Vaanan Murugathas

Mathematical Analysis of Microarrays Being Used In Medicine

Canada's aging population is rapidly increasing to the point at which according to the Canadian government by 2030 they will make up one in four Canadians (Canada, 2014). This demographic is heavily reliant upon our medical system however, it seems that ease of access and accessibility has not been an area of focus in the medical system. A majority of the older demographic have disabilities or medical conditions that makes repeated back and forth trips to the hospital extremely strenuous on the patients and their families. However, if these patients don't go for these visits and do these quotidian tests it may allow a condition to form putting the patients lives at risk. When these fellow citizens of Canada are given a choice between an extremely stressful situation or putting their lives at risk, this shows a clear problem within the system. Covid-19 has highlighted this problem by truly showing how reliant the elderly are upon doing these simple tests. A majority of this demographic have underlying health conditions which require them to go into hospitals and get tested, and we truly saw them suffer due to the lack of access caused by Covid-19.

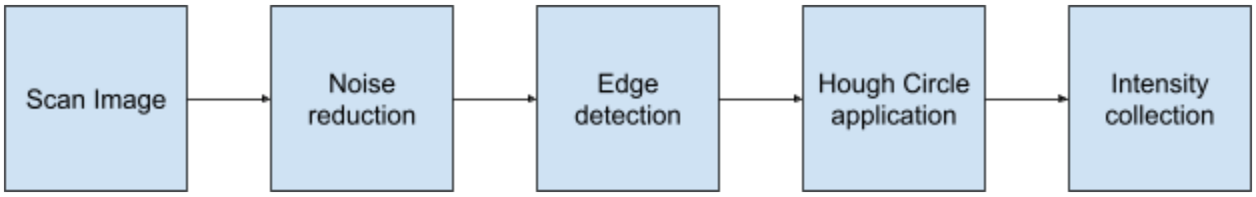
This summer I was spending a lot more time with my grandparents. My grandmother struggles with diabetes and a number of other underlying health conditions which forces her to go to the hospital often. Every other week I would see them scrambling to get into contact with other family members in order to help them get to the hospital. I could clearly see during these days the amount of stress that my grandparents were dealing with and yet all I could do was sit there helplessly. Naturally as any grandchild in this situation would do I began to do my own research into ways that could make the lives of my grandparents just a tiny bit easier. At first, I was doing research into methods that could make transportation less of a hassle, but I quickly realized that this was nothing more than a band-aid solution. The root of the problem lies in the fact that the elderly were forced to constantly do these tests. This led me to looking into microarray testing.

Microarrays have been used recently in the testing of Covid-19 antibodies. The way microarrays work is rather simple. A small blood sample is collected then stored in a vial. Then chemical detectors are put in tiny wells in which very small amounts of blood are placed to react. These detectors will have a visual indicator to show the results of the reaction. These wells are then photographed and by doing analysis of the intensities of light reflected in each well they can be used to give information on not only the presence but also the amount of antibodies present within the blood (PMC, 2020). One of the key advantages of microarray testing is not only that it requires only one small prick of blood in order to do a multitude of tests but also that this blood sample can be very easily transported. By connecting the microarray testing with my grandparents' problem, I recognized that microarray testing is the future of medicine. This report investigates the math involved in the processing of data collected from microarray testing as well as how a statistical analysis can be done in order to test blood for common tests done to diabetes patients.

Part A: Image processing

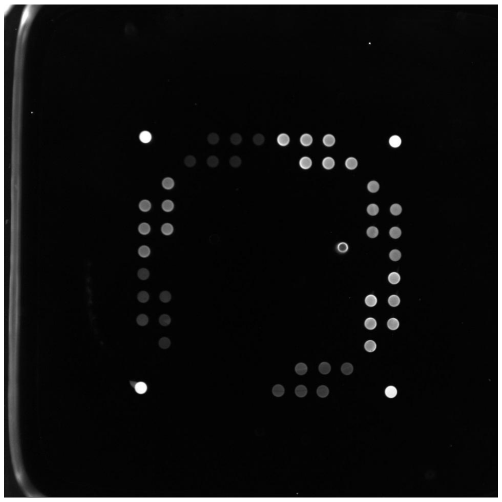
Although this IA would've been much simpler if I simply received the final intensities in microarray testing I wanted to investigate this concept from a much more holistic standpoint which meant that I would write my own image processing code in python in order to collect data from sample microarray images that can be found in the appendix. A simple outline of the steps involved in the processing of microarrays can be seen in **figure 1.**

Figure 1: The process which is used in order to analyze images from microarrays



A simplified version of the code consists of the following steps. First, the code has to detect where the circles seen in **figure 2** are located in the screen and then proceed to collect the intensity that is at the centre of each circle. However, in order to do this the code uses a couple of fascinating mathematical concepts.

