individual nuclei and foci within them once its many parameters have been optimized for the dataset of investigation. The team confirmed this by comparing the CellProfiler counts to the lab's current technique of hand counting nuclei and foci within multiple images. However, an issue encountered when working on adjusting the CellProfiler pipeline for this dataset was the high amount of blur found on the images. Referring to Figure 2B, the green channel has a high amount of green haze that covers the frame. This issue appears most when foci are obscured by the haze. The foci identification functions by separating points of high-intensity green from the background of low intensity, so a green blur across the frame leads to a masking of foci in certain frames. This blur is not consistent across every frame, which meant we had to find the optimal thresholds that would lead to the highest number of foci identified in each frame. These thresholds included limiting the foci to a certain size, and a radius from the foci to use as a reference for the background intensity.

Altering the existing CellProfiler code was more difficult than we had initially predicted, as it became apparent each pipeline was built with the purpose of being run independently. The original plan was to take the cell tracking and speckle identification CellProfiler pipelines, identify the useful parts of both, and merge them together into a single pipeline that produced our desired end result. When working with the individual tasks within each pipeline, we discovered that they are not built in the modular manner that would allow for this technique. While the CellProfiler interface shows each step as wholly distinct from the one before it, in fact there are high amounts of overlap and dependencies shared. Separating and re-combining the code requires a greater knowledge of the application than we have, more akin to a developer of CellProfiler. To combat this limitation, we previously decided the optimal strategy for us would be to use CellProfiler to obtain a large amount of data from the image sets, then write our own programs to interpret the data and track the foci over time. A separate issue that we have realized recently is that each image is not taken from the exact same location. There is inconsistent variation in the microscope camera's positioning from one timepoint to the next. The change in camera position can vary from a few microns to entire cell lengths from the prior image, as shown in Figure 15.

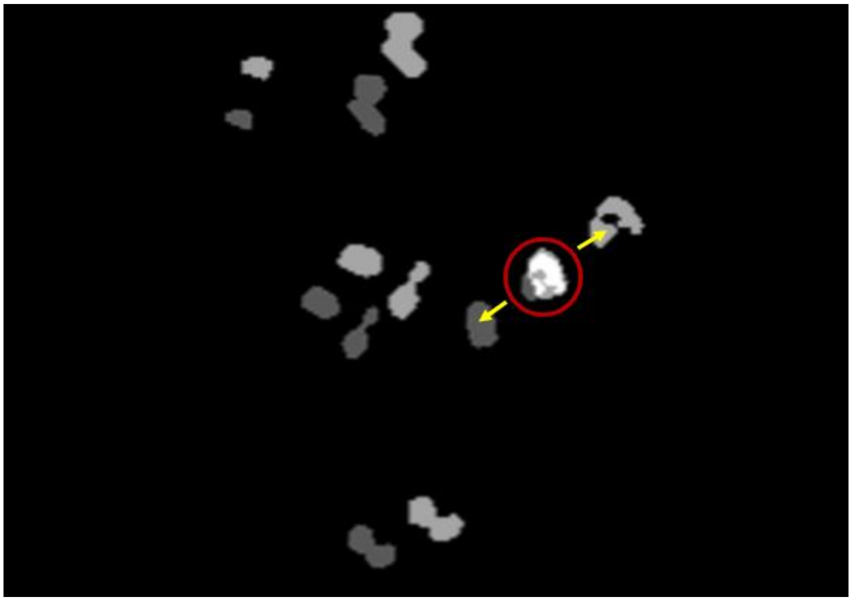


Figure 15. Demonstration of image overlap between frames.

These variations mean that using X-Y coordinates could be an impossible method of tracking foci, and we will need to develop a new technique to accomplish this tracking. A possible small solution we tried was to shift the entire image to overlap correctly. However, the cells are motile so there is no way we found that could identify a standard center. Using moving objects to standardize against other moving objects led to far too much variation. One possible solution to this is to identify foci in reference to the morphology of the cell they are found in. This solution may be feasible, but would require image processing that we are unable to develop on our own. A deeper investigation of CellProfiler mechanics and possible outside programs is necessary to determine how we will combat the inconsistency of the X-Y coordinates.

References

Albanie, S. (2019, June). Euclidean Distance Matrix Trick. Retrieved from Visual Geometry Group, University of Oxford.

Cohn, Martin A, and Alan D D'Andrea. “Chromatin recruitment of DNA repair proteins: lessons from the fanconi anemia and double-strand break repair pathways.” Molecular cell vol. 32,3 (2008): 306-12. doi:10.1016/j.molcel.2008.10.009

Eliceiri, K. W., Berthold, M. R., Goldberg, I. G., Ibáñez, L., Manjunath, B. S., Martone, M. E., Murphy, R. F., Peng, H., Plant, A. L., Roysam, B., Stuurman, N., Swedlow, J. R., Tomancak, P., & Carpenter, A. E. (2012). Biological imaging software tools. *Nature methods*, 9(7), 697–710. https://doi.org/10.1038/nmeth.2084

Khoronenkova, S., Dianov, G. (2015). ATM coordinates SSB repair to prevent DSBs. Proceedings of the National Academy of Sciences Mar 2015, 112 (13) 3997-4002; DOI: 10.1073/pnas.1416031112

Kobiyama, K., Kawashima, A., Jounai, N., Takeshita, F., Ishii, K. J., Ito, T., & Suzuki, K. (2013).
Role of Extrachromosomal Histone H2B on Recognition of DNA Viruses and Cell Damage. Frontiers in genetics, 4, 91. https://doi.org/10.3389/fgene.2013.00091

McQuin C, Goodman A, Chernyshev V, Kamentsky L, Cimini BA, Karhohs KW, Doan M, Ding L, Rafelski SM, Thirstrup D, Wiegraebe W, Singh S, Becker T, Caicedo JC, Carpenter AE (2018). CellProfiler 3.0: Next-generation image processing for biology. PLoS Biol. 16(7):e2005970 / doi. PMID: 29969450

Miwa S, Tome Y, Yano S, Hiroshima Y, Uehara F, Mii S, et al. (2013) Single cell time-lapse imaging of focus formation by the DNA damage-response protein 53BP1 after UVC irradiation of human pancreatic cancer cells. Anticancer Res. 33(4):1373–1377.

Navarro-Serer, B., Childers, E. P., Hermance, N. M., Mercadante, D., & Manning, A. L. (2019). Aurora A inhibition limits centrosome clustering and promotes mitotic catastrophe in cells with supernumerary centrosomes. *Oncotarget*, 10(17), 1649–1659. https://doi.org/10.18632/oncotarget.26714

Titus J, Haugland R, Sharrow S, Segal D
Texas Red, a hydrophilic, red-emitting fluorophore for use with fluorescein in dual parameter flow microfluorometric and fluorescence microscopic studies

J. Immunol. Methods, 50 (2) (1982), pp. 193-204

Panier, S., Boulton, S. Double-strand break repair: 53BP1 comes into focus. Nat Rev Mol Cell Biol 15, 7–18 (2014). https://doi.org/10.1038/nrm3719