Contents

20/10/2020

Page 1-4: **Contents**

Page 5: **Introduction Notes**

Page 5-6: **Optical Correlator**

- What the correlator is
- Why it is useful
- Discovered Correlation types and discussion

Page 6-7: **Python**

- Why use Python
- Timing in Python
- Python Libraries discussion

Page 7-8: **Spatial Light Modulator**

- Research into what they are
- Discovered applications

Page 8-11: **Masks for SLMs**

- Types of mask effect
- Modelling the discoveries

Page 12-14: **4F Correlator**

- Typical setup
- Researched uses of correlators
- First full system models

Page 14: **Charged-Couple Device (CCD and CMOS)**

- Research into what they are
- Why they are necessary

Page 15: **Digital Micromirror Device (DMD)**

- Research into what they are
- Why they are necessary

Imhotep Booth (Tepi) – Lab Book; Design an Optical Computer

Page 15-16: **Optical Computer Full Example Setup**

- Setup and aims
- Weeks (1-5) & (5-10) Report

Page 17-21: **Effect of Defects in the System**

- Types of Defect
- How to measure effects
    - MSE advantages and its flaws
    - SSIM advantages and its flaws
- Defect in Original Image
- Defect in Fourier Spectrum

Page 22-25: **Modulation Transfer Function**

- What it is
- Why this is useful to the system research
- Modelling the MTF

Page 26-30: **Future System Plans**

- Why more details are necessary
- Commercially available SLMs
- Commercially available DMDs
- Commercially available CCDs
- Plan for optimising the design of the system

Page 30-38: **Christmas Holiday Work**

- What are Holograms
- Why this is useful
- Diffractio Library
- Holopy Library

Page 38-42: **Look-up Table**

- How to create it
- Improvements to simulation
- What the results mean

Imhotep Booth (Tepi) – Lab Book; Design an Optical Computer

Page 43-48: **Fringe Patterns**

- Why they matter
- What they are
- Re-simulating the diffraction patterns
- Comparison of various patterns

Page 49-54: **Video Tracking**

- How the video tracker works
- What the results show for the first tests
- Improvements to make to the system
- Results of improvements taken to the system
- Discussion of areas yet to improve

Page 54-61: **The Hologram Analysis System (THAS)**

- What THAS is designed to do
- Resulting data produced
- Improvements to make to the system

Page 61-69: **The Hologram Analysis System 2 (THAS2)**

- Improved system results
- Discussion of the improved system

Page 70-72: **Computational Analysis of THAS2**

- Adjustments to THAS2
- Resulting CPU and Memory data

Page 73-87: **The Hologram Analysis System 3 (THAS3)**

- What needs to be improved and how
- Aims of THAS3
    - Improved accuracy of detections
    - Larger number of detections
- Results of adaptions to THAS2
- Discussion of THAS3 capabilities

Page 88-92: **Optical Computer Implementing THAS3**

- Optical Computer setup
- Design details
    - Light source
    - DMDs
    - Filters and SLMs
    - CCDs
    - Analysis System (running THAS3)
- Anticipated results of the Computer
- Discussion of the Optical Computer's capabilities

Page 92: **Conclusions**

Page 92: **References**

Page 93-148: **Appendices**

Introduction Notes

This project is about designing an optical computer based on Fourier transform architecture. Therefore, understanding how lenses perform Fourier transforms at the speed of light. Recreating this phenomenon using Python simulations of various images in a variety of ways. Ultimately, explaining why Fourier transforms performed by lenses are useful in terms of computing with reference to typical applications and interactions with other optical components. And deducing an effective design to overcome a set of predetermined difficulties based on the understanding produced from the discoveries.

Optical Correlator

- What the correlator is

An optical correlator is a device for manipulating the phenomenon of Fourier Transforms (in lenses) to compare two input signals for whatever means necessary. The signals are bidimensional functions with one being the image to be analysed and the other containing the reference information. Typically used for tracking systems and identification/recognition systems.

Phase correlation is another application of Fourier transform. Phase correlation is an estimative method for deducing the relative translation offset between two comparable images, or other forms of data. It relies on a frequency-domain depiction of the data (Fourier domain) and is usually used in image recognition. The term is applied in cross-correlation techniques that isolate the phase-angle information from the Fourier-space representation of the cross-correlogram. This is experimented with in the Python file "Python\Phase correlator\.Phase angle.py" and may be used later on dependent on how the course of the research pans out.

- Why it is useful

Digital computers can compute the calculations required to perform Fourier transforms however with increasing time required dependent on array size. These complex calculations can then be used in applications such as image template matching of phase correlation. However, when using a lens to perform the same set of feats, the calculations are performed at the speed of light and are not limited by the speed of electron flow in the computers or array size. This gives the possibility of performing image template matching at the speed of light on considerably large arrays.

- Discovered Correlation types and discussion

There are three common types of correlation being that of coherent light (LASERs), incoherent light, and the joint transform correlator. Comparing the coherent and incoherent, the incoherent correlator uses the same operating principles with the difference occurring with the products produced (spectra intensity of the two compared functions for the incoherent, and just the product of the two functions for coherent). For the comparison of the coherent and the joint transform correlator the difference occurs in the functioning principle whereby the Fourier Transform of both the reference and the image are performed simultaneously. Thus, negating the need of the adaptive filter. A common optical correlator is the 4F correlator where the image is Fourier transformed onto a filter and then inverse Fourier Transformed back again to be captured by a camera.

Python

- Why use Python

Python can be used in order to model the effects of the optical computer and collect data effectively. Python not only has a vast collection of existing libraries suited to the use of optical apparatus, but it also has a vast population of users able to help at any time regarding typical coding errors encountered. Python can take in an image to undergo a calculated Fourier Transform to produce an output image much like that of an optical correlator.

This is shown in the files of "Python\Shape comparison\.A1 FT.py" vs "Python\Shape comparison\.A2 FT.py" comparing the effect of location on a feature within the image being transformed so as to show the location is independent of the output image.

Other such tests on the operation of Fourier Transforms within Python are shown in the Shape comparison folder including inverting the colours in "Python\Shape comparison\.A White FT.py", the superposition of features in "Python\Shape comparison\.AO FT.py", and that of a more complex image like the Mona Lisa in "Python\Shape comparison\.Mona FT.py". The latter originally comes in the JPEG format however this causes Frequency-space compression issues compared to that of the PNG format which it was converted to. The transform of this complex image has the typical bright centre due to the lower frequencies undergoing superposition and creating a bright spot.

Due to the way lenses work it should also be possible to 'decode' a Fourier transform and undergo an inverse Fourier transform and see the original image by putting the Fourier image back through the lens in reverse. In terms of Python this is performing the inverse Fourier transform calculations as shown in the INV FT python files. This idea should be explored later.

- Timing in Python

The array size in the python images being processed has an impact on how quickly the calculations can be performed. Having array sizes in the 2^n scale is much faster to compute than some other random array size. To test this the timing function in python can be utilised to compare array sizes and see which performs the faster computations. However, it must be noted this should not affect the optical computer as it does not perform calculations, rather it simply performs superposition of the light where necessary according to the laws of diffraction. Refer to files in "Python\Timing" for simulations.

- Python Libraries discussion

NumPy: This library adds an extensive use of arrays and matrices into the python world to utilise. This is necessary due to the Fourier transform being calculated within the program. Due to it being an open software since 1995 and combining the prominent predecessors into a single library it combines a multitude of people's best work.

OpenCV: A library aimed at real time computer vision to allow images to be input into python quickly. It does so by accelerating GPU features used to process images. Popular uses of this library are in image tracking, 2D and 3D feature tool kits, facial recognition systems, object detection, amongst others. These same features are prominent in optical computers.

Matplotlib: This library is typically used for embedding plots into programmes in association with NumPy. Providing a variety of 2D or 3D plots in full colour makes it perfect for display input images, output images and graphs etc used in the simulated optical computer.

Prysm: A library dedicated to optical systems and their analytics. It requires the Jupyter Notebook which is an open-source web application for its statistical modelling features and data visualisation. Containing everything from films to transfer functions this library, being dedicated to optical engineering, will fit the needs of analysing the optical simulations.

Spatial Light Modulator 27/10/2020

- Research into what they are

A spatial light modulator (SLM) is an object that executes a spatially varying modulation on a ray of light such as in the case of an overhead projector transparency. Computers are used to typically control this transparency when the matter of an SLM is mentioned. Hence, typically SLMs are used in overhead projectors to project computer screen content. Future development has led to SLM technology being incorporated into the projector itself after the 1980s.

The transparency (commonly known as view-foil) is a slender sheet of transparent flexible material onto which diagrams can be drawn, such as cellulose acetate. The sheets are then projected onto a larger screen to be viewed overhead.

LCD technology is commonly used in projectors which, when applied in the way described, is termed a Spatial Light Modulator (SLM). The LCD is a transmissive form of SLM, whereas other technologies such as Texas Instrument's DLP are reflective SLMs utilising the same methodology but in a different manner. There are variations of projector which do not use transparencies in this way. Instead, they produce their own light source; an example of such is organic light-emitting diodes (OLEDs) used in projectors with no SLM technology.

- Discovered applications

Applications:

- Laser beam Modulation
- Coherent-wavefront Modulation
- Imaging/Projection technology
- Wavelengths Selective Switching
- Optical Tweezers
- Holography: memory, recording, security amongst other such technology to do with holography
- Laser beam Shaping
- Beam Splitting
- Phase Shifting

Masks for SLMs

- Types of masks and filters

Types of filters include Spatial Lowpass and Highpass, Butterworth Lowpass and Highpass, Gaussian Lowpass and Highpass, and finally phase-shift mask. Phase-shift masks are photomasks that take advantage of the superposition created by phase-differences. Usually used in improving image resolution (photolithography). There are alternating and attenuated phase-shift masks. It is based on the optical thickness of the medium the light is passing through to alter the phase by some extent.

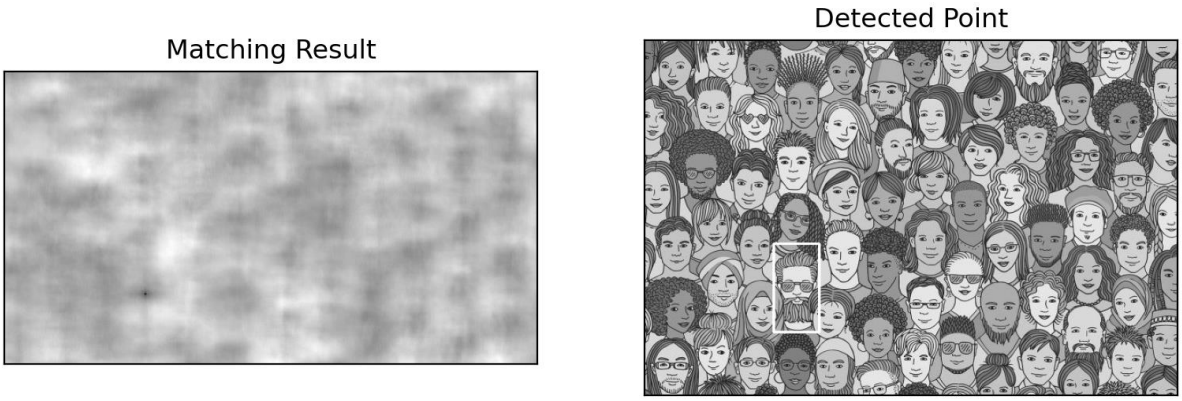
- Modelling the discoveries

Matching templates with transforms is a useful application utilising image correlation specifically cross-correlation. In signal processing, cross-correlation is a measure of similarity of two series as a function of the displacement of one relative to the other. It is

commonly used for searching a long signal for a shorter, known feature. It has applications in pattern recognition, electron tomography, averaging, cryptanalysis, neurophysiology, and single particle analysis. When modelling this it can be seen visually where some template matching methods work better than others. Specifically, the third method, Fig.3, of Template Matching Cross-Correlation does not work nearly as well as its counterparts.

(Refer to Appx.1 for the final code, or see "Python\Image Tracker\.Matching FT.py" for the file)

1. **method=TM_SQDIFF** (see Fig.1)



$$R(x,y) = \sum_{x',y'} \left(T(x',y') - I(x+x',y+y')\right)^2$$

Figure 1: TM_SQDIFF Python image

2. **method=TM_SQDIFF_NORMED** (see Fig.2)

$$R(x,y) = \frac{\sum_{x',y'} \left(T(x',y') - I(x+x',y+y')\right)^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

Figure 2: TM_SQDIFF_NORMED Python image

3. **method=TM_CCORR** (see Fig.3)

$$R(x,y) = \sum_{x',y'} \left(T(x',y') \cdot I(x+x',y+y')\right)$$

Figure 3: TM_CCORR Python image

4. **method=TM_CCORR_NORMED** (see Fig.4)

$$R(x,y) = \frac{\sum_{x',y'}\left(T(x',y') \cdot I(x+x',y+y')\right)}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$$

Figure 4: TM_CCORR_NORMED Python image

5. **method=TM_CCOEFF** (see Fig.5)

$$R(x,y) = \sum_{x',y'} \left(T'^{(x',y')} \cdot I'^{(x+x',y+y')}\right)$$

Where

$$T'(x',y') = T(x',y') - \frac{1}{w \cdot h} \cdot \sum_{x'',y''} T(x,y)$$

$$I'(x+x',y+y') = I(x+x',y+y') - \frac{1}{w \cdot h} \cdot \sum_{x'',y''} I'^{(x+x',y+y')}$$

Figure 5: TM_CCOEFF Python image

6. **method=TM_CCOEFF_NORMED** (see Fig.6)

$$R(x,y) = \frac{\sum_{x',y'}\left(T'(x',y') \cdot I'(x+x',y+y')\right)}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$$

Figure 6: TM_CCOEFF_NORMED Python image

Figure 7: Multiple Detections such as numbers 1-9 on a sudoku board

A variation including multiple detections in a single image was created in the file "Python\Image Tracker\Multi-image Tracker\.sudoku FT.py" (see Fig.7, Appx.2). However, it is unlikely this will be continued with the progress of the optical computer due to its complexity. If it is the case that a system with multiple detections in a single instance is required, then this can be used.

The different numbers 1-9 are denoted with a different colour for each match, for example every 4 is denoted by a yellow box.

4F Correlator

03/11/2020

- Typical setup

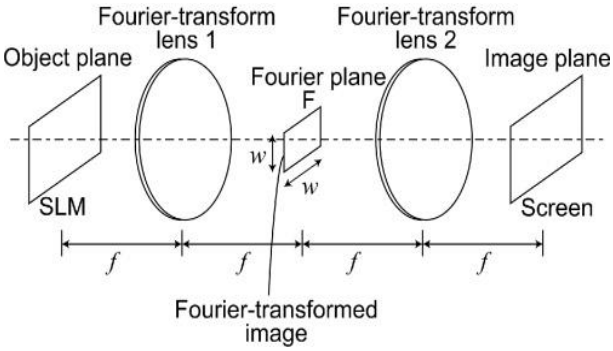


Figure 8: 4F Correlator typical setup with input on the left to output on the right

A typical arrangement in optical correlators is the 4F correlator. The input signal is written to a transmissive SLM which is illuminated with a laser. This is Fourier transformed with lens 1 and this is then modulated with a second SLM containing the filter. The resultant is again Fourier transformed with lens 2 and the result is captured on a camera or screen as shown in Fig.8.

- Researched uses of correlators

Filters used in the correlator are typically "Matched Filters", or "Phase-only Filters". The matched filter maximizes the signal-to-noise ratio and is simply obtained by using the Fourier transform as a filter of the reference signal. The phase-only filter is easier to implement due to limitation of many SLMs and has been shown to be more discriminant than the matched filter. Such uses of this are in cases like optical security verification whereby images are compared like a lock and key to determine if the correct key is presented at the input once it has undergone the Fourier Transformations.

- First full system models

Given the research conducted so far into how to utilise the Fourier Transformation to go from image to Fourier plane as demonstrated in the Shape comparison folder mentioned on ***Page 4 - Python, Why use Python.*** And the research into SLM filter types on ***Page 6 - Masks for SLMs, Types of Masks and filters***, it should be possible to construct the SLM filter types in python and combine them to create the first fully simulated 4F model from input to output.