Figure 2: Sample image collected from microarray that displays indicators showing if the patient has covid-19 antibodies



The first layer of math involved is called Canny Edge Detection. What it does is a multiple step process in order to calculate a threshold of intensity difference in order to create an outline of all objects (Towardsdatascience, 2019). The first step involved in Canny Edge Detection is noise reduction. Noise is referring to light that appears in the background of the image due to slight flaws in the process of data collection. In order to minimize the impact of said noise, a Gaussian filter is applied. Next paragraphs explain how this filter is implemented.

A Gaussian filter is applied to each pixel by taking into account the pixels around said pixel. The filtering operation is performed using a kernel. The kernel consists of pixels in the area surrounding the pixel being analyzed that are being used to affect the value of the pixel. To apply this filter the pixels in the kernel are being used to model a Gaussian function:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where $-5 \leq x \leq 5$ and $-5 \leq y \leq 5$ and $\boldsymbol{\sigma}$ represents the standard deviation which will be $\frac{1}{3}$.

The reason why x and y are limited to that range is because the Gaussian filter is only applied within the size of the kernel. Within my code the kernel size is 5x5 because after trial and error this kernel size was able to adequately reduce noise without distorting the intensity values being collected by a significant amount. Similarly, one third was chosen for the standard deviation because there is a consensus among those who use Gaussian blurs that a standard deviation of one third leaves the best segmentation results (Sciencedirect.com, 2020). This math is automatically done within the code however a sample calculation and explanation will be provided.

Figure 3: Example of a 3x3 gaussian kernel that can be calculated using double integrals and be used for convolution of images

1	2	1
2	4	2
1	2	1

However, these values that appear within the kernel are not randomly allocated. The purpose of these kernels is to create a weighted average that follows the patterns of a 2D Gaussian distribution. Hence, the name Gaussian kernel is given. Each of the values of the kernel are determined by using a double integral. The reason why a double integral is being used is that the value in each of the boxes within the kernel is determined by the area it takes up under the normal distribution. Since this is a 2D normal distribution, in order to determine the values of each box within the kernel a double integration is required (Wolfram, 2021). An example of this type of calculation is:

$$f(x,y) = \int_{y-0.5}^{y+0.5}\int_{x-0.5}^{x+0.5} \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\, dxdy$$

Where (x,y) are the coordinates of the pixel in comparison to the central pixel and **σ** represents the standard deviation which is still one third.

In order to integrate the normal distribution the simplest method that I found after doing some research is using the error function in order to be able to compute the value of $f(x,y)$ (Wolfram, 2021). Note that erf denotes error function. Thus the new equation for $f(x,y)$ is:

$$f(x,y) = \frac{1}{4}\left(erf\left(\frac{x+0.5}{\sigma\sqrt{2}}\right) - erf\left(\frac{x-0.5}{\sigma\sqrt{2}}\right)\right)\left(erf\left(\frac{y+0.5}{\sigma\sqrt{2}}\right) - erf\left(\frac{y-0.5}{\sigma\sqrt{2}}\right)\right)$$

Each of these error functions need to be calculated; however, to avoid redundant calculations I will show the work for one and then simply plug in the rest of the values. If we were to calculate the values for (x,y)=(1,1) the following will be done:

$$erf\left(\frac{1+0.5}{\frac{1}{3}\sqrt{2}}\right) = \frac{2}{\sqrt{\pi}}\int_0^{3.1819} e^{-t^2}dt$$ (Note: 3.1819 is the simplified value of $\left(\frac{1+0.5}{\sigma\sqrt{2}}\right)$)

$$erf\left(\frac{1-0.5}{\frac{1}{3}\sqrt{2}}\right) = \frac{2}{\sqrt{\pi}}\int_0^{1.0607} e^{-t^2}dt$$ (Note: 1.0607 is the simplified value of $\left(\frac{1-0.5}{\sigma\sqrt{2}}\right)$)

With these equations we can plug them into the calculator in order to receive the following values:

$$erf\left(\frac{1+0.5}{\frac{1}{3}\sqrt{2}}\right) = 0.9999$$

$$erf\left(\frac{1-0.5}{\frac{1}{3}\sqrt{2}}\right) = 0.8663$$

Now that we have the values of the error function we can plug them back into the equation for $f(x,y)$.

$$f(x,y) = \frac{1}{4}(0.9999 - 0.8663)(0.9999 - 0.8663)$$

$$f(x,y) = 0.004462$$

$$f(x,y) \approx 0.004$$

Therefore when (x,y) = (1,1) the value of the kernel is 0.004. Using the kernels that are calculated in the program we convolute the image in order to create the blur effect. This is explained next.