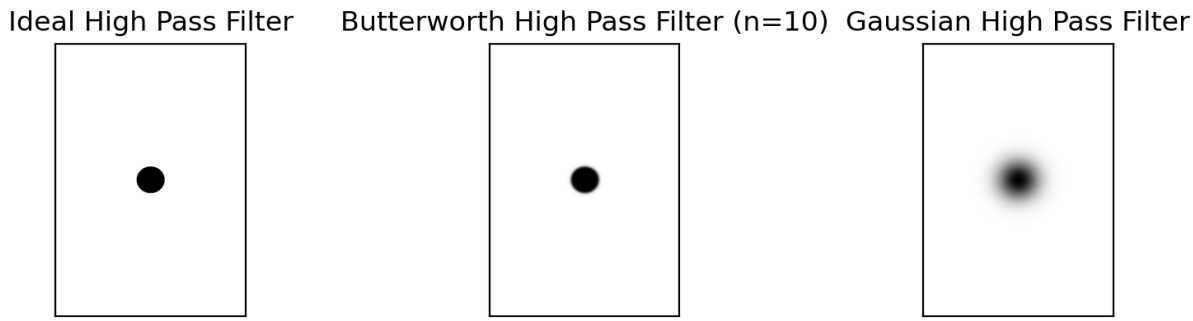


Figure 9: Ideal, Butterworth and Gaussian High Pass filter types simulated in python

The simulated Ideal High Pass, Butterworth High Pass and Gaussian High Pass Filters are shown in Fig.9 along with the Low Pass versions in Fig.10. The file location is "Python\4F Correlator\.All H+L Filter display.py".

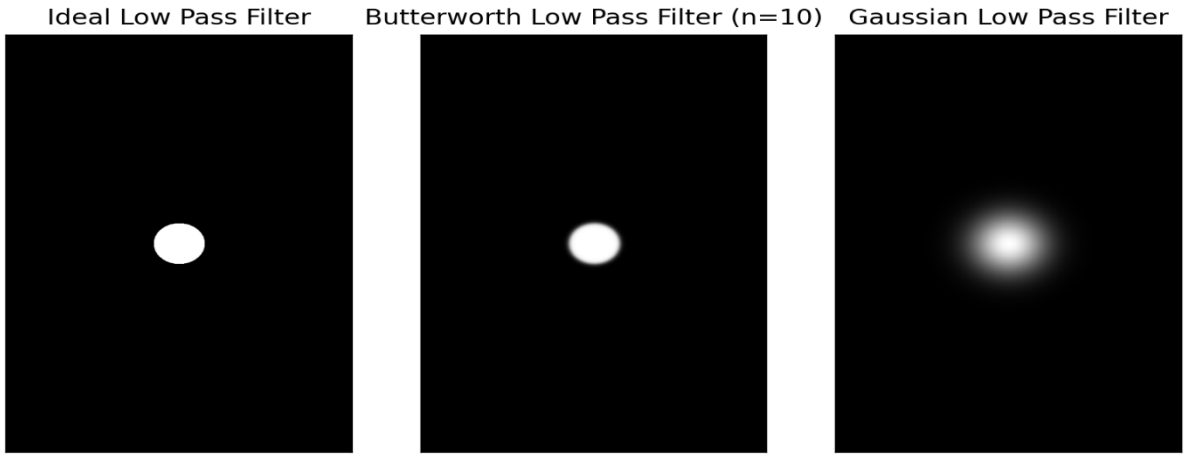


Figure 10: Ideal, Butterworth and Gaussian Low Pass filter types simulated in python

Following that, the first full system processed images were simulated using the combined programmes to date as shown in Fig.11 for the Low Pass filters as these are the clearer of the two groups of filters to visually see. File location of "Python\4F Correlator\.All LPF Processed.py"

Figure 11: First full system processed images for 4F correlator with all Low Pass filters

In order to better visualise the entire process from input image to final output image at each of the stages, another python programme was constructed showing each of the steps along the way from left at the input to the right at the output. This is shown in Fig.12 or seen in the location "Python\4F Correlator\.4F Ideal LPF.py"; the code for which is present in Appx.3.

The '*Fourier*' image is the process of putting the image through the first of the 2 lenses present in a typical 4F setup. Proceeding to the chosen filter type of a Low Pass Filter ('*LFP*') as shown. The '*Centered * LPF*' image is the multiplication of the latter 2 images to produce

this output. Going on to the '*Decentralised*' step whereby the zero-frequency component of the Fourier image which was previously shifted to the centre for visual purposes is now shifted back for calculation of the final 'Processed Image' whereby the second of the 2 lenses inverse Fourier Transforms the image back to a recognisable format. It can be seen upon inspection that the image has lost some of its detail which is to be expected given the LPF removes the higher frequency components of the image reconstructing to a blurred version of the 'Original'.

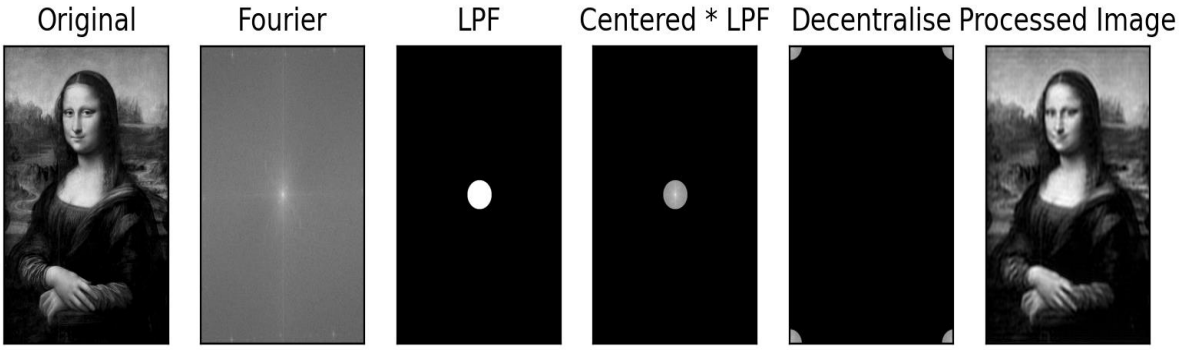


Figure 12: The first fully simulated stage progression of the optical computer; specifically, the 4F correlator

Charged-Couple Device (CCD and CMOS)

- Research into what they are

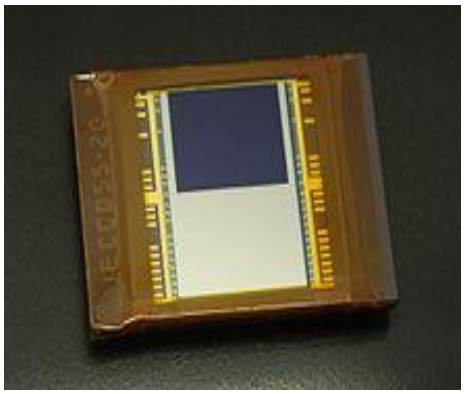


Figure 13: A typical frame transfer CCD sensor

A charge-coupled device (CCD) shown in Fig.13 is an integrated circuit containing an array of linked, or coupled, capacitors. Under the control of an external circuit, each capacitor can transfer its electric charge to a neighbouring capacitor. CCD sensors are a major technology used in digital imaging and are commonly used in camera technology for light detection. This makes a useful component for mapping the inbound light to the sensor in a digital format for further analysis or comparison purposes in optical computing.

- Why they are necessary

When the light from the input to the optical correlator has undergone each of the stages within the optical computer the resulting output is typically in the form of a 2-dimensional image which can be seen on a screen/ blank white sheet. However, in this form the output is not very useful analytically speaking. It would be better to map out that image on the screen and format it digitally onto a computer in order to commence analysis of luminance, position, contrast, or other such features in the image electronically. This is where the CCD comes in very useful as its ability to accurately map out the resulting processed image digitally allows such analysis to become possible. More specifically, the later generations

and evolutions of the CCD into the Complementary Metal Oxide Semiconductor (CMOS) technology can pick up more detailed data at the output stage for more accurate analysis.

Digital Micromirror Device (DMD)

- Research into what they are

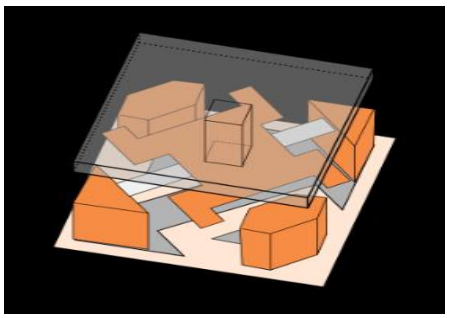


Figure 14: DMD showing the mirror mounted on the suspended yoke

In the DMD, shown in Fig.14, each micromirror corresponds to a pixel in the display. Its rotation is actuated by the electrostatic force between the mirror and electrodes underneath. The micromirror position is digital, in that it is positioned in one of two states created by rotating the micromirror until it contacts a mechanical stop. By doing this the DMD can cast an image specifically according to the digital input data.

- Why they are necessary

As opposed to using a laser or other strong light source to illuminate a transmissive SLM in order for the optical correlator to receive an input through the rest of the system, a DMD would allow the same input to the optical correlator but with a more detailed set of inputs possible. Being a reflective SLM in nature, the DMD is able to manipulate the phase and transparency of the input but with less losses due to the reflective nature of the system allowing near 100% transmission of the input compared to a transmissive SLM which would by its very nature absorb a portion of the input signal resulting in signal loss.

Optical Computer Full Example Setup 10/11/2020

- Setup and aims

The basics of the system require an image to be formed and a processed imaged to be outputted from an optical computation. A first attempt of such a computer is shown in Fig.15, which is essentially the optical layout for the simulations shown on ***Page 12 - 4F Correlator, First full system models***. This requires an input plane, namely a laser be it the 633nm He-Ne or the common 532nm laser. This then goes through a beam expander to make alignment and other such set up processes less errorsome proportionallity wise. Simply, with a larger beam and the same size errors in the adjustments, the proportionality of the errors decreases. With preferably as described above, a DMD capable of casting an image with minimal signal losses. The expanded beam then passes the DMD and the first SLM mask or kernel dependent of which optical computation it is undergoing. Once filtered as required it proceeds to the first of the Fourier Objectives in Lens 1. Another filter is

positioned on the Fourier plane between the two objective lenses again dependent on the type of computation the beam is undergoing. In the case of the diagram a High Pass Spatial Filter is shown. Finally going through the second lens to be analysed by the CCD camera at the 4th focal length from the start of the procedure. The CCD is used for its light detection capabilities in high detail and formatting that into a digital image to be sent back to the original computer for analysis and comparison.

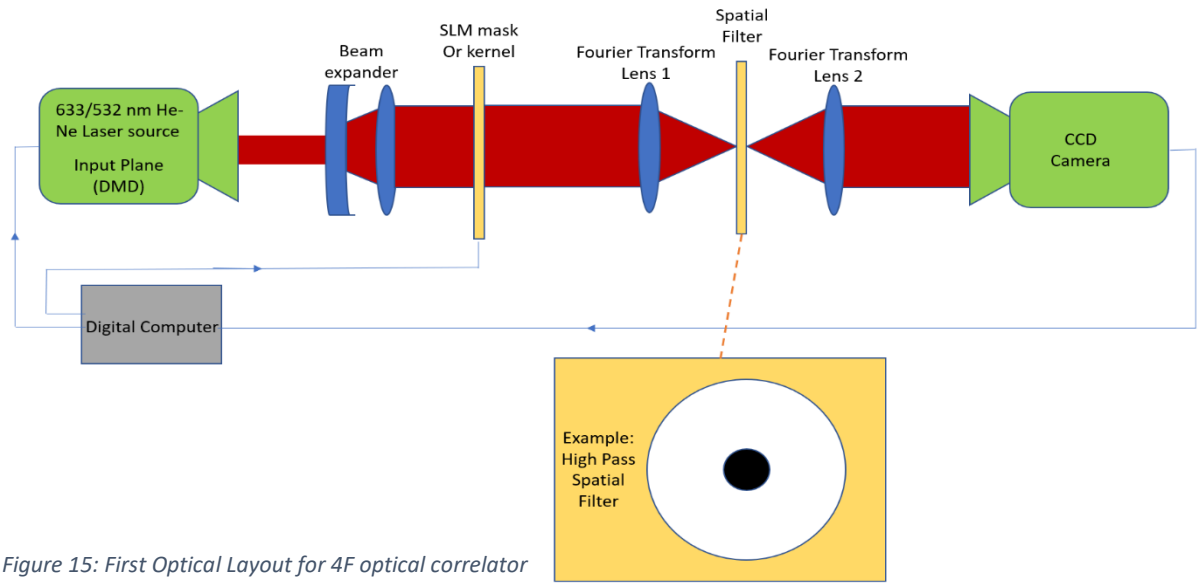


Figure 15: First Optical Layout for 4F optical correlator

- Weeks (1-5) & (5-10) Report

After constructing the Optical Layout diagram shown in Fig.15, it highlights the areas which require more detailed information to enhance the accuracy of the simulations. Such areas are the properties of the DMD producing the images, typically its limits of projection. For example, 'what happens to the processed data once the DMD begins to have pixel failure?' In terms of the SLM filters used for the kernels, the specific transmissive properties need to be researched to find their limits. Such as the case of speckled noise or defects entering the system both before and after the first lens and how this affects the outputted data to the digital computer. The lenses used for the Fourier Transformations themselves will exhibit effects dependent on the size and shape of the lenses according to the Modulation Transfer Function and other such aberrations associated with this. This needs to be researched in order to incorporate it into the design principles of the optical correlator. Finally, the CCD picking up the image must have limitations needing to be discovered in order to determine the maximum amount of degrading the entire system can undergo before the defects make the resulting data nugatory. Once all these areas have been looked at the overall design limitations of the optical computer will be complete.

Effect of Defects in the System 17/11/2020

- Types of Defect

Introducing defects into the system in order to determine the limits of the optical computer. A total of 2 types of defect are to be implemented at this stage, the first a speckled defect with random location covering the entire input image and the second an image containing failed pixel outputs.

- How to measure effects

- ### MSE advantages and its flaws

  When using the MSE to predict human perception in terms of image fidelity as well as image quality it becomes apparent that, dependent on the type of distortion upon images, the values of the MSE do not correspond to a visual human comparison of the same image distortions. Specifically taking the rotational distortion of an image resulting in a drastic increase in the MSE value when visually the image is almost identical in quality. A property of MSE is that if the pixels within the image are reordered in the original image and the resulting image in the same manner the MSE value will remain unchanged. This can cause a severe perceived fidelity issue wherein the reordered resulting image can look much noisier than the original resulting image when white gaussian noise is added.

- ### SSIM advantages and its flaws

  The Structural Similarity Index is a perceptual metric that quantifies the image quality degradation that is caused by processing such as data compression or by losses in data transmission. This metric is basically a full reference that requires 2 images from the same shot, this means 2 graphically identical images to the human eye. The second image generally is compressed or has a different quality, which is the goal of this index. SSIM is usually used in the video industry but has a strong application in photography and image processing. SSIM only measures the perceptual difference between two similar images. It cannot judge which of the two is better: that must be inferred from knowing which is the original one and which has been exposed to additional processing such as compression or filters.

  There are several types of Structural similarity algorithms in use such as Multi-scale SSIM, Percentage pooling SSIM, Complex-Wavelet SSIM, Gradient-based SSIM and Three-Component SSIM. They all aim to advance the predictive accuracy of SSIM relative to human bias however they also increase complexity of the computation.

Faster SSIM algorithms are in production drawing on the efficient performance of these different types individually in order to bring them together as a whole. Typically, SSIMs contain 3 different terms being that of the luminance, contrast, and structure of two comparative images to capture different characteristics of similarity between them.

$$\text{(Luminance) } l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$$

$$\text{(Contrast) } c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$$

$$\text{(Structure) } s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

Where $C_1 = (K_1L)^2$, $C_2 = (K_2L)^2$, and $C_3 = C_2/2$, are small constants; L is the dynamic range of the pixel values, and K1 ≪ 1 and K2 ≪ 1 are scalar constants. The constants C_1, C_2, and C_3 ensure stability when the denominator approaches zero (the singularity case). Combining the three terms, the common form of SSIM is:

$$SSIM(x,y) = [l(x,y)][c(x,y)][s(x,y)]$$

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

This results in an overall SSIM value which is the average of all the values within an 11 x 11 isotropic Gaussian weighting window passed over the image. This weighting window prevents unwanted artifacts arising during the image mapping stages. Looking explicitly at the components making up the SSIM value, the luminance term plays a less significant part than its counterparts perceptually. It should be noted the 3 components are entirely independent of each other. The contrast and structure terms involve the computation of variance which is the most time-consuming part of the calculation within the algorithm.

Limitations encountered with the SSIM tend to be with the image quality definition as well as the natural image complexity problem. Essentially although distortions may be clear visibly, they may not so objectionably. This does not equate visual error to that of quality loss and so, the association between image fidelity and quality is reasonable at best. Secondly, the complexity problem is how most psychophysical tests are carried out using simplistic patterns. However, these images are much simpler than that of any real-world image being tested with SSIM. Essentially the superposition of many hundreds of more simplistic patterns leading to the question; can a model really compare two more complex patterns and generalise hundreds of simpler patterns to evaluate how these more complex images are correlated.