The way that convolution works is rather simple. Let p(x,y) represent the unblurred image, let f(x,y) represent the kernel and let b(x,y) represent the blurred image. The relationship that convolution creates between the three is:

$b(x,y) = f(x,y) * p(x,y)$ Note: * is a symbol used to denote convolution

What the convolution looks like in a practical sense is like the following. The grid of ones can represent the intensities found in a sample image. If the pixel at the centre was the one we were analyzing, and we wished to apply a 3x3 Gaussian convolution kernel, this would result in the kernel at **figure 3** being applied to the image at **figure 4** in order to alter the image. To get the new value of the central pixel, all 9 of the pixels would be multiplied by the according box within the kernel. After this the new values of the pixels would be summed then divided by the original sum of the kernel. This would give the new intensity of the central pixel. In this case since all the pixels are equal to 1 the pixels after being multiplied by the value of the kernels would sum to 16. The sum of the original kernel is also 16 and after division the blur would leave the value of the central pixel completely unchanged. This convolution is then repeated for every single pixel within the image to create the desired reduction of noise (Aryamansharda, 2021).

Figure 4: Sample 3x3 array of pixels in an image all pixel intensities are the same and thus won't be affected by any blurring tools.

1	1	1
1	1	1
1	1	1

After this blur is applied and the noise has been reduced the first stage of edge detection can commence with the help of sobel kernels. Sobel kernels are used to emphasize the edges and make edges extremely distinct. In order to do this a kernel of both x and y are created in order to approximate the derivatives of the change in horizontal and vertical directions.

Figure 5: Sobel kernels in the x-direction and y-direction, when both are taken into account it can result in a holistic convolution with a magnitude and direction shaped by both kernels

X – Direction Kernel

-1	0	1
-2	0	2
-1	0	1

Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

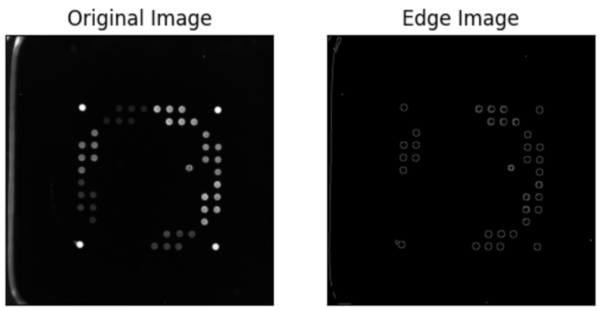
The edge detection is done by convolving the kernels with the image at each pixel. Considering the fact that a sample calculation of convolving kernels with an image has been done above this sample calculation would be somewhat redundant. All that needs to be known that is new content is how convolution works when applying two kernels. As seen in **figure 5** the sobel kernels are present in both axes. This causes the resulting convolution to be a vector. Let M represent the magnitude of the new pixel and θ represent the angle in which the magnitude is facing

$$|M| \ = \ \sqrt{M_x^2 + M_y^2} \qquad \theta \ = \ tan^{-1}(\frac{M_y}{M_x})$$

Where M_x represents the intensity of the central pixel after a convolution is completed with the x-direction kernel and M_y represents the central pixel after a convolution is completed with the y-direction kernel. The final magnitude M is the resulting intensity in the pixel and the reason theta is collected is due to the fact that it's useful to know the angle in which the edge is facing in the next step (Automaticaddison, 2019).

Finally the last stage of Canny Edge Detection is non-maximum suppression. The purpose of this step is to completely reduce the edge to 1 pixel in order to maximize the accuracy of the location of the edge detection. This process is done by taking into account the magnitude and direction collected thanks to the sobel kernels. The intensity of each non-zero intensity pixel is compared to it's two neighboring pixels along the direction of the gradient. If a pixel has a lower intensity in comparison to the neighboring pixel it's intensity will be set to zero thus creating the fine line. This creates the final touches to the filtration seen in **figure 6.**

Figure 6: The impact of canny edge detection on an image on the left is the source and on the right it displays detected thin edges of the circles