- Defect in Original Image

```
C:\Windows\py.exe
------Original VS Processed 1------
==================================
MSE of Fourier Noise 1 to Fourier Original: 21.475008171246962
SSIM of Fourier Noise 1 to Fourier Original: 0.2125183631747335
_____________________________________
MSE of Noise 1 Output to Original Output: 23.616694610318028
SSIM of Noise 1 Output to Original Output: 0.42751474471631873


------Original VS Processed 2------
==================================
MSE of Fourier Noise 2 to Fourier Original: 29.92967827309339
SSIM of Fourier Noise 2 to Fourier Original: 0.1427126349444097
_____________________________________
MSE of Noise 2 Output to Original Output: 156.6581156902087
SSIM of Noise 2 Output to Original Output: 0.14946578430131005


-----Processed 1 VS Processed 2----
==================================
MSE of Fourier Noise 1 to Fourier Noise 2: 1.6328958213525744
SSIM of Fourier Noise 1 to Fourier Noise 2: 0.15694531988885455
_____________________________________
MSE of Noise 1 Output to Noise 2 Output: 64.50271611386874
SSIM of Noise 1 Output to Noise 2 Output: 0.7046980621567349
```

Figure 16: MSE and SSIM value comparison for the defect case implemented on the original image

The MSE (Mean Square Error) was found in several cases shown in, Fig.16, as well as the SSIM (Structural Similarity Index) for comparing images both before processing and after. It was found that although good results were obtained in this manner there is the possibility that noise comes in during the Fourier section rather than the Input Image which is later to be simulated. It can be seen from the image that the higher frequency in the Fourier Spectrum has been lost to noise.

It should also be noted in the final comparison of Processed 1 VS Processed 2, specifically the SSIM of Noise 1 Output to Noise 2 Output, that the value of 0.704 is notably high in comparison to the other values presented. This is essentially the level of similarity between the two noisy signal outputs. The images clearly have the starting feature (letter A) the same in both however this has only been at most 42% (0.427) correlated before with the lesser of the two noisy signals with the original signal. This means that some other starting feature common to both noisy signals has been picked up in similarity, which must be the noise itself. In other words, it suggests it may be possible to detect noise of this sort, and if it can be detected it could be accounted for in the design principles.

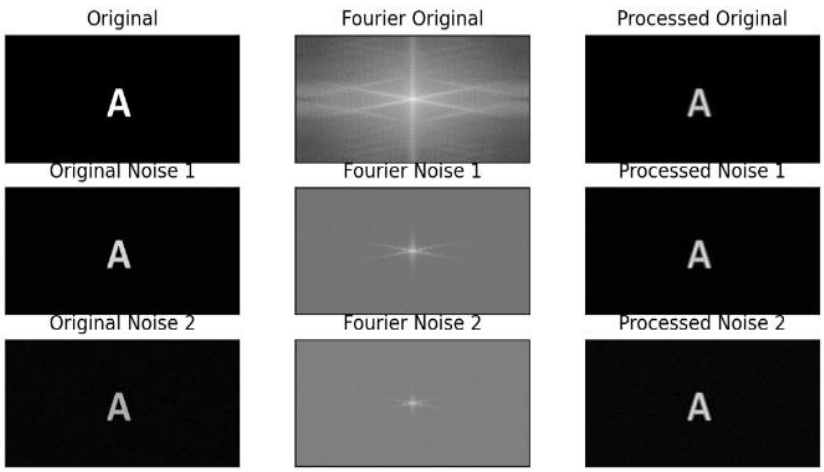


Figure 17: Visual representation of Original image under the effect of the defect, as well as the Fourier stages and final processed images used for the comparison

In Fig.17, the original image can be seen undergoing noise implementation which is then carried through and processed to show the result in a couple of noise levels as compared to the unaffected original for clarity. Despite the intense noise in the Fourier Plane the resulting output is visibly barely affected.

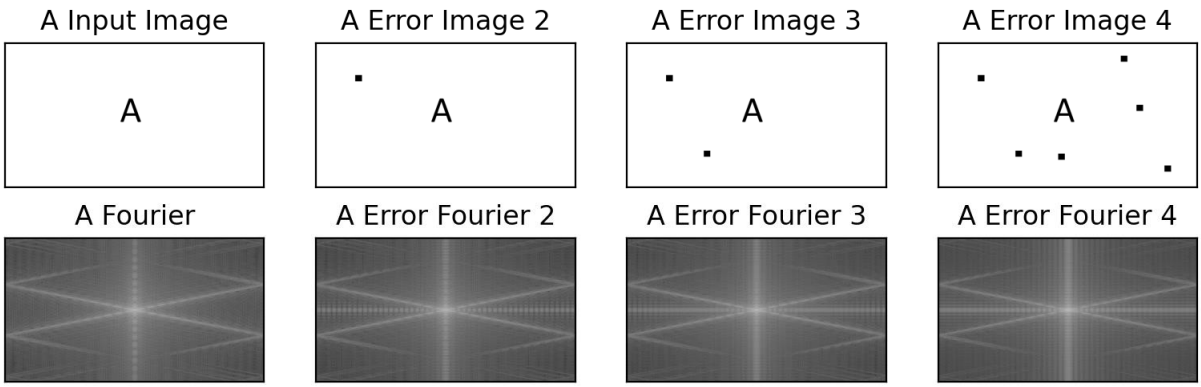


Figure 18: A variety of pixel defects in the original input image compared for effects in the Fourier Transform

There is also the second defect which is the DMD defect whereby a pixel failure occurs on the original image going into the first of the two Fourier lenses. This can be seen in Fig.18, where the colours have been inverted to make the pixel cleared to see compared to before.

As can be seen, there is minimal difference between the Original Fourier Transform and that of the Defected Fourier Transform. For the Defects 2, 3 and 4 there is an increasing number of pixels affected to see its overall effect on the Fourier Transform. Visually, despite the large number of pixel failures, the Fourier Transform for Defect 4 with 6-pixel defects is just as affected as the Fourier Transform for Defect 2 with only 1-pixel defect.

Essentially, the number of pixel defects present in the original image will all have the same effect on the Fourier Transform as a single pixel defect, dependent on the shape of the pixel defect and not its position. This could therefore potentially be accounted for in the final Fourier images and removed.

- Defect in Fourier Spectrum

Placing the spectral error during only the Fourier stage encountered several issues. Previously, the processed image was formed by taking the '**Image',** FT to get the '**Original'** then reallocating zero-frequency component to '**Centre'** before taking a magnitude to produce the '**Fourier Spectrum'** finally inverse FT the '**Centre'** to get '**Results'**. When the noise was implemented at the image stage the process was not affected at all. However, upon implementing it during the midway stage issues occurred.

The noise implemented alters what the starting image would have looked like for it to produce a Fourier Transform with such features. This original image is not known and cannot be found. By placing noise on the '**Fourier Spectrum'** image it becomes impossible to inverse Fourier transform as the '**Centre'** image makes the '**Fourier Spectrum'** but the **'Centre'** image is needed to be inverse Fourier Transformed and the act of taking the magnitude and absolute values of something cannot be reversed to find what it was before this happened. Instead, the noise itself had to be rewritten to allocate for this issue. Such that, the noise is implemented in the '**Original'** image and carried through the steps to the '**Fourier Spectrum'** to produce the noisy Fourier image. Therefore, the noisy '**Centre'** used

to create the Fourier is now known and this can be used to inverse Fourier transform the Spectrum back into the simulated starting Image. This can be seen in the Python code present in "Python\Error\.Noise in Fourier FT.py", and the output of this can be seen in Fig.19 below. The code can be seen in Appx.4 as well for clarity.

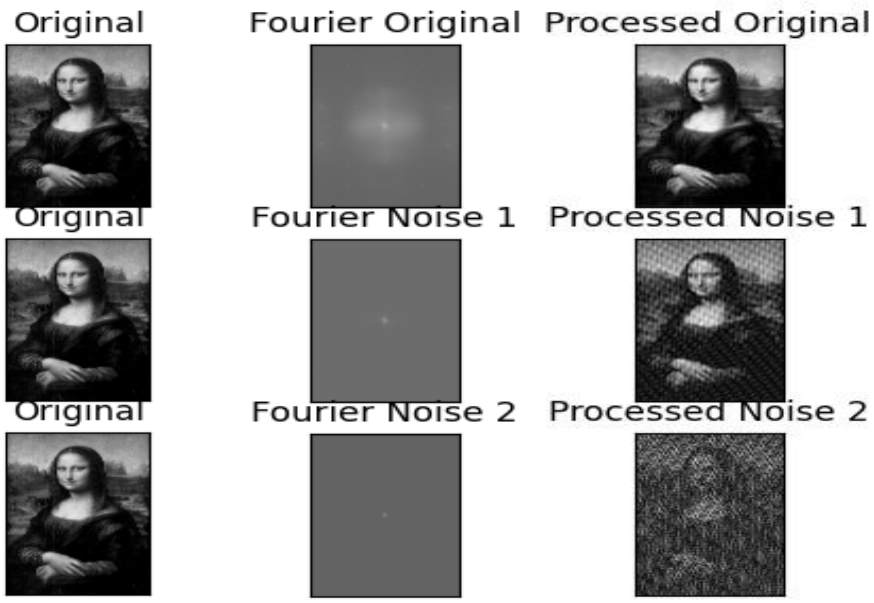


Figure 19: Mona Lisa image with a variety defect levels implemented during the Fourier stage

It can be clearly seen here that the level of noise on the Output image for the increasing defect level is progressively worse. To the point the Processed Noise 2 image is barely recognisable. However, when seeing the noise in the Fourier Transforms for comparison, there is visually little extra noise between the images. In other words, the effects are amplified during the inverse Fourier Transform process to create a noisier resulting image.

```
C:\Windows\py.exe
------Original VS Processed 1------
==================================
MSE of Fourier Noise 1 to Fourier Original: 8.439261197388909
SSIM of Fourier Noise 1 to Fourier Original: 0.3033136697597349
_____________________________________
MSE of Noise 1 Output to Original Output: 919.2032311993185
SSIM of Noise 1 Output to Original Output: 0.6263275654035345


------Original VS Processed 2------
==================================
MSE of Fourier Noise 2 to Fourier Original: 14.288028058757336
SSIM of Fourier Noise 2 to Fourier Original: 0.23304867839771543
_____________________________________
MSE of Noise 2 Output to Original Output: 5308.5265317284675
SSIM of Noise 2 Output to Original Output: 0.4484609377378525


-----Processed 1 VS Processed 2----
==================================
MSE of Fourier Noise 1 to Fourier Noise 2: 0.9379515603737263
SSIM of Fourier Noise 1 to Fourier Noise 2: 0.8925565067295951
_____________________________________
MSE of Noise 1 Output to Noise 2 Output: 4214.318646911987
SSIM of Noise 1 Output to Noise 2 Output: 0.5349880166738737
```

Figure 20: MSE and SSIM value comparison for the defect case implemented on the Fourier image

In Fig.20 is the MSE and SSIM values for the process shown in Fig.19. As was seen in the previous set of results when the defect was present in the original images, the SSIM values for the Processed 1 VS Processed 2 are abnormally high again in comparison. Valued at 54% similarity between the two noise signals, compared to 62% for the lesser of the two noisy signals and the clear original. While not as high as was discovered in the previous case in Fig.16, these values are still high enough to prompt future investigation into if it is possible to detect noise in the image, and possibly counter it in some way once detected.

Modulation Transfer Function 24/11/2020

- What it is

When attempting to compare the performance of optical systems, a commonly used measure is the modulation transfer function (MTF). MTF is used for a variety of components such as a spherical singlet lens to more complex ones such as a multi-element telecentric imaging lens assembly. The significance of MTF is considered in the general principles of components, importance, and characterisation as shown below in practical examples.

Components: Resolution and contrast. Resolution is an imaging system's ability to distinguish object detail. It is often expressed in terms of line-pairs per millimetre (where a line-pair is a sequence of one black line and one white line). This measure of line-pairs per millimetre (lp/mm) is also known as frequency. The inverse of the frequency yields the spacing in millimetres between two resolved lines. Bar targets with a series of equally spaced, alternating white and black bars (i.e. a 1951 USAF target or a Ronchi ruling) are ideal for testing system performance. For all imaging optics, when imaging a pattern, perfect line edges become blurred to a degree, see Fig.21. High-resolution images are those which exhibit a large amount of detail as a result of minimal blurring. Conversely, low-resolution images lack fine detail.

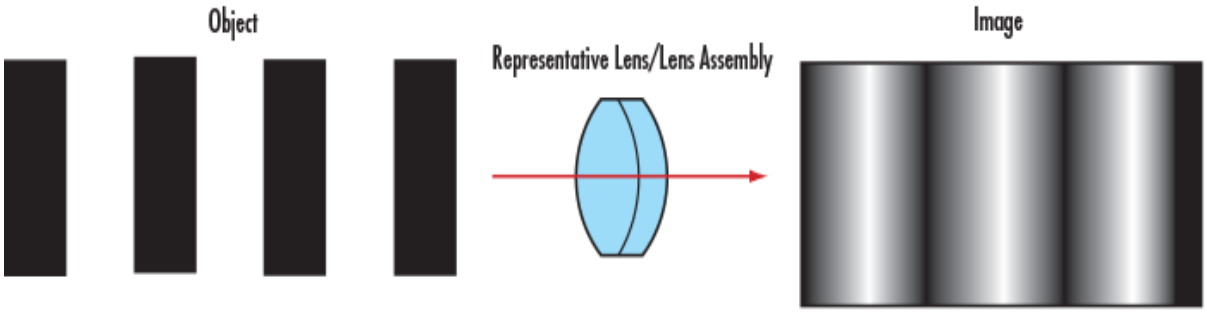


Figure 21: Visual representation of how an image is affected traveling through a lens leading to blurred edges and therefore loss of spatial frequency

Object resolution is calculated using the camera resolution and the primary magnification (PMAG) of the imaging lens (Equations 1 – 2). It is important to note that these equations assume the imaging lens contributes no resolution loss.

$$(1)\quad \frac{\text{Camera Resolution [µm]}}{\text{PMAG}} = \text{Object Resolution [µm]}$$

$$(2)\quad \text{PMAG} \times \text{Camera Resolution} \left[{}^{lp}/_{mm}\right] = \text{Object Resolution} \left[{}^{lp}/_{mm}\right]$$

Consider normalising the intensity of a bar target by assigning a maximum value to the white bars and zero value to the black bars as seen in Fig.22. Plotting these values results in

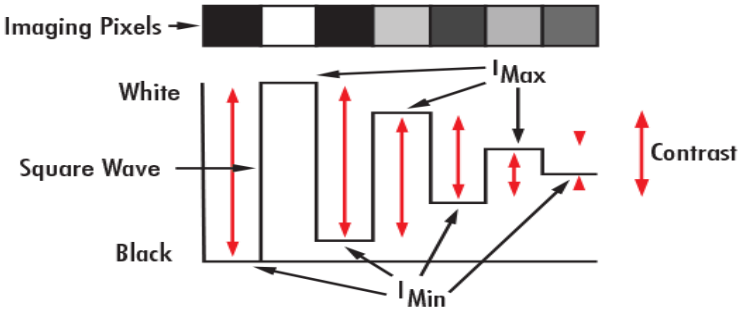


Figure 22: Assigning a maximum value to the white bars and zero value to the black bars

a square wave, from which the notion of contrast can be more easily seen. Mathematically, contrast is calculated by taking the labelled I_{max} values along with the labelled I_{min} values in the figure to the left and combining them as shown in Equation 3:

$$(3)\ \%\ \text{Contrast} = \left[{}^{I_{max} - I_{min}}/_{I_{max} + I_{min}}\right]$$

For an aberration-free image with a circular pupil, MTF is given by Equation 4, where MTF is a function of spatial resolution (ξ), which refers to the smallest line-pair the system can resolve. The cut-off frequency (ξ_c) is given by Equation 6.

$$(4)\ \text{MTF}\ (\xi) = \frac{2}{\pi}(\varphi - \cos\varphi \cdot \sin\varphi)$$

$$(5)\ \varphi = \cos\varphi^{-1}\left(\frac{\xi}{\xi_c}\right)$$

$$(6)\ \xi_c = \frac{1}{\lambda \cdot (f/\#)}$$

Importance: MTF is one of the best tools available to quantify the overall imaging performance of a system in terms of resolution and contrast, see Fig.23. This shows how the MTF values of different components in a system can be combined to give the System MTF. As a result, knowing the MTF curves of each imaging lens and camera sensor within a system allows a designer to make the appropriate selection when optimising for a particular resolution.