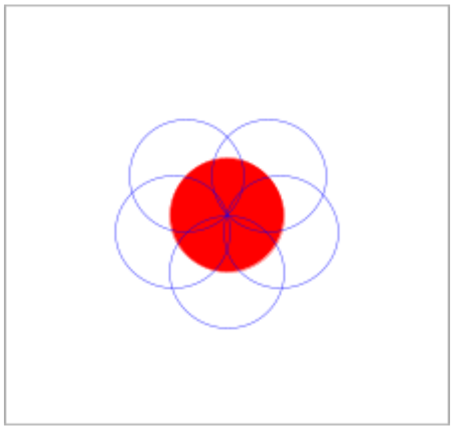


The second step to determine the location and intensities of each circle is done using the Hough Circle Transformations. This operation estimates the circle that best fits the detected edges in the previous step. When graphing a circle the equation used to describe the circle is:

$$r^2 = (x - a)^2 + (y - b)^2$$

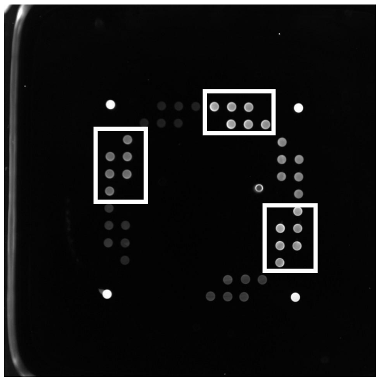
Where r is the radius of the circle, (a,b) are the coordinates of the centre of the circle and (x,y) are stand-ins for any point on the outline of the circle. In order to identify the locations of circles seen in the edge image of **figure 6** the Hough Circle algorithm is used (GeeksforGeeks, 2019). It is important to firstly note that r is already known. As can be seen in **figure 6** as well as the sample data found in the appendix, these circles are all of constant size. Thus, rather than it being necessary to find r, it has been set as a constant. The locations for (x,y) have also already been discovered. These locations are the edges of the circles that the Canny Edge Detector has already exposed. With r and (x,y)'s potential locations being revealed, the process to determine the circle's centre is rather simple. Taking the base circle outline, a random point on the edge of the circle is selected and a circle with radius r is drawn with that point at the centre. As seen in **figure 7**, although this is at a reduced scale the drawing of the circle around the edge is repeated multiple times. Now, in order to determine the location of the centre of the base circle or the location of (a,b) all that needs to be done is to see the location at which all the circles intersect. The reason why this works is that all the circles have the same radius, meaning putting the centre of the circle at the edge of the base circle allows the new circle to intersect with the centre of the base circle. By taking the intersecting point of all these circles that have been drawn the point (a,b) can be determined (Thales Sehn Körting, 2020).

Figure 7: What the Hough circle algorithm does in order to find the center of a circle



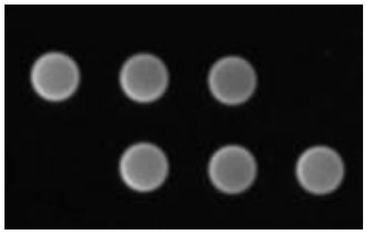
Finally comes the easiest stage which is the data collection section. Using the Hough circle transformation we are able to determine the center of the circle. However, due to my inexperience in coding I was unable to have the code organize the data collected from each image in the desired fashion. The reason why it's important for the data to be organized is that each of those rectangles seen in **figure 8.1** represent a different antibody testing. If they are mixed the data will no longer be relevant or useful.

Figure 8.1: Visual demonstration of desired organization of data collection.



This forced me to adapt in order to be able to collect data in an organized fashion. I had to manually segment each of the photos so that the only groups of circles present in each image were those that had the same antibody marker. This created images like that in **Figure 8.2** as the new source of data collection. I recognize that in the future when I develop this further in an attempt to help my grandmother having the code do this automatically would be necessary. I repeated this process for each image I had access to. This created images like the following.

Figure 8.2: Manually segmented image of only one antibody location within the microarray



Originally in my code I was simply taking the centre pixel of the circle and getting an intensity from that point. However, I realized that sometimes the centre pixel is not providing an accurate representation of that circle's intensity for a variety of reasons. This could be because the pixel happens to be a dead pixel (No intensity due to issues with the camera) or if the pixel has been affected by noise that bypasses the noise suppression. In order to accommodate for this I decided to take the average of the pixels that form a plus around the central pixel. To get these values I recognized that the five pixels would be (x,y), (x+1,y), (x-1,y), (x,y+1) and (x,y-1) where (x,y) is the central pixel. After collecting the intensities at each pixel, it was time to simply take the mean by adding the five intensities together and dividing by five. Taking the mean from pixels did not have a large impact on pixels that gave an accurate result; however, it did significantly improve the accuracy of the formerly inaccurate pixels. I think that having some error in the collected data is inevitable even at a professional level. Thus, it makes it even more critical that the algorithms used are able to ensure that this does not have an impact on the analysis. Although averaging may serve as a temporary solution, I think it's important that in the future I look into methods that are able to collect the intensity of pixels in a circle in a much more holistic sense.

Part B: Statistical analysis

I was able to get access to discarded sample data from a company called SQI Diagnostics. Due to the current circumstances of lack of tests they were running Covid-19 antibody tests. SQI uses microarrays in order to determine the presence of the three antibodies that our bodies create in response to Covid-19. I was able to ask my mother who currently works there for access to data that is not critical to the research but would still be able to show certain trends. Thanks to that I have a variety of images from Covid-19 antibody testing within microarrays. The next stages of the math are going to be based on a hypothetical scenario in which we are going to pretend the markers that were used in tests actually represent plasma glucose tests, A1C tests and glucose tolerance tests (NIH, 2019). These are tests that my grandma often has to complete when going for a check-up at the doctor's office.

In order to complete a statistical analysis it is first necessary to understand what type of distribution we are working with. For each of the three tests I have access to 300 data points. If the data points were simply determining whether the patient has a certain amount of glucose the data could be classified as binomial however, the data is indicative of specific levels of glucose and among different patients (trials) there is no constant probability for having or not having a certain level of glucose. It also does not have the qualifications to be classified as a Poisson distribution because there is no interval or average rate of occurrence. Finally, it is not normally distributed because the data is not perfectly symmetrical. However, if we were to create samples and have the sample size be greater than thirty using the central limit theorem the distribution can be modelled as tending towards a normal distribution and can also be used in hypothesis testing.

A couple of assumptions need to be made in order to proceed with the statistical analysis. Firstly, due to the fact that there are no baselines for what the intensity of the average human being is, it will be assumed that the average of every single data point combined is the population mean. As for the standard deviation rather than getting further inaccuracies if a t-test is used the standard deviation won't have to be inaccurate because the unbiased estimator can be used. Another assumption being made is that the data collected before running the statistics is all from completely healthy normal patients. Using sample sizes of 30 an example of what the data distribution could be modelled as is seen below.

$$\overline{X} \sim N(83.8, \frac{(24.8)^2}{30})$$

With this distribution set a number of things can be done. Firstly it's important to note that these thirty data points are present in each image. This means that every image can be analyzed to determine whether or not the data present is found within the data set. Thus the objective of using statistics in this scenario would be when receiving a new microarray image to determine whether or not that patient's data is classified as healthy. In order to do this firstly we will take a sample microarray image with the exact same tests however, the state of the patient is unknown to me. I will then proceed with an analysis using a t-test.

Figure 9.1: The sample data of unknown origins that has been manually segmented in order to separate the three "diabetes tests"

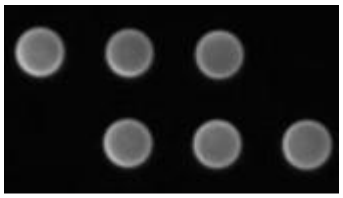
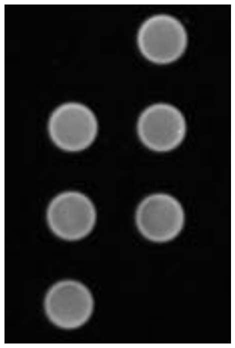
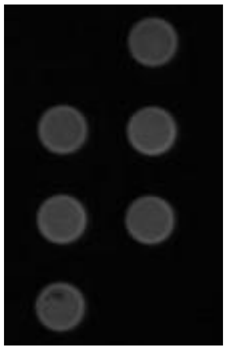
Q1:  Q2:  Q3: 

Figure 9.2: Intensities of the data with unknown origins collected via the code:

Sample number	Intensities of Q1	Intensity of Q2	Intensity of Q3
1	104	129	63
2	103	123	64
3	104	123	62
4	101	125	63
5	103	129	63
6	97	124	63
7	96	122	64
8	95	123	60
9	93	126	61
10	97	125	60
11	104	116	59
12	101	119	59
13	105	113	59
14	100	112	60
15	104	112	59
16	115	129	65
17	114	128	66
18	112	131	64

19	112	130	65
20	118	129	65
21	105	118	64
22	106	119	64
23	102	119	63
24	105	117	62
25	105	120	63
26	91	121	59
27	92	118	60
28	93	121	58
29	93	121	61
30	91	121	59