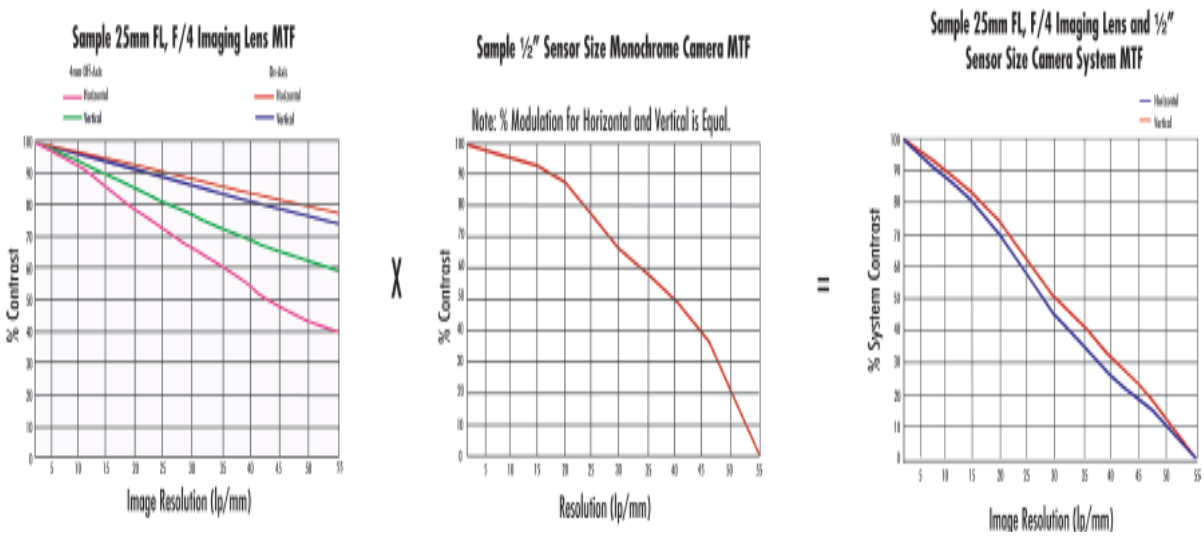


Figure 23: System MTF is the Product of the MTF of Individual Component: Lens MTF x Camera MTF = System MTF

Characterisation: Modulation transfer function (MTF) is one of the most important parameters by which image quality is measured. Optical designers and engineers frequently refer to MTF data, especially in applications where success or failure is contingent on how accurately a particular object is imaged. To truly grasp MTF, it is necessary to first understand the ideas of resolution and contrast, as well as how an object's image is transferred from object to image plane. Understanding and eventually interpreting MTF data is a very powerful tool for any optical designer. MTF can make selecting the appropriate lens a far easier endeavour - despite the multitude of offerings.

- Why this is useful to the system research

In order to design an optical computer all of the components must be considered, not only independently but as a whole, working together. The Modulation Transfer Function does exactly this. It allows the analysis of individual components such as the MTF of a lens used for Fourier Transforming the signal, as well as combining the effects of a signal being passed to another component in the system as shown in Fig.23. Simulating this effect would be useful for the same reason, and eventually incorporating it into the whole system simulation to give a more accurate representation of how defects affect such a system. Defining limits of individual components as well as the effects of components working together.

- Modelling the MTF

Using the Prysm library in Python as shown in Appx.5 in order to model a full optical system from start to finish to show its MTF. Consisting of the Fourier Transform lens and a CCD/sensor to pick up the light once it has passed through the lens as well as a Low Pass Filter, as was previously shown in the full system optical layout. Specifically, the system contains a circular aperture within a squared 10mm area as shown in Fig.24 with an effective focal length of 50mm, and the pixels on the sensor are 5 microns in diameter. The Point Spread Function (PSF) of this lens is shown in Fig.25.

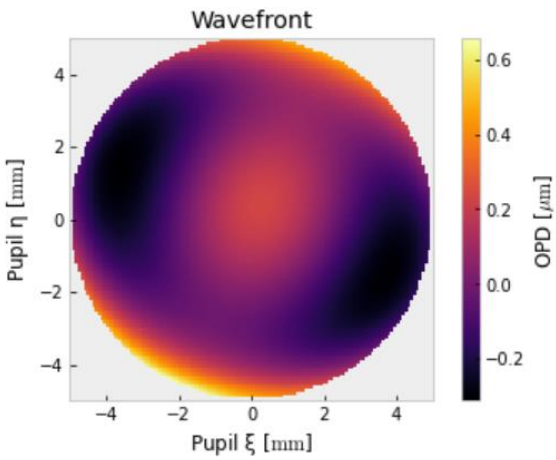


Figure 24: Circular aperture within a squared 10mm area showing the optical Path Difference levels in microns

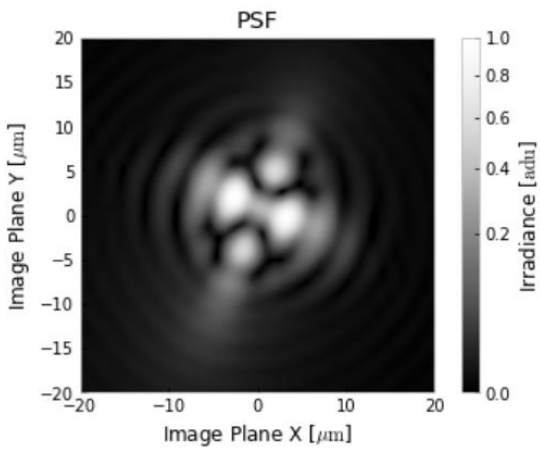


Figure 25: Point Spread Function showing the lens impulse response

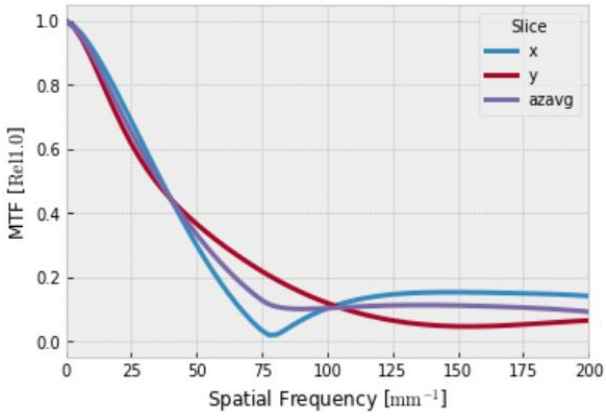


Figure 26: MTF of the aperture based on the PSF data

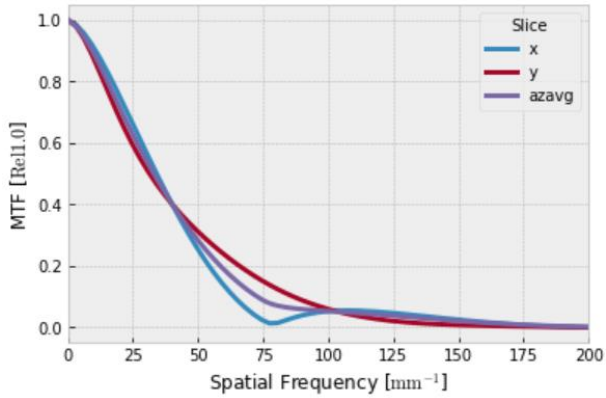


Figure 28: MTF of the entire system based on the PSF data

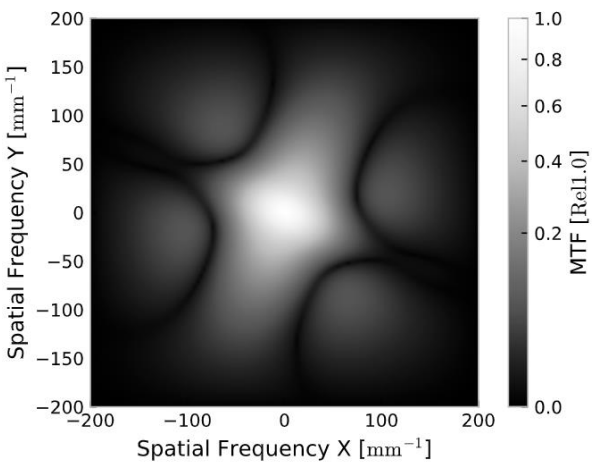


Figure 29: 2D map of the axial Spatial Frequencies according to their MTF levels

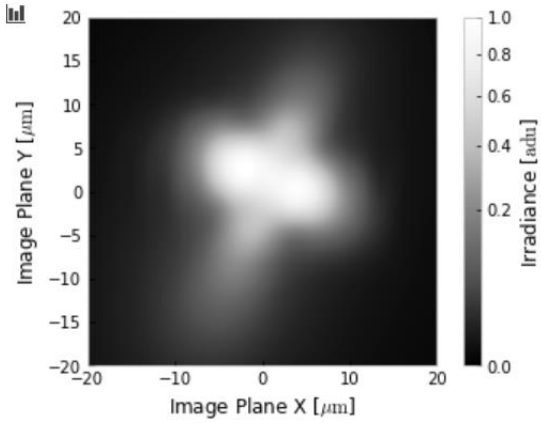


Figure 27: PSF of the entire system including the pixel in the CCD

The MTF of this aperture given these properties is shown in Fig.26. Progressing through the system leads to the PSF of the entire system shown in Fig.27 after having passed through the CCD/sensor pixels. And finally, the MTF of the entire system is shown in Fig.28. Putting the data from Fig.28 into another format so that the X and Y spatial frequencies can be viewed together produces Fig.29.

As it can be seen the Prysm library is a powerful tool for optical designers allowing all this data from only a few inputs. There is nothing specific to highlight in these figures as the data input is just an example to showcase the functions. Utilising these functions in cooperation with the previous python codes will enable greater depth in understanding what is happening at every stage of the optical computer. It can be said though that the MTF values of the entire system reach zero at a spatial frequency of $200\ mm^{-1}$, which when compared to the pixel size of 5 microns can be seen to be correct as in this case:

$$Frequency\ (mm^{-1}) = \frac{1}{Pixel\ size\ (mm)}$$

Future System Plans 01/12/2020

- Why more details are necessary

In the aforementioned topic on Modulation Transfer Function, the processing power of the Prysm library was shown, allowing detailed analysis of an entire system albeit a simplistic one. This ability would allow greater analysis of the previous systems described not only for their individual components but also as an entire system output. Therefore, by researching deeper into this area and attaining more details it would progress the design of the optical computer in all areas. It was noted the level of detail that is needed to be put into the Prysm system is based on real world values of the components in use such as the lenses and pixel sizes etc for the sensors. Therefore, more details behind the specifics of these components are necessary to utilise the entirety of the last program as currently only example runs are possible until specific system designs are suggested. And so, below are lists of the available components in a variety of areas which may be necessary in advancing the use of Prysm, specifically based on the previously looked at areas of the optical computer.

- Commercially available SLMs

The table below in Table.1 (Ref.1) provides an overview of commercially available SLMs whose principles of operation are based on the use of liquid crystals (LC), and microelectromechanical systems (MEMS). In the case of LC SLMs, two different manufacturing technologies can be distinguished, using transmissive liquid crystal (TLC) and reflective liquid crystal on silicon (LCOS) structures.

Also worth highlighting, is the group of liquid crystal SLMs based on ferroelectric LC (FLCOS), which have the highest frame rate among all LC-based devices, as well as some unique modulation properties. MEMSs are represented by two fundamentally different types: systems based on a micromirror matrix, such as DMDTM (digital micromirror device) and AMM (active micromirror matrix), and systems based on grating light valve (GLVTM). Many of these devices have specific features that must be considered when designing an optical system. Thus, the specific features of Hamamatsu (X13138) liquid crystal SLMs are linear phase characteristic, high accuracy of phase control, and high light efficiency for specific wavelengths due to the use of a dielectric mirror.

The Syndiant (SYL2341) devices have the smallest minimum pixel size, they are the most compact and, therefore, in demand for head-mounted displays. The frame rate of all types of MEMS-based SLMs ranges from 10 kHz to 1 MHz, which is especially attractive for the development of high-transfer-rate optical digital devices. However, it is important to keep in mind that devices of the DMDTM and AMM types have a fundamentally binary amplitude modulation characteristic, and GLVTM-based devices, as a rule, have high resolution in only one spatial coordinate, so that for operating in 2D mode in an optical scheme temporal scanning in low-resolution coordinate is necessary. The frame rate ~1 MHz of the 1D LM device (Fraunhofer IPMS) allows achieving a frame rate up to 500 Hz for a 2D function

consisting of 2000 lines, which is comparable to the frame rate for the best LCD-based SLMs, and the resolution of 8000 pixels per line makes possible imaging with record-breaking spatial resolution.

The Santec (SLM-200) devices possess high flatness (up to l/40), as well as high input signal resolution of 10 bits, which makes it possible to carry out high-precision control of the beam phase. The Meadowlark Optics (HSP1920) device is controlled by an analogue driver, which allows increasing the frame rate to 422 Hz.

Model	Manufacturer	Technology	Resolution	d_0 /μm	v /Hz
LCX202A	Sony	TLC	1924 × 1024	7.2	240
SXRD241A	Sony	LCOS	1920 × 1080	4.25	60
4K	Compound Photonics	LCOS	4096 × 2160	3.015	120
HX7318	Himax	LCOS	1366 × 768	6.0	360
D-ILA	JVC	LCOS	4096 × 2400	6.8	220
LC-2012	Holoeye	TLC	1024 × 768	32	60
GAEA-2	Holoeye	LCOS	4160 × 2464	3.74	60
LETO	Holoeye	LCOS	1080 × 1920	6.4	60/180
X13138	Hamamatsu	LCOS	1272 × 1024	12.5	60/120
OP02220	OmniVision	LCOS	1280 × 720	4.5	300
RDP700Q	RaonTech	LCOS	2560 × 1440	6.05	360
SLM-200	Santec	LCOS	1200 × 1920	8.0	60/120
SYL2341	Syndiant	LCOS	3840 × 2160	3.2	60
HSP1920	Meadowlark Optics	LCOS	1920 × 1152	9.2	30–422
JD4704	Jasper	LCOS	4160 × 2464	3.74	180
QXGA	ForthDD	FLCOS	2048 × 1536	8.2	5700
UXGA	MDCA	FLCOS	1600 × 1200	6.3	540
DLP7000	Texas Instruments	DMD	768 × 1024	13.6	32552
DLP9500	Texas Instruments	DMD	1080 × 1920	10.8	23148
1M SLM	Fraunhofer IPMS	AMM	2048 × 512	16	2000
1D LM	Fraunhofer IPMS	AMM	8000 × 1	10	1 MHz
G8192	Silicon Light Machines	GLV	8192 × 1	5	250 kHz

Note: d_0 is the pixel size and v is the frame rate.

Table 1: A table of Commercially available SLMs as of 2020

- Commercially available DMDs

In Table.2 (Ref.2) is a list of commercially available DMDs akin to that of the SLMs list above.

Description	Display Resolution (Max)	Input frame rate (Max) (Hz)	Array diagonal (in)	Micromirror array size
DLP® 0.47 4K UHD HSSI DMD	4K UHD (3840x2160)	60	0.47	1920x1080
DLP® 0.23 HD DMD	720p (1280x720)	60	0.23	960x540
DLP® 0.2 nHD DMD	nHD (640x360)	120	0.2	640x360
DLP® 0.2 WVGA DMD	WVGA (854x480)	120	0.2	854x480
DLP® 0.23 qHD DMD	qHD (960x540)	120	0.23	960x540
DLP® 0.23 1080p DMD	1080p (1920x1080)	60	0.23	960x540
DLP® 0.3 720p DMD	720p (1280x720)	120	0.3	1280x720
DLP® 0.33 1080p DMD	1080p (1920x1080)	60	0.33	1368x786
DLP® 0.47 4K UHD DMD	4K UHD (3840x2160)	60	0.47	1920x1080
DLP® 0.47 1080p DMD	1080p (1920x1080)	120	0.47	1920x1080

Table 2: A table of Commercially available DMDs as of 2020

- Commercially available CCDs

Below in Table.3 (Ref.3) is a list of the commercially available CCDs akin to that for the SLMs.