With this data a t-test can be done on each sample. Firstly, with Q1, the null hypothesis is H_0: 83.836 and the alternate hypothesis is H_1: > 83.836. These hypotheses indicate whether the data from the new image would be classified as a healthy patient (H_0) or a patient that has more blood sugar than a regular patient (H_1). As stated in previous paragraphs, the mean of the population was assumed to be the average of all data points I had been able to collect. By putting all the data into L_1 in the TI-84 calculator and doing 1-stat analysis on the list I received 83.836 as the population mean. I then placed the data under Q1 in **figure 9.2** into L_2. Using a t-test in the TI-84 with the two inputs, the p-value I received was 2.2910-4. In advance, I decided that p-values less than 0.05 were significant. The justification for this was a consensus that statistical significance can be determined if the p-value is less than 5% (Springeropen, 2020). Since 2.2910-4 < 0.05 we can reject the null hypothesis meaning that the patient has high blood sugar and should continue with their diabetic treatment. With Q2, the null hypothesis is H_0: 98.084 and the alternate hypothesis is H_1: > 98.084. The justification for the population mean is the same as Q1 except the data being averaged comes from Q2. I replaced the data in L_2 with the data under Q2 in **figure 9.2**. With the TI-84 I was able to determine that the p-value is now 2.6710-21. Clearly 2.6710-21 < 0.05 meaning that we can once again reject the null hypothesis. In this case this would indicate that the patient has tested positive for having a higher than average blood sugar for the past three months. Finally, in Q3, the null hypothesis is H_0: 41.753 and the alternate hypothesis is H_1: > 41.753. Once again the justification for the population mean is the same and it has been calculated with all the data from Q3. I plugged the values under Q3 in **figure 9.2** into L_2 with this data using the TI-84 the p-value that was calculated was 2.3210-29. Due to the fact that 2.2910-4<0.05 this allows us to reject the null hypothesis which means that the body of the patient handles glucose less effectively which would once again encourage further treatment options.

Conclusion

What started as a simple method to increase my understanding of a technique that could help my grandparents quickly became a topic that fascinated me and caused me to delve deeper and deeper into the algorithms and math behind these concepts. This process was riddled with difficulties. I faced simple issues like requiring a lot of manual labour in order to collect data and manipulate said data. However, I also faced bigger challenges like my curiosity forcing me to learn how to code an image processing software and said software requiring me to learn completely new and complex mathematical concepts. I particularly had a tough time learning about the Gaussian blur. It forced me to be able to understand and analyze a 2D Gaussian function as well as learn about integration and convolution. However, this math truly enlightened me about many of the things I have taken for granted in terms of image processing. Whether it's applying a blur in photoshop or something as mundane as applying a filter to my Snapchat photos I can now clearly see the role that kernels, convolution and integrals play in these simple activities. Not only do I now have a much better understanding of the way in which microarray testing actually works, but I also see the impacts of this math that I've learned in my daily life.

It is important to recognize that there are a lot of limitations that are created within my investigation of the subject. An obvious area that would have room for improvement is collecting more data points from an individual patient. This would mean that when it comes to modelling the sample with central limit theorem since n would be larger it would be closer to a normal distribution. This results in less inaccuracies within the hypothesis testing and will reduce the possibility of type I and type II errors occurring. Another limitation that I briefly touched upon was in the data collection stage. There should be no manual labour to collect data as this obviously will make people view the product as less legitimate. This should have been optimized however, I was teaching myself how to code, and so I have limitations in terms of the skill necessary to apply the automation. Within the data collection it would be better if the intensity collected is not simply a reflection of the central five pixels in a circle but instead to provide a holistic intensity within the circle. A method that I thought of for this was to apply some of the mathematical knowledge I gained during this process. The collected intensity could be based on a weighted average where the weight of the points will be modelled via a 2D Gaussian distribution. This would be a similar concept to the kernels that were used in the Gaussian blur and would make the intensity collected much more representative of the circle. Finally, the last major limitation in this investigation relates to the Hough Circle transformation. While I was able to conceptually grasp what the algorithm was doing in the background, when I attempted to do research into how the algorithm was actually working I quickly recognized that I was out of my league. A majority of the words and symbols that were being casually tossed around were completely unfamiliar to me and no matter how many YouTube videos I watched on the concept I simply didn't have an adequate base of knowledge to be able to comprehend it. The math behind the generation of circles and the accumulator matrix continues to intrigue me, so it might be a concept that I revisit after improving my math and coding skills.

Although there were many challenges and limitations to the investigation, the primary goal was achieved. With this thorough understanding of both how the testing itself actually works and how the data collected from this testing can be analyzed. With this information it will genuinely help in the development of my longer term project. The goal of this investigation has always been to have an impact on my grandparent's life in order to facilitate one aspect of their stressful lives. This investigation is the first step of many involved in helping not just my grandparents but also a rapidly growing aging population.

Bibliography

Canada. (2020). *Seniors Action Report | Employment and Social Development Canada* Retrieved May 17, 2021, from https://www.canada.ca/en/employment-social-development/programs/seniors-action-report.html

PMC. (2020). *A Modular Microarray Imaging System for Highly Specific COVID-19 Antibody Testing* Retrieved May 17, 2021 from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7263497/

Towardsdatascience.com. (2019). *Canny Edge Detection Step by Step in Python — Computer Vision*

Retrieve May 17, 2021, from

https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

Sciencedirect.com. (2020). *Gaussian Blur|Machine learning assisted segmentation of scanning electron microscopy images of organic-rich shales with feature extraction and feature ranking*

Retrieved May 17, 2021, from

https://www.sciencedirect.com/topics/engineering/gaussian-blur

Wolfram.com. (2021). *Gaussian Integral | Mathworld*

Retrieve May 17, 2021, from

https://mathworld.wolfram.com/GaussianIntegral.html

Aryamansharda.medium.com. (2021). *Image filters: gaussian blur*

Retrieved May 17, 2021, from

https://aryamansharda.medium.com/image-filters-gaussian-blur-eb36db6781b1

Automaticaddison.com. (2019). *How the sobel operator works*

Retrieved May 17, 2021, from

https://automaticaddison.com/how-the-sobel-operator-works/

GeeksforGeeks.com. (2019). *Circle detection using opencv | Python*

Retrieved May 17, 2021, from

https://www.geeksforgeeks.org/circle-detection-using-opencv-python/

Thales Sehn Körting. (2020). *How circle hough transform works. Youtube*

Retrieved May 17, 2021, from

https://www.youtube.com/watch?v=Ltqt24SQQoI

National institute of aging. (2019). *Diabetes in older people*

Retrieved May 17, 2021, from https://www.nia.nih.gov/health/diabetes-older-people

SpringerOpen. (2020). *Statistical significance: p value, 0.05 threshold, and applications to radiomics – reasons for a conservative approach*

Retrieved May 17, 2021 from https://eurradiolexp.springeropen.com/articles/10.1186/s41747-020-0145-y

Appendix

Figure 9: Python code that applies the math of canny edge detection and houghcircles in order to collect the intensity of wells within microarrays that have been manually segmented. (Note: The 5th line of code where 'F04_2_q3.jpg' is located will be altered when collecting data. The image that the data is being collected from must be manually changed in the code will collecting data.

```
import cv2
import numpy as np

# Read image.
img = cv2.imread('F04_2_q3.jpg', cv2.IMREAD_COLOR)

# Convert to grayscale.
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Blur using 3 * 3 kernel.
gray_blurred = cv2.blur(gray, (3, 3))

# Apply Hough transform on the blurred image.
detected_circles = cv2.HoughCircles(gray_blurred,
                    cv2.HOUGH_GRADIENT, 1, 20, param1 = 50,
                param2 = 35, minRadius = 1, maxRadius = 40)

# Draw circles that are detected.
if detected_circles is not None:

    # Convert the circle parameters a, b and r to integers.
    detected_circles = np.uint16(np.around(detected_circles))

    for pt in detected_circles[0, :]:
        a, b, r = pt[0], pt[1], pt[2]

        # Draw the circumference of the circle.
        cv2.circle(img, (a, b), r, (0, 255, 0), 2)

        # Draw a small circle (of radius 1) to show the center.
        cv2.circle(img, (a, b), 1, (0, 0, 255), 3)
        cv2.imshow("Detected Circle", img)

        intensity1 = int(gray[b,a])
        intensity2 = int(gray[b + 1,a])
        intensity3 = int(gray[b - 1,a])
        intensity4 = int(gray[b,a + 1])
        intensity5 = int(gray[b,a - 1])
        avgintensity =
str((intensity1+intensity2+intensity3+intensity4+intensity5)/5)

        #Appending collected data to a separate .txt file
        a = open('intensitylist.txt', 'a')
        a.write(avgintensity)
        a.write('\n')
    a.close()
    cv2.waitKey(0)
```

Figure 10: A small selection of the total amount of microarray images I analyzed in order to get the population mean