Part Number	Focal Length	F/No.	Format Size	Mount
VP611	1.5mm	2.7	1/4"	M12 P=0.5
VP601	1.84mm	2.6	1/4"	M12 P=0.5
VL-2020	2.0mm	2	1/3"	M12 P=0.5
VL-2520	2.5mm	2	1/3"	M12 P=0.5
VL-2818	2.8mm	1.8	1/3"	M12 P=0.5
VP-283	2.8mm	2.8	1/3"	M12 P=0.5
PTS-2.9M12	2.9mm	4.5	1/4"	M12 P=0.5
VL-3120M12	3.1mm	2	1/3"	M12 P=0.5
PTS-3.2M12	3.2mm	4.5	1/3"	M12 P=0.5
VE300UA	3.6mm	2	1/4"	M12 P=0.5
VE300UB	3.6mm	2	1/4"	M12 P=0.5
VL-3620	3.6mm	2	1/3"	M12 P=0.5
VL-3650	3.6mm	5	1/3"	M12 P=0.5
PTS-3.7M12	3.7mm	4.5	1/2"	M12 P=0.5
VL-4018	4.0mm	1.8	1/3"	M12 P=0.5

Table 3: A table of Commercially available CCDs as of 2020

- Plan for optimising the design of the system

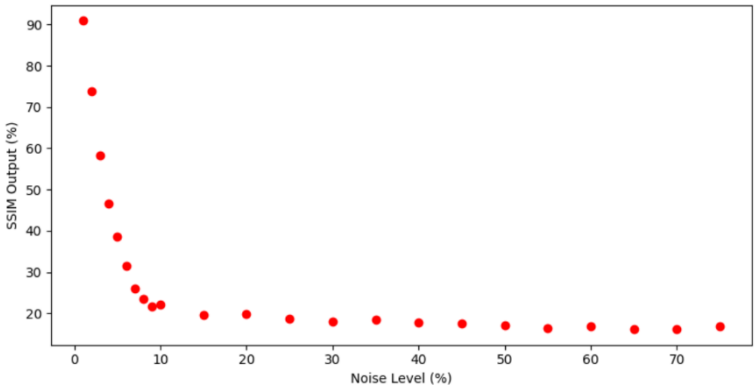


Figure 30: SSIM values for the 'A' image output as noise levels are increased in the Fourier Stage

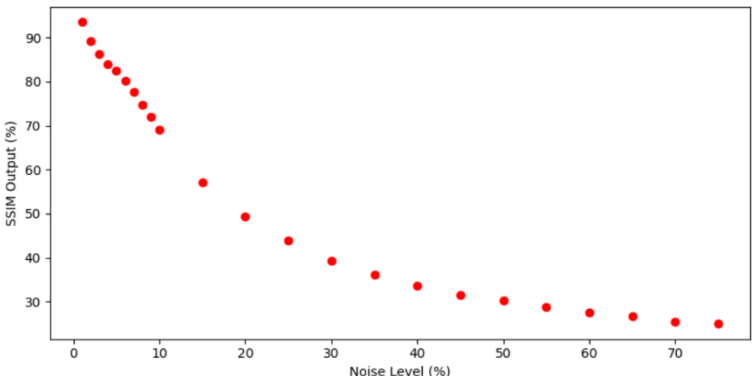


Figure 31: SSIM values for the 'White A' image output as noise levels are increased in the Fourier Stage

Upon seeing the clear difference in outputs when the noise levels are varied on ***Page 19 – Effects of Defects in the System, Defect in Fourier Spectrum***, it became apparent to graph the SSIM values against the noise levels for several results, not just the 2 found before, in order to see the relationship they share. The first set of results (see file location: "Python\Error\..SSIM vs Noise.py") shown in Fig.30 exhibit a clear cut off point at approximately the 10% noise level input which is where the similarity loss between images apparently ceases.

This could be due to the images starting to look almost entirely speckled and so are 'similarly speckled' images rather than 'similarly A' images. To test this the greyscale on the A image was inverted to give a larger white background. A clear difference can then be seen in the cut-off point whereby it leaps from around 10% to around 20% but much harder to say exactly where this time due to less clarity, see Fig.31.

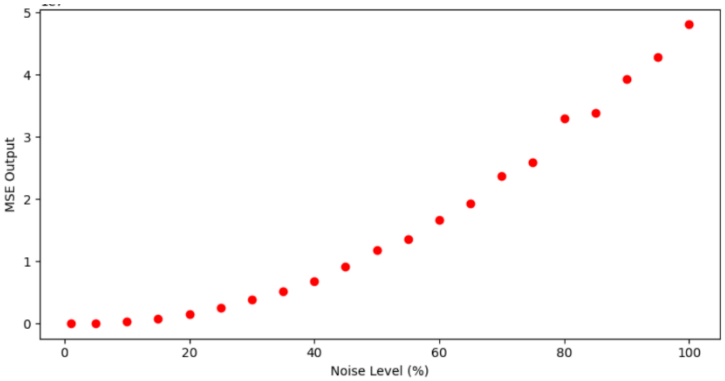


Figure 32: MSE values for the 'A' image output as noise levels are increased in the Fourier Stage

Furthering this, the MSE values were also plotted in each case (Fig.32 and Fig.33 respectively) and finally the RMSE values in each case (see file location, "Python\Error\..RMSE vs Noise.py") to achieve as much understanding as possible of the relationships at hand.

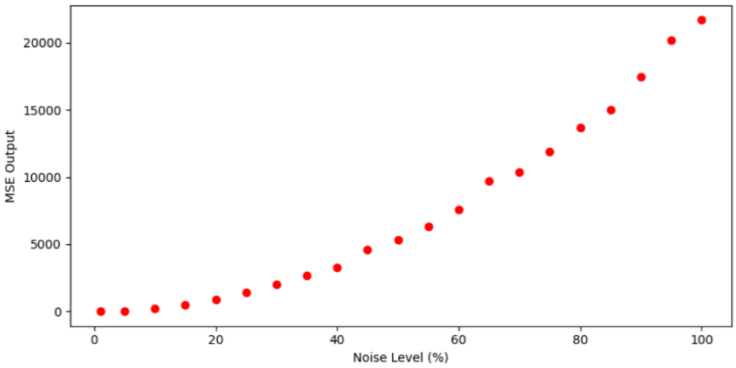


Figure 33: MSE values for the 'White A' image output as noise levels are increased in the Fourier Stage

Next the Mona image is tested and graphed in SSIM values for comparison on a much more structural image, Fig.34. It should be noted loading times become significantly longer due to the different array sizes which may need to be optimised for future progress in this area.

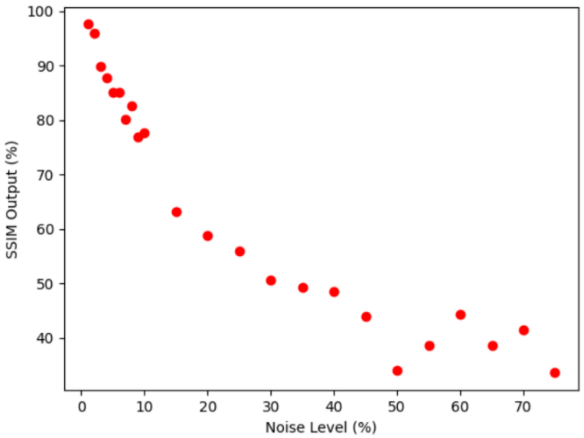


Figure 34: SSIM values for the 'Mona' image output as noise levels are increased in the Fourier Stage

As can be seen by comparing these graphs there is some similarity in their structure but dependent on probably the complexity of the image there is variation in the error of the SSIM values (Fig.34 specifically). To optimise the designs in the future, a way of measuring this error must be found and plotted. Not only that, but a larger variety of images must be tested with different properties to distinguish what is causing these fluctuations, and therefore why. Once that is understood, then the defects entering the system in terms of speckled noise will be better understood and be incorporated into the design principles. Once the fundamentals of how noise affects the outputs is recognised, then the application of the aforementioned specific components can begin. And with it the optimisation of choices based on how the components interact in terms of their MTF on the system. For a final goal for the system, a specific example should also be chosen for the use

of the optical computer to test the full set of design principles on. Such as a recognition system designed to find Wally in the famous 'Where's Wally' images, using the aspects discovered here and during the proposed future research to combine chosen components effectively to overcome possible defects implemented in the system to the best of its abilities; or a tracking system for monitoring bacterial movement designed to overcome implemented defects in various stages of the computer, while still being able to decipher different movement features as best it can to identify the bacterium.

Christmas Holiday Work 15/12/2020

- What are Holograms

Holograms are constructed using the science of holography. Holograms use diffraction rings from interference patterns to create the three-dimensional light field, thus still containing depth in a two-dimensional image. When this two-dimensional image is lit correctly, the interference pattern in the image performs diffraction on the incident light to reconstruct the original light field formed from the three-dimensional object that the interference came from. Specifically, an illumination beam and a reference beam are required to record and create the holographic image. The illumination beam acts on the object in question creating an object beam which is picked up by a photographic plate (such as silver halide). At the same time, the reference beam is also picked up in phase with the object beam by the photographic plate creating the hologram. The process of reconstruction is effectively the reverse of this recording process whereby a plane reconstruction beam is acted upon the photographic plate creating reconstructed wavefronts away from the plate. These wavefronts correspond to the original image in three-dimensions therefore a virtual three-dimensional image is seen because of it.

- Why this is useful here

Invented by Dennis Gabor in 1947, it has since progressed into many practical areas for utilisation such as holographic video microscopy. This can be used to study colloidal spheres in three-dimensions typically for tracking purposes. Such spheres as bacteria on the microscopic scale would be able to be tracked for their movement. It should be noted it is in this movement that the bacteria are able to be specified as certain bacteria move in certain ways. The issue that arises when tracking bacteria is that they move in a three-dimensional environment and therefore would move in and out of focus of a lens. However, using holographic techniques the bacteria moving out of focus is no longer an issue as this will create diffraction rings in the hologram. These rings can be used to reconstruct the original three-dimensional image and thus tracking of the bacteria is possible without the issue of constantly refocusing the lens. A typical reconstruction routine used for back-propagation of the light field is that of Rayleigh-Sommerfeld. This gives an approximate analysis of the hologram and so, analysis of the errors and such associated with this holographic technique

would better allow the design of optical computers to be used in this area of research. Optical computers are better suited to this area of research due to the utilisation of holograms which involve diffraction patterns. It is these diffraction patterns which make optical computers superior for analysis over their counterpart as diffraction is more closely related to Fourier Transform mathematics; the same mathematics looked at previously in the course of this research.

- Diffractio Library

This is a library for diffraction and interference optics analysis specifically using the Rayleigh-Sommerfeld back-propagation routine for calculations. It is written in Python akin to the previous libraries talked about thus far. It deals with various scalar fields and masks in two or three dimensions which would be a useful starting point to get accustomed to using holographic data and analysis. It uses Mayavi for the three-dimensional object recreations as shown in Fig.35 below. The top images and the bottom left image are plane images able to be moved across the volume to see the two-dimensional data and the bottom right image is the entire volume representing only the intense accumulations of wavefronts. It can be seen that the wavefronts focus down to a point in the volume. The data inputs used are the wavelength of light set at 0.642 microns, the refractive index of the medium at 1.3326, and several lens parameters used in this example test run.

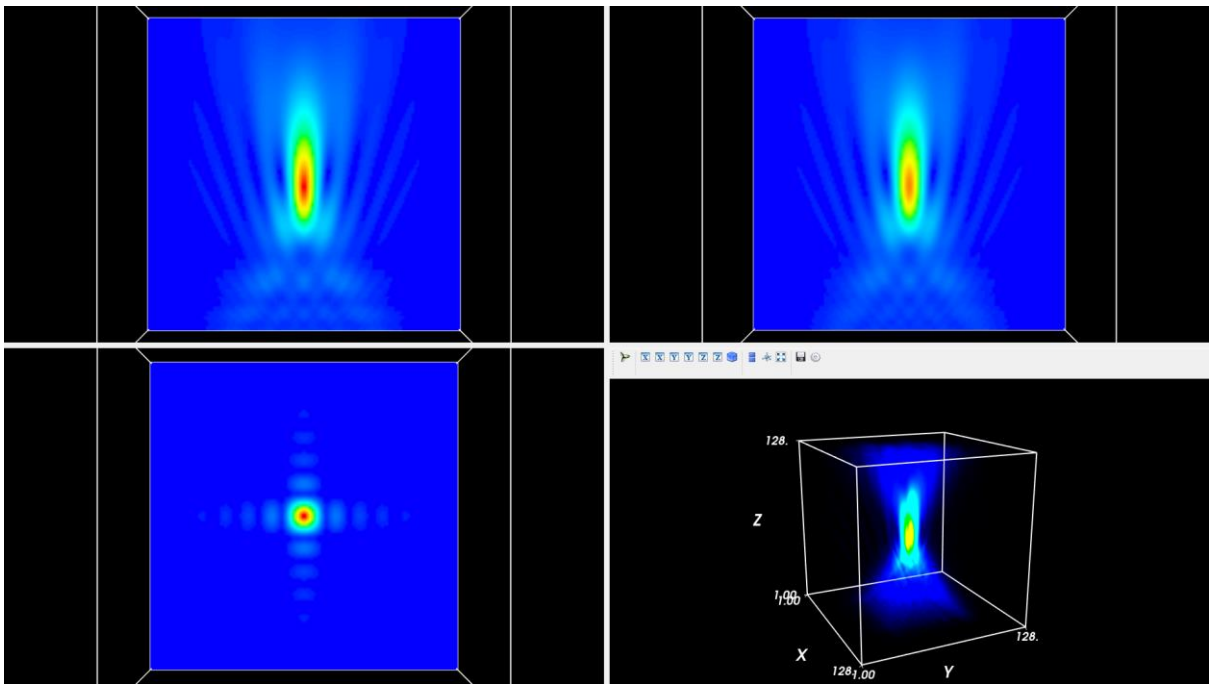


Figure 35: Diffractio example test run using Rayleigh-Sommerfeld reconstruction routines to recreate a volume space. The top images are the X and Y planes in 2D respectively (left to right), and the bottom left image is the Z plane. The bottom right is the reconstructed 3D volume showing intense points of superposition only

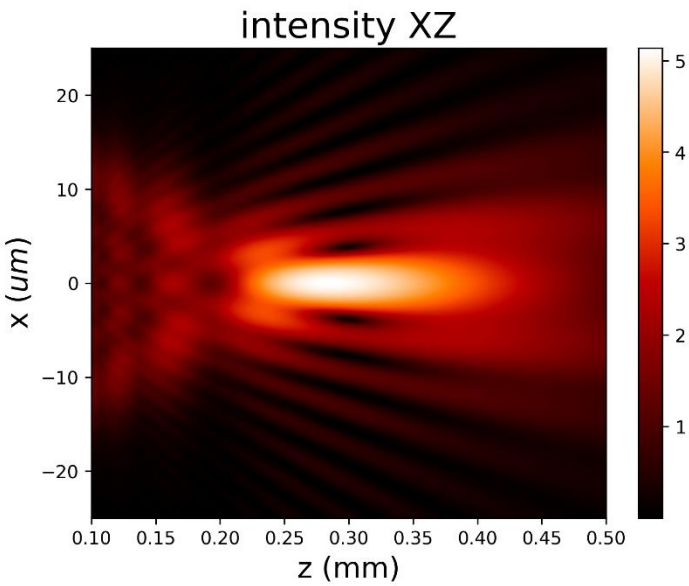


Figure 36: XZ intensities from the reconstructed volume in Fig.36

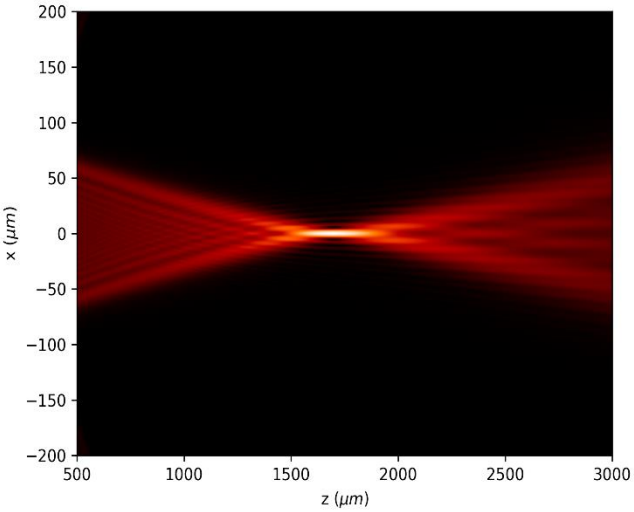


Figure 37: XZ intensities from the Rayleigh-Sommerfeld routine for a second example lens

The python coding is also designed to output the XZ intensity from the Mayavi plot for more accurate visualisation and analysis as see in Fig.36. It can be seen that towards the lower Z range that the wavefronts appear to form possible rings and as they move along the Z-dimension they combine to form a solid focused point. In terms of future analysis, this will be the reconstructed 3D object (bacteria specifically) from holographic data. Focusing in more detail it would be useful to redo a similar example in just two dimensions to better see the level of accuracy capable.

Therefore, Fig.37 shows just the reconstruction of a 2D plane specifically the XZ intensity again over a larger space and a different lens in terms of focal length and radius parameters. As was seen in Fig.36 the same focal bright point is shown and likewise clearer rings towards the lower Z values. However, now towards the higher Z values it can also be seen that the rings again begin to form as it once more goes out of focus. Along with Fig.37 the Python program creates two more files showing the Mask and the Source used in the calculations for clarity. These are shown in Fig.38 and Fig.39 respectively for clarity.