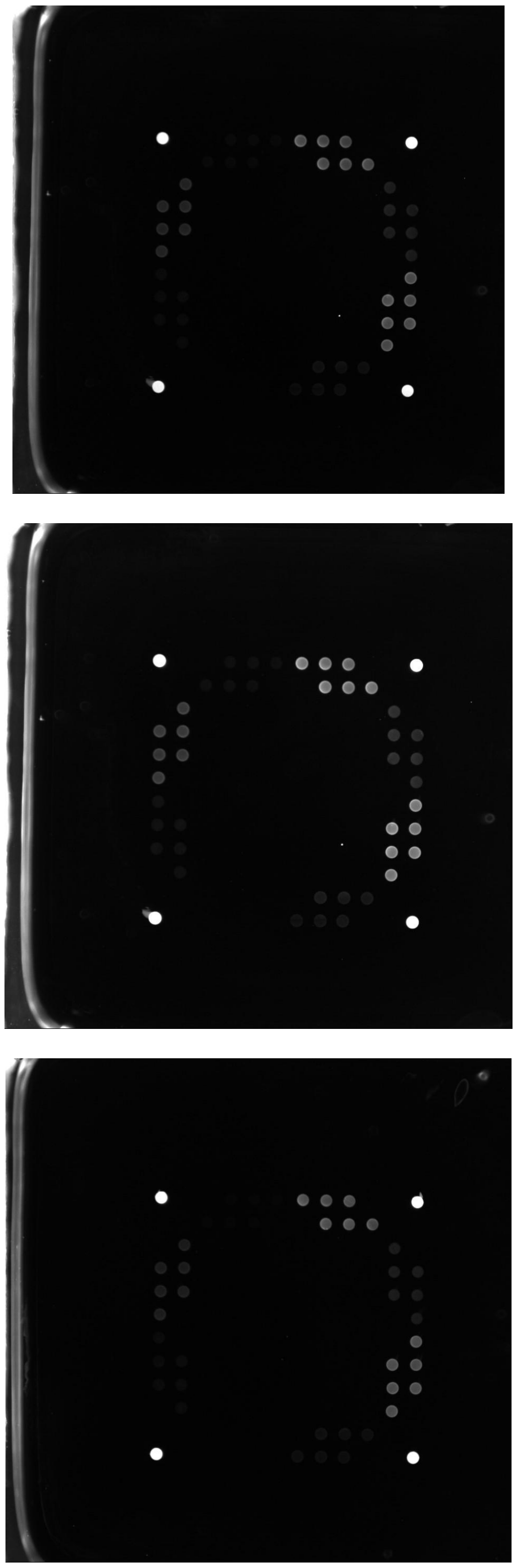


Figure 11: A small segment of the data that was collected in order to get the population mean

Sample number	Intensities of Q1	Intensity of Q2	Intensity of Q3
1	74.4	117.6	33.2
2	59.2	111	31.4
3	64.8	115.6	31.6
4	70.6	128.4	29.4
5	65.6	119.2	31.6
6	71	121.2	33
7	125.2	70.6	55.6
8	106.6	66.8	54
9	107.6	69.4	54.6
10	119.2	73.8	54.4
11	120.4	76.8	48.4
12	98	74.2	53.2
13	77.4	73.6	48.4
14	71.2	74.2	41.6
15	77.8	70.4	46
16	83	78.2	42.8
17	78.8	73.8	41.6
18	66.4	72	43
19	129.6	124.2	28.8
20	130.6	124.4	25.2
21	118.2	117.6	28
22	135.2	121.2	24.8
23	132.6	128.8	25.2
24	113	120.2	25.8
25	62.4	122.4	33.8
26	58.6	133.6	32
27	62	134.8	34.6
28	56.6	115.4	31.4
29	50.8	126.6	31.6
30	49	127	30
31	103.4	81.4	55.6

32	85.6	73.6	53.8
33	102.4	75.2	49.4
34	81.8	80.8	58.6
35	95	74.8	54
36	102.8	68.8	53.4
37	60	78.4	54.2
38	66.6	71.6	54.4
39	67.2	75.8	60.8
40	56	72.2	51
41	52.2	69	53
42	62	72	49.2
43	96.8	125.2	36.6
44	94.2	116.2	31
45	103.6	114.4	33
46	85	131.4	30
47	108.8	113.8	33.2
48	110.8	123.2	31.4
49	54.8	119.2	37.4
50	55.4	113.6	33.6
51	55.6	120.8	33.2
52	53.8	123.4	36
53	52	127.2	33.8
54	49.2	119.2	34.8
55	91	71.8	57.2
56	92.8	70.2	59.2
57	94.6	73.4	57.4
58	90.6	74.8	55.2
59	83.4	74.2	60.8
60	87	74	55.4