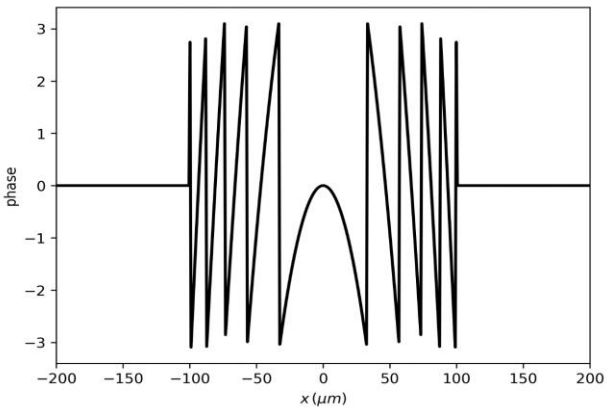


Figure 38: Mask from calculations in Fig.37

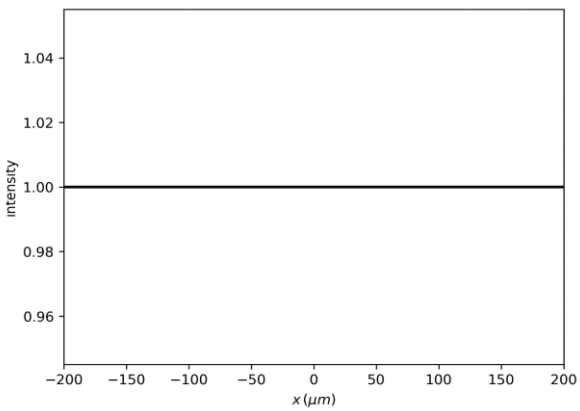


Figure 39: Source from calculations in Fig.37

Using Diffractio has shown the level of detail capable when using the Rayleigh-Sommerfeld routine to reconstruct objects causing the diffraction patterns however it was done by passing light through lenses and seeing the focal point rather than actual objective reconstruction. This does use the same physics principles, but real-world data would be better for accuracy during analysis. Therefore, using the same knowledge acquired during the Diffractio testing, moving on to Holopy testing in the future should be the goal.

- Holopy Library

This is again a library in Python for continuity and possible future combination of previous python adventures. It is used for holographic analysis and light scattering phenomenon specifically in backward propagation of light, reconstructing 3D volumes. Originating in the Manoharan Lab at Harvard, this python library should be capable of performing real-world reconstruction of objects forming the holograms. Some example holographic data provided is shown in Fig.40 where the Image is the 2D plane from a microscope looking at a collection of bacteria as was suggested on ***Page 27 – Future System Plans, Plan for optimising the design of the system***.

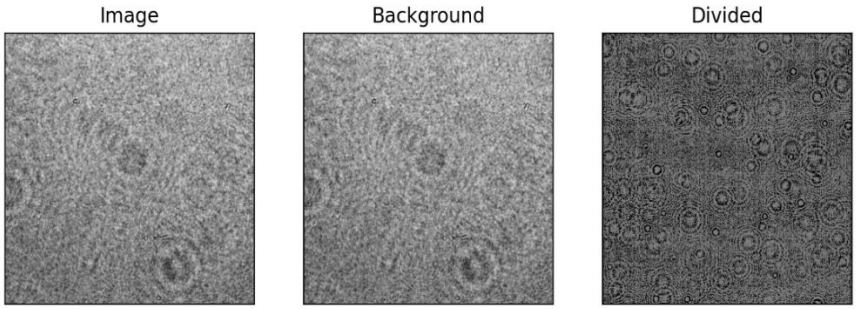


Figure 40: The real-world data in the Image and Background sections with the calculated Divided output from array values being the hologram to use

The Background is the same 2D plane from the microscope without the objects of interest. The resulting Divided image is the output when the array values in the preceding two images are divided (Image array/Background array). This is done using OpenCV for the sake of better seeing what the Divided image looks like visually. This is the hologram which is to be utilised by Holopy library in due course.

The Holopy input parameters are the Image and Background arrays as were shown in Fig.40 as well as the refractive index of the medium being 1.3326, the wavelength of the light source being 0.642 microns, and finally the image scale in terms of microns/pixel which is 0.703. From this the reconstruction of the volume can begin showing frames of the hologram as the Z-dimension is increased. In other words, as the volume is moved through along the Z axis, the slices of the volume in the XY plane are outputted. This can be seen in Fig.41 where several slices of the volume have been shown in progression starting in the top left and moving in a reading fashion (left to right and down). In these images two areas have been highlighted for particular attention. That of the blue area and the red area in squares. The blue square contains a bacterium which starts almost in focus at approximately 5 microns, and it can be seen how it moves further out of focus as the Z axis is traversed. This was expected from the Diffractio testing previously where the diffraction rings begin again after the focal point is passed. The red square contains another bacterium but has the entire

process from a defocused point through to focused and once more defocused as the Z axis is traversed. The focal point is shown at approximately 77 microns.

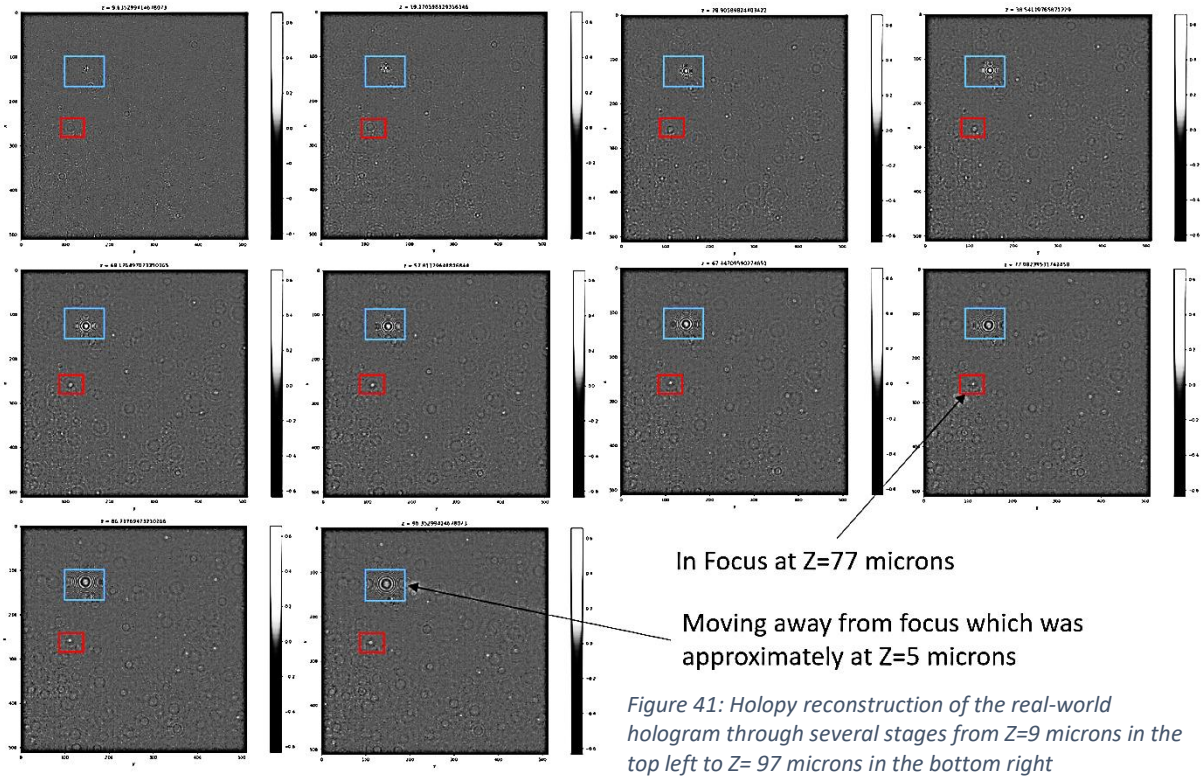


Figure 41: Holopy reconstruction of the real-world hologram through several stages from Z=9 microns in the top left to Z= 97 microns in the bottom right

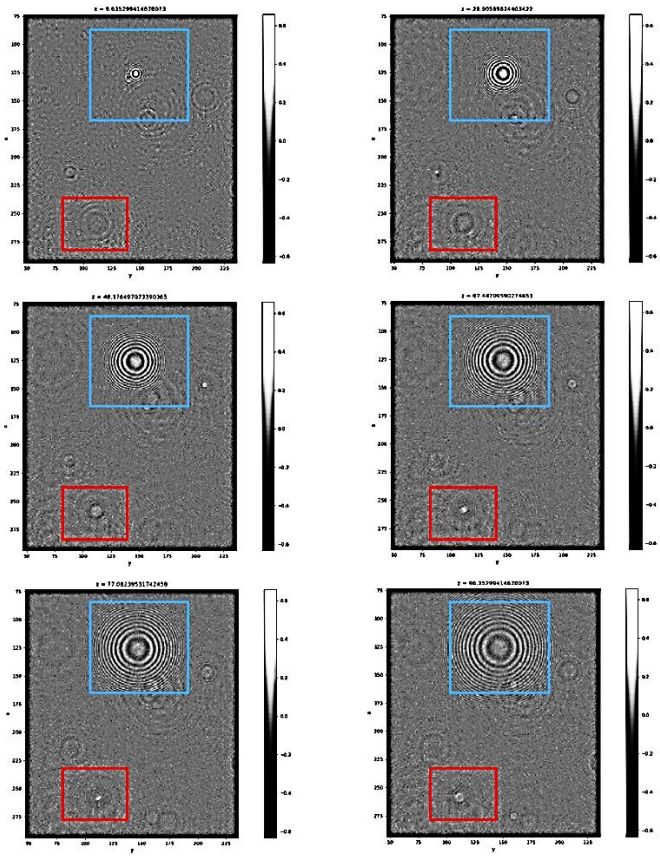


Figure 42: Showing just the pixels of interest containing the blue and red squares mentioned in Fig.41

For clarity Fig.42 shows just the pixels of interest compared to that of Fig.41 which has the full 512x512 pixel array. Larger gaps in the Z value to increase the level of contrast have been implemented in Fig.42 as well to greater benefit the visual aspect of change. The image which is in focus for the red bacterium is the middle-right image at again 77 microns.

It should be noted that the gaps/changes in the Z value have been carefully chosen. The reconstruction image contains complex numbers, it was found that these images can be better viewed if just the imaginary part of the reconstruction is viewed. This contains phase sensitive data and so as progression through the Z dimension occurs the images go in and out of the phase period, the

contrast in the image changes as shown in Fig.43 where the gaps are kept at a usual space ignorant of this matter. To overcome this issue the gaps/changes utilised are integer numbers of the wavelength in the medium producing a constant level of contrast throughout as seen in Fig.42 and Fig.41.

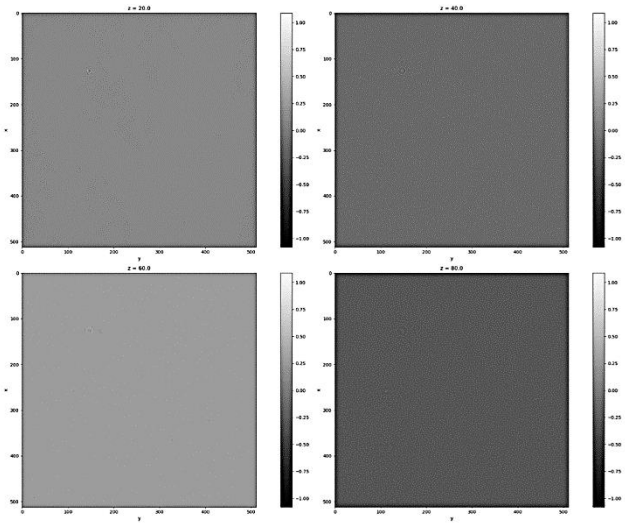


Figure 43: Demonstration of the altering contrast of the output slices due to ignorance of the phase sensitive phenomenon

From looking at the results thus far it can be noted that the accuracy of the coordinates of the bacteria in the three-dimensional space is very limited, especially due to the phase phenomenon aforementioned leaving gaps in the seen data. Due to this a method for more accurately locating the bacterium is required. Holopy provides a 'guessing' system whereby approximate locations of bacteria are provided and the program attempts to locate the said bacteria in a given volume. Once done it outputs its coordinates with sub-pixel accuracy. This can be seen in Fig.44 which is further analysis of the red bacterium previously spoken about. The calculated outputs are:

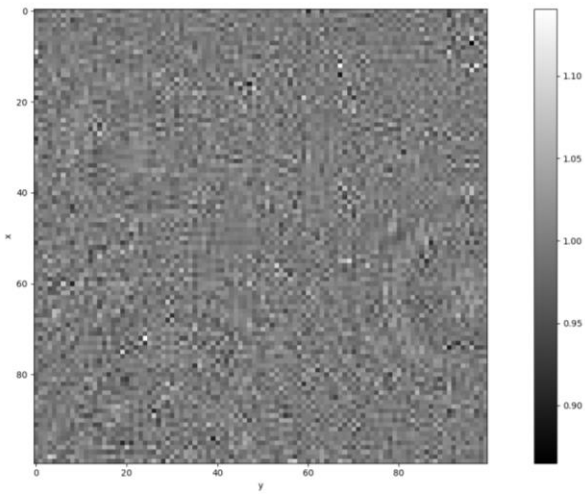


Figure 44: The sub-image of the original 512x512 pixel image showing the 'red bacterium' which is being calculated for its coordinates

Sphere (n = 1.58, r = 1.046788845508687, centre = [258.6120805666572, 110.9196005430164, 77.99825690859299])

The refractive index of the bacterium is given, along with the estimated radius, and the coordinates of the bacterium. The guess of 77 microns turns out to be very close to the actual calculated depth. However, analysis should be conducted into how much error is associated with these calculations as well as how much defect can be implemented into the system before it breaks down.

Out of curiosity and for possible future development, the images of different depth spheres of similar size to that of the red bacterium were calculated to visually see the diffraction interference patterns which are created. Using these template images and the previous optical computer application on ***Page 9 – Masks for SLMs, Types of masks and filters*** for locating templates within images, it may be possible to quickly look up the approximate depth of a bacterium in a full-size image without having to pinpoint it out and search for it (as was done in Fig.44) by comparing its diffraction rings to that of the existing template

images for similar sized bacteria at similar depths. These diffraction rings at increasing depths are shown in Fig.45 below.

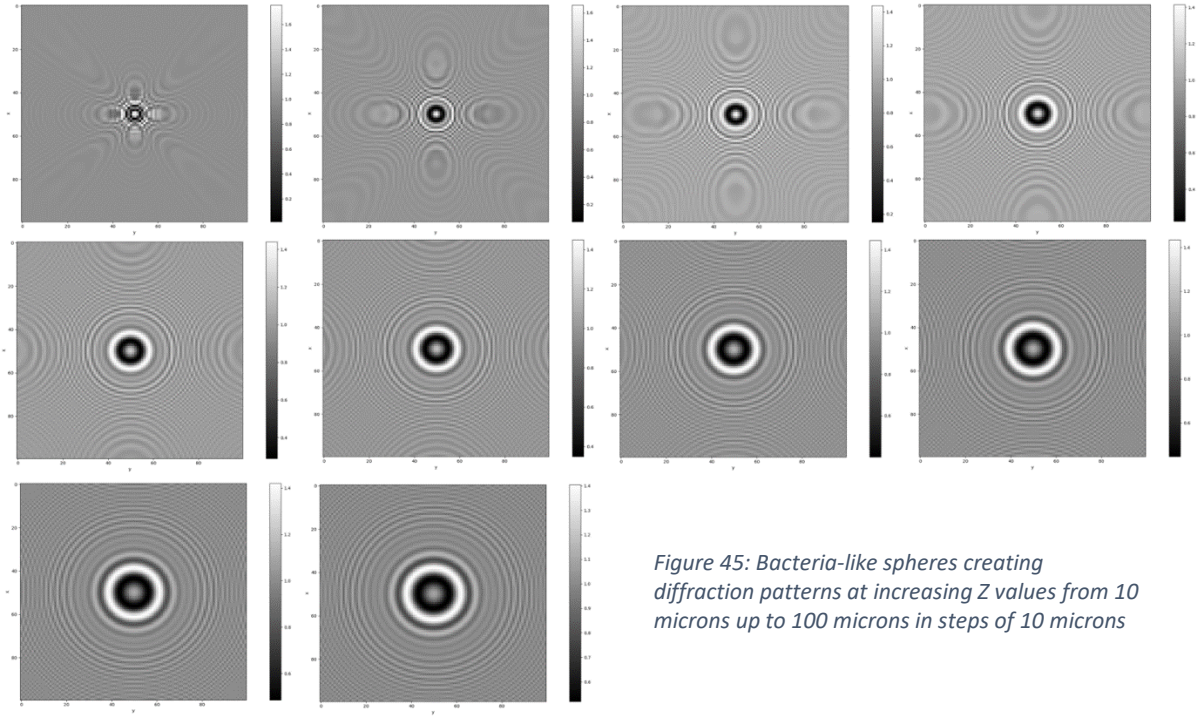


Figure 45: Bacteria-like spheres creating diffraction patterns at increasing Z values from 10 microns up to 100 microns in steps of 10 microns

While it is useful to be able to quickly approximate the depths of a bacterium in a full-image (512x512 pixels) without having to do any deeper analysis, it would also be useful to be able to visualise the entire volume in question completely three-dimensionally and see the approximate locations of all the bacterium within the volume simultaneously and interactively. This is not currently possible on Holopy however with the development that was done in Diffractio using Mayavi, it may be possible to perform this feat. After several failed iterations, the image in Fig.46 shows the first volumetric display of the entire volume being tested. There are however a few difficulties yet to overcome. Namely, the bacteria attempting to be viewed are all relatively small compared to the overall dimensions of the volume and so it may be difficult seeing the highlighted bacteria. Secondly, the display itself is not functioning correctly as of yet. Holopy uses xarray.data types which, albeit a cleaner and more effective way of controlling massive data sets, is more complex than just a simple numpy.ndarray. It is these ndarray types that the Mayavi library utilises to create its three-dimensional volumes. Extracting the data from the xarray into a simple ndarray however comes with its difficulties. It seems as though during the extraction and reshaping of the data into a single,

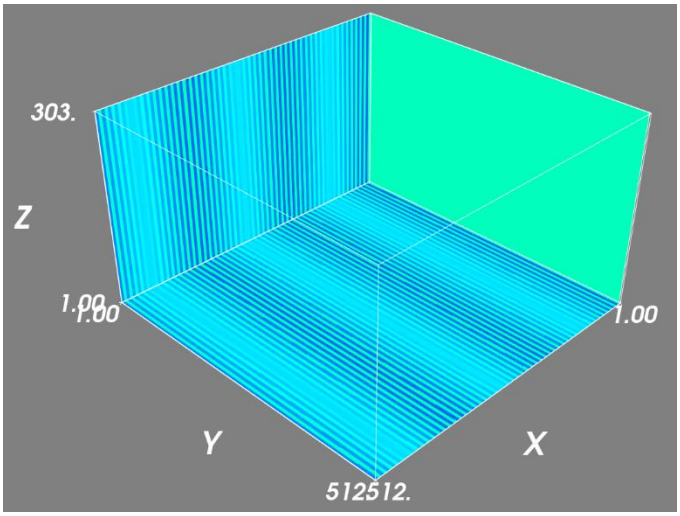


Figure 46: First full 3D display attempt to visualise all the bacteria within the specified volume

large ndarray that the data is being misplaced in some way due to a programming error. This is believed to be the case as the data must be being formatted into the right dimensions as a single, large ndarray in order for Mayavi to have outputted anything. And it is known that the data is acquired in gaps along the Z axis with slices building their way up to create the entire volume. However, upon inspection of the Fig.46 it can be seen the sliced behaviour is in the ZY plane along the X axis not the Z axis. Finally, the lack of any bacteria being found within the volume shows that it is not just a mix up between dimensional arrays but rather the entire data itself has been altered and placed incorrectly resulting in nothing tangible to be seen anywhere. Therefore, it is likely to be a programming error extracting the data from one array into another: literally placing all the right data, just not necessarily in the right order.

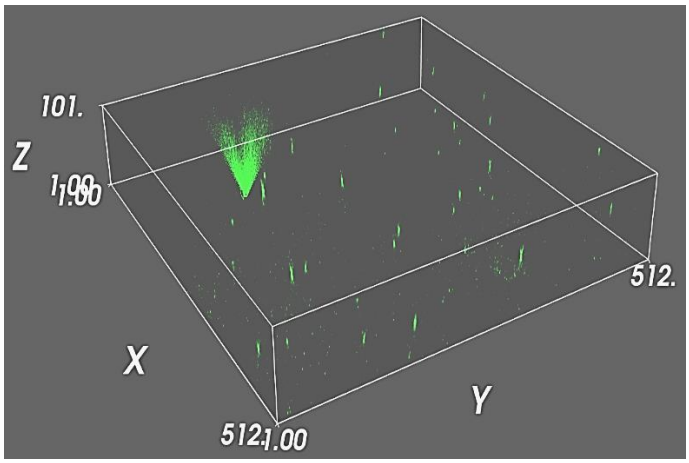


Figure 47: 3-Dimensional reconstruction of bacterial positions

In Fig.47 can be seen the working three-dimensional model. Various alterations were made to accomplish this namely: the 3-channel issue whereby the xarray data was collecting red, blue, and green values for every pixel which led to the 303 high Z axis rather than the 101 as expected. Upon analysis it was seen the red pixel was the only contributor which had any use and so this was the chosen array set to be extracted. Previously the set was extracted into a single dimensional array and reshaped into the correct three-dimensional array however this led to incorrect array placement issues as discussed before. The resolve is to utilise MArray or Masked Array sets which correctly extract the data into the right placement. However, when doing so the axis themselves become mixed. This is easily altered within the Mayavi control panel back to the correct formatting such that X becomes Z, Y becomes X and Z becomes Y as shown in Fig.47. Finally, although the data is correctly interpreted now into Mayavi it is not visible to the human eye clearly and so further alterations (dependent on choice) can be made within the control panel. Specifically contrast settings were altered to 0.965 in the *Volume, Mapper, Final colour level* settings and 0.001 for the *Final Colour Window* for this example.

Furthering this the next steps are giving data points highlights so that they can be more clearly seen as currently the focusing leaves fringe like effects which blur the bacteria's actual position. This is attempted in Fig.48 using contours set to highlight the intense areas which

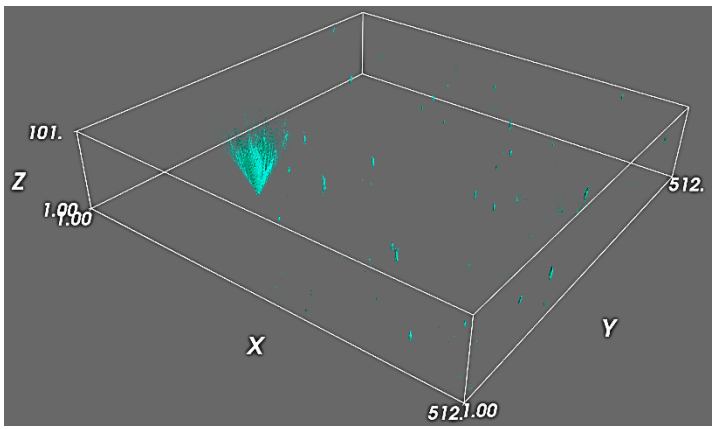


Figure 48: Contour version of 3D display for reconstruction of bacteria

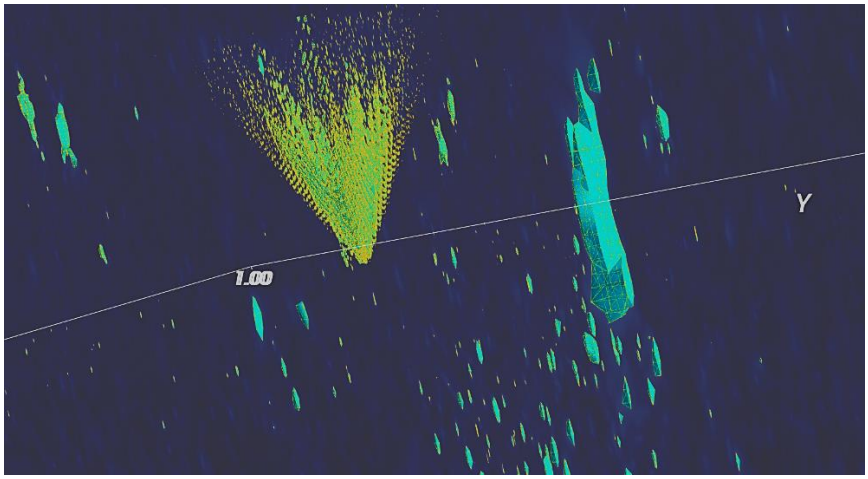


Figure 49: Contour and Volume plots combined to see the overlay of the bacterium position with the intense superposition locations seen in Fig.47

are of interest. However, during this process it was found that when altering contour ranges that some bacteria came in and out of sight and so another programme to combine both the 3D reconstruction of light fields and contours would be useful to see if all the bacteria are being correctly highlighted. This is seen in Fig.49 where the combination of both plots is tuned to highlight the volume in which the bacteria is predicted to be contained. While all of this is visually useful, analysis-wise it is less so as currently it is not possible to select data points within this volume for pinpoint dimensions. A future improvement would be to enable the ability to select visual bacteria plots and have the dimensions of them displayed for analysis. This would also benefit error analysis later on as there is clearly a fringe around the bacteria which should be more rounded. Finding how much percentage of the visual output is fringe would allow even more progression into removing this fringe to leave a clearer volume in which the bacteria actually are.

Look-up Table

26/01/2021

- How to create it

From Fig.45 it was suggested that a look-up table could be created to scan a reference image quickly with an optical computer and attempt to match existing diffraction images from a library to estimate the depths of the bacteria creating said diffraction patterns in the reference image. From Fig.7 on ***Page 9 – Masks for SLMs, Modelling the Discoveries*** it was shown how to multi-match several different images from a library onto a reference image such as the sudoku board. Using the same programming with a few tweaks should allow the library to work for the Holopy reconstruction as well. Visually when comparing the library by eye to the reference image there is a clear issue in that the library images contain much darker patches than that of the reference image. Because of that, none of the library images can be matched by eye to any position in the reference image. The likely reason for this is the simulation itself used to create the library images in Fig.45 creates a singular event. This event is 'pure' in the respect that it has not had outside interference from other diffraction events. The superposition of these outside diffraction events is likely to cause the darker patches to become lighter based on constructive superposition. With a larger set of outside events taking place the overall effect is that all of the diffraction rings appear faded as can be seen in the reference image. In Fig.50 is the first attempt at template matching

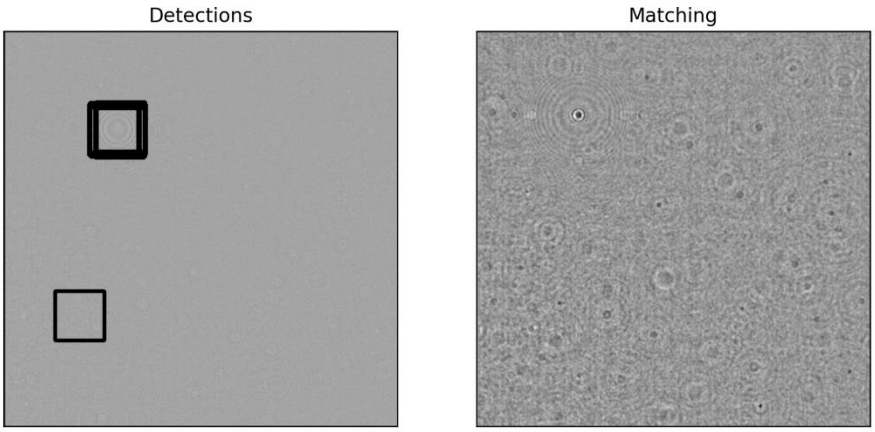


Figure 50: First attempt at diffraction matching with threshold at 0.1 for Z=100 image only

diffraction rings. The threshold value used for matching was set low at just 0.1, and the only image being attempted to be matched was the Z=100 image, refer to Fig.45 to visually see what this looks like. The reason the threshold is set so low is because any higher and the matcher is unable to detect any matches at all, likely due to the previously explained reason. Also the use of only image Z=100 is to create a less complex image before building up on this foundation once it is corrected to work better.

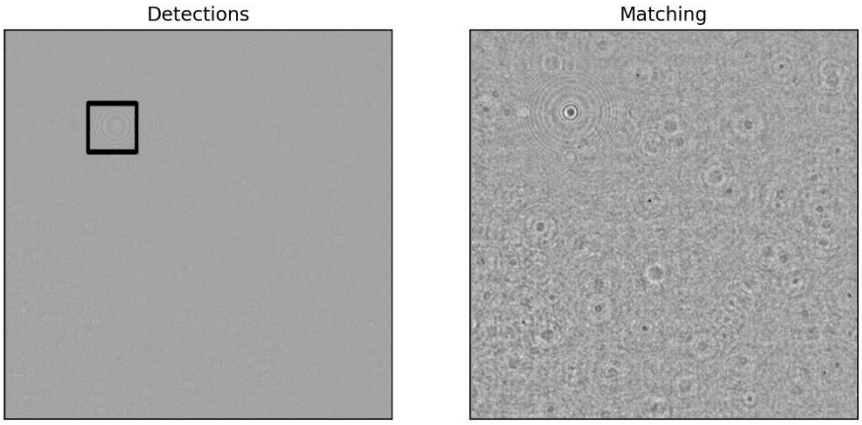


Figure 51: Second attempt at diffraction matching with threshold at 0.1 for Z=90 image only

The addition of the Matching section is due to the extra information necessary to visualise what the simulation is seeing when it calculates for matches. Upon inspection of this it can be seen that the darker spots are supposed to indicate close matches. There are infact quite a few specific dark patches within the matching image. Indicating that there are many close candidates however only the 2 sections show in detections crossed the threshold value. Going onto the next Z=90 image may cast more light on what is happening to lead to such low detections. This is seen in Fig.51 where once again the threshold value is set to the low 0.1. There is just the one detected spot however from the 3D reconstruction of the bacterial volume it is known that there is an anomaly in this spot casting diffraction patterns at every Z level. Looking again specifically at the Matching section it can once again be seen that there are several dark patches but none surpassing the threshold value other than the detected spot again. Comparing the previous Z=100 matching section with this Z=90 matching section as seen in

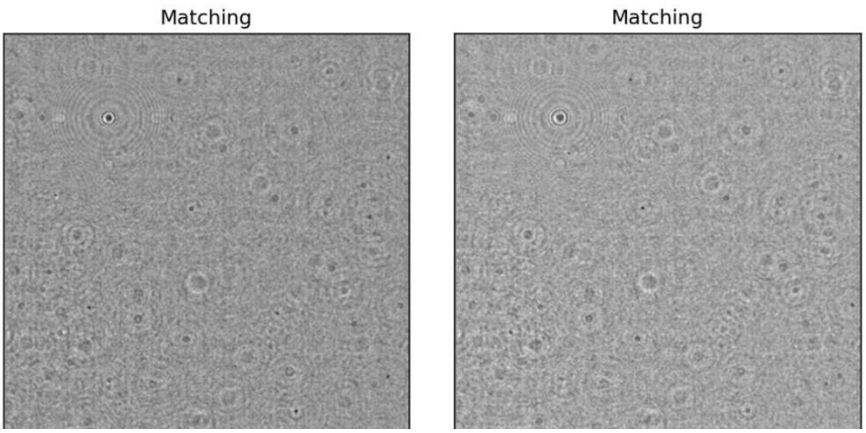


Figure 52: Comparing Matching images with Z=100 on the left and Z=90 on the right

Fig.52 it can be seen that there has been a small step with only slight variation in the two images. This was expected as the Z=90 to Z=100 is only a single 10 micron step. Specifically some spots can be seen to be getting darker and others are getting lighter. This indicates that some bacteria are coming into focus while others are now moving away from focus. In otherwords, there are bacteria which are somewhere between the Z=100 and Z=90 values which would be in focus, and as the 2 stages are attempted to be looked at for bacteria which are in focus, it is struggling due to there being limited bacteria which are actually in focus at exactly the Z=100 or Z=90 levels. This means smaller increments must be taken for the library images than just the current 10 micron steps in order to produce more matches accurately. An alternative is to further decrease the threshold value now not only to account for the outside event interference but also to account for diffraction rings being at close to but not exact increment values of depth.

- Improvements to simulation

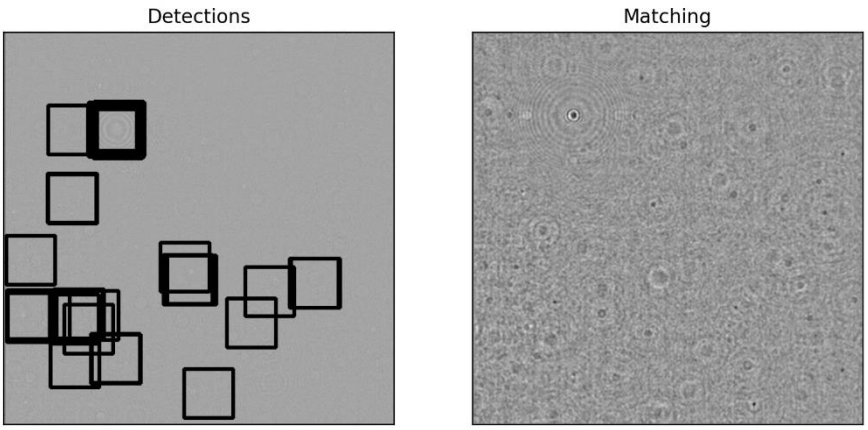


Figure 53: Improved attempt at diffraction matching at threshold of 0.07 for Z=100 image

Decreasing the threshold value lower to 0.07 creates the images in Fig.53. It can be seen that there are now considerably more detections for the Z=100 image compared to before which was the wanted result. However, the accuracy of these detections is to be questioned. There are several overlapping sections indicated by the thicker boarder lines. These overlaps and surrounding zones near the original detection zones might be able to be explained. The reason for this can be found when looking more closely at the actual library images and understanding what is happening in the zones in question. In Fig.54 is a close up view with indicated areas of interest likely leading to the overlapping detections. The Yellow rings denote the diffraction pattern where the bacteria in question actually is to be found. The Blue rings indicate the diffraction patterns which appear visually similar to that of the Yellow section in the horizontal and vertical directions from the central spot. The Red rings indicate similar to the Blue where there are diffraction patterns similar to that of the Yellow section however this time in the diagonal directions from the central spot. These diffraction patterns in the Red and Blue sections occur naturally from the central Yellow section's diffraction rings. However this means when the

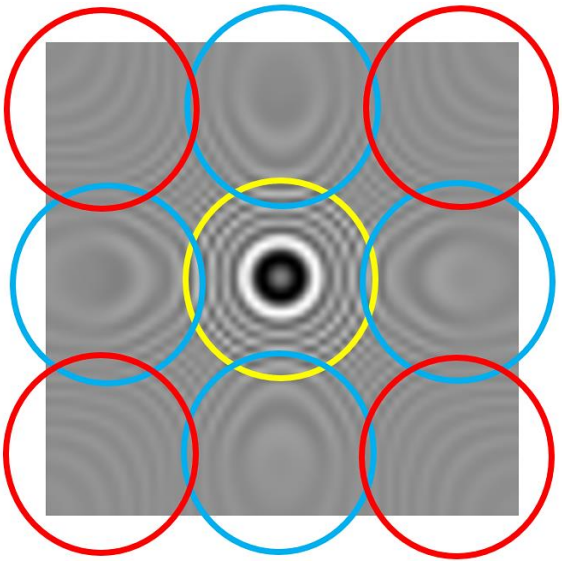


Figure 54: Library image for Z=20 with areas of interest indicated. Yellow, Blue and Red rings denoting different affects in each respective area

matcher scans the reference image for diffraction rings it may pick up this Blue or Red sections and mistake them for the Yellow section especially at such a low threshold value. Comparing back to Fig.53 it can be seen that the zones in question have repeated detections mostly at right angles to the originally detected spot as well as diagonally from the original detected spot as indicated in Fig.55.

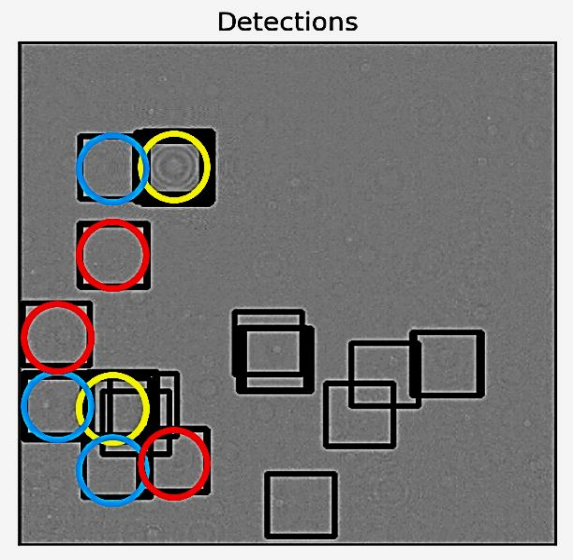


Figure 55: Improved detection image from Fig.53 analysed and labelled with section indicators as described in Fig.54

As can be clearly seen the areas indicated in Yellow are the original detections made in Fig.50 when the threshold was 0.1. Surrounding these areas are the Blue and Red zones with a single Blue and single Red zone around the recognised anomoly in the top left corner, also 2 Blue and 2 Red zones around the second of the 2 originally detected zones. There are remaining zones detected on the right-hand-side of the figure. These zones do not correspond to the previously detected zones and so must be due to diffraction rings which are close to the Z=100 depth but unlikely to be exactly at that depth, either above or below that depth being scanned for. Looking into just how deep these bacteria are into the image using the 3D reconstruction simulation for comparison between the two methods would be useful for error analysis. Furthermore, the aforementioned thicker overlaps found in Fig.53 only occur in the Yellow sections. The reason for this could be again explained by the library images. The library images were created by simulating a sphere of approximate dimensions to that of the previously detected bacterium at different depths and mapping the diffraction patterns caused by this sphere. The library image therefore relies on the idea that the bacteria in the reference image are not only of the approximate dimensions chosen for the sphere but also the idea that the bacteria are spherical. If the bacteria are infact not completely spherical but perhaps more elliptical in nature, it would result in similar diffraction patterns being created in the reference image specifically at the apex at either end of the ellipsoid, seen in Fig.56. These patterns occur in very close proximity to each other and would appear like 2 separate diffraction patterns from 2 separate spheres at the depth in question. Leading to an overlap as seen in Fig.53. While the image in Fig.56 is simply a recreation, it can be seen how at least two spherical diffraction patterns would be detected from an elliptical bacteria creating diffraction patterns similar to that shown here. It should be mentioned the extent of the separation shown in Fig.56 has been exagerated to emphasis the effect in question. The bacteria being analysed are in fact e-coli which do indeed have an ellipsoidal structure.

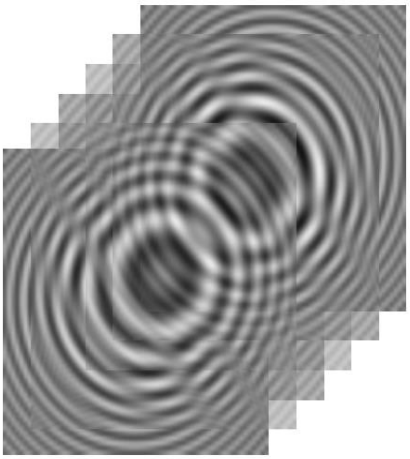


Figure 56: Visual representation of ellipsoid by compiling several spherical diffraction patterns in a diagonal fashion

The second improvement method was to take smaller increment steps. The Look up table creator which generates these library images can be found at "Python\holopy\Coding\Template matching\...Large LUT creator.py". This generator creates images in steps of 1 micron as opposed to 10 microns generated by the Small LUT creator. Using the Single image matcher previously used for

the figures above, for values at Z=90 up to Z=95 and a threshold of 0.08 creates the images shown in Fig.57. As can be seen there is some slight variation from each layer to the next. Specifically notable is once again the 2 main zones are present along with the Red and Blue sections as was mentioned before. However, in images Z=90 up to Z=93 there is the presence of a detection located between the two original zones which disappears as of Z=94 onwards. This is a detection which is now no longer above the threshold value implying if there is a bacteria there then it is likely to be at most Z=93, the minimum is yet to be located. However, in images Z=90 up to Z=92 and Z=95 there appears to be a similar event however in Z=95 the detection reappears and may continue being detected after this. This is contradictory evidence to the latter.

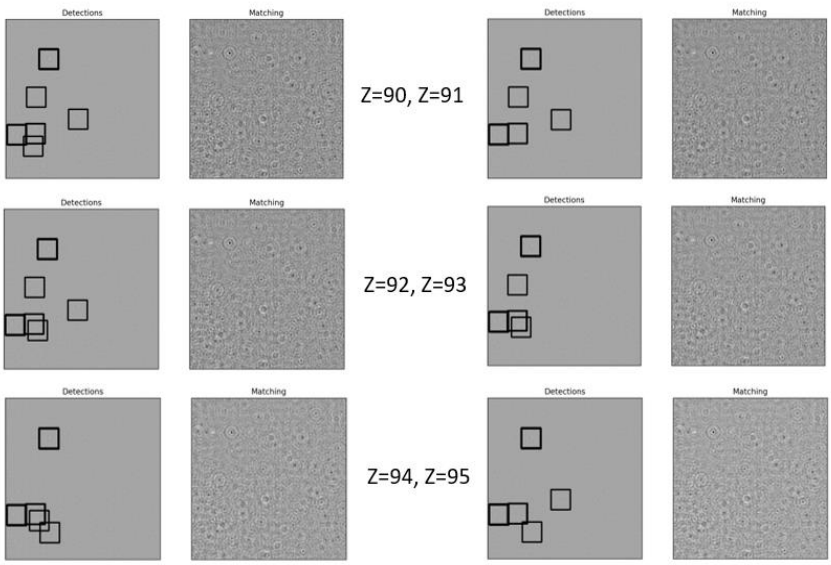


Figure 57: Images of Z=90 up to Z=95 in steps of 1 micron with detections for the single image matcher

- What the results indicate

In order to determine the indication of the changes to the matching simulator it is necessary to compare the collected simulations to that of a previous simulator which is expected to be correct. In so doing error analysis can be carried out leading to a set of conclusions which can be drawn on the changes performed. Taking the images in Fig.57 as the starting point, and visually comparing to the original reference image in order to determine the X and Y coordinates of the supposed detections. Once determined, these coordinates are fed into the fitting value finder at "Python\holopy\Coding\Value finder\fitting.py" as was done on ***Page 34 – Christmas Holiday Work, Holopy Library***. This produces a table of results as shown in Table.4. The points in question are indicated in Fig.58.

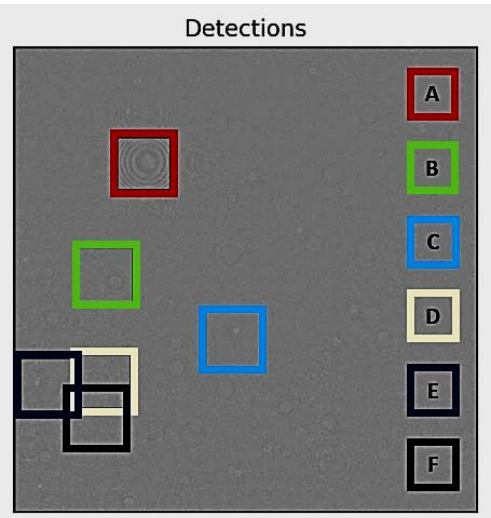


Figure 58: Key indicating the points in Table.4

Point	X - Fitting	Y - Fitting	Z- Fitting	X - Matcher	Y - Matcher	Z - Matcher
A	125	142	5	127	146	93
B	259	111	78	244	97	93
C	313	245	145	318	234	93
D	369	92	48	366	91	93
E	N/A	N/A	N/A	372	52	93
F	N/A	N/A	N/A	396	83	93

Table 4: Fitting results VS Matching results for points indicated, Fig.58

Fringe Patterns

02/02/2021

- Why they matter

Taking a closer look into the repeated patterns formed as shown in Fig.54 may reveal ways to improve the accuracy of the matcher. To take a closer look at these patterns the holopy 2D reconstruction routine is used, found at "Python\holopy\Coding\Template matching\...Small LUT creator.py". Previously the 2D slices created at different Z depths formed a grid of 64x64 pixels, with the main set of diffraction rings centred in the middle of that grid. Using a larger grid with the main set of diffraction rings positioned towards the corner of the grid allow the maximum coverage to look at the surrounding patterns caused.

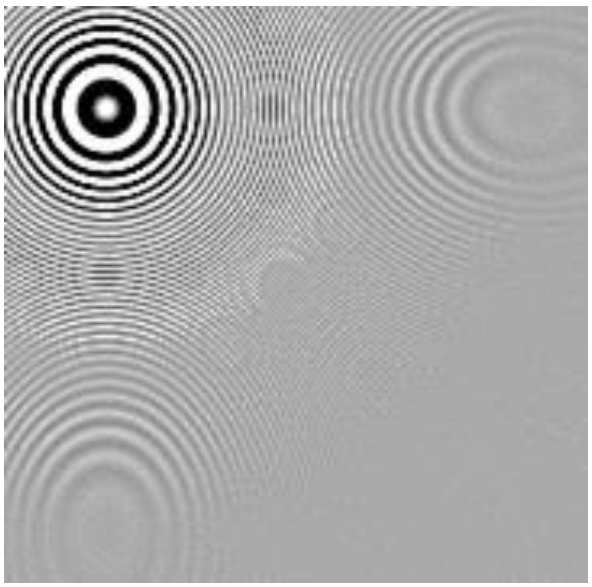


Figure 59: Grid of 512x512 pixels showing specifically the surrounding diffraction ring patterns to a simulated reconstructed of a sphere's hologram. The grid was then focused down to the area of interest only

This is shown in Fig.59, where the grid size has been increased to 512x512 pixels and the main diffraction rings are positioned towards the top left corner. The previously highlighted sections of Yellow, Red, and Blue all still exist in the increased grid reconstruction. However, considerably more smaller diffraction ring patterns are now visible. The rings appear to form in symmetrical formats which occur down specific lines only.

This is more clearly highlighted in Fig.60 where the visible ring's centres have been indicated by blue dots and the specific lines upon which the ring centres occur are indicated by the red lines. It becomes apparent that a pattern of some sort is taking place where the points of interest occur at very precise locations along specific lines which also form at very precise angles to each other. Several other positions for the diffraction ring centres are not labelled in Fig.60 to improve the clarity of the image. There are in fact even more centres towards the lower right section of the figure which have faded out so considerably it takes a fair amount of image contrasting to be able to make out the extra points. These points too lie on the indicated Red lines.

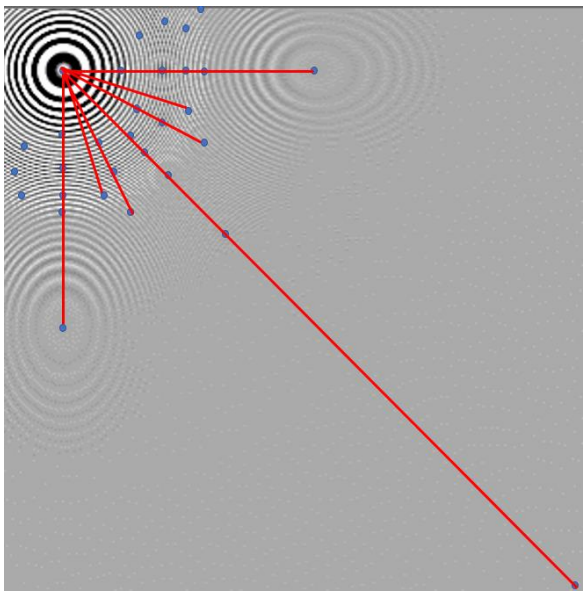


Figure 60: The reconstructed hologram shown in Fig.59 with highlighted points of interest being the Blue dots for the diffraction ring centres and the Red lines for positions where ring centres occur

- What they are

A possible but likely explanation for this is based on several factors. Namely, the object being used to simulate the patterns shown is a perfect sphere, the 2D image created is composed of small pixels, and the image itself is a hologram containing holographic information. The simulation attempts to show a hologram which is a 2D image containing 3D information using pixels. The 3D information is that of a sphere which has been converted into the 2D hologram by using pixels which take on specific brightness levels. Each pixel brightness is a step difference to the neighbouring pixel which means going from top to bottom of the image in Fig.60, each pixel seen is a step up or down to the pixel below it and above it. However this causes an issue, as taking steps in a 2D image containing 3D information means that the 3D image from which the hologram was made must also move in steps of some sort. A sphere does not have steps or technically cubes (given a 3D interpretation of a 2D square pixel is a cube) making up the object. The sides of the sphere are perfectly smooth, or mathematically the dimensions of the cubes making up the sphere must be infinitesimal. This would require infinitesimal pixel dimensions to create a correct interpretation of the 3D sphere in 2D. This means the surrounding patterns seen clearly in the figure above are in fact an illusion due to the screen portraying them consisting of pixels.

The mathematics performed in order to calculate the brightness of each pixel does not occur in step-like fashion or rather it takes place with steps of infinitesimal value. This means despite the appearance, the program showing the simulation is not under the effect of the illusion. This effect only takes place after the simulation passes the results to the screen for us to visualise. Therefore, what we are seeing is not actually the correct calculated result of the simulation, but rather an approximated version with illusion effects.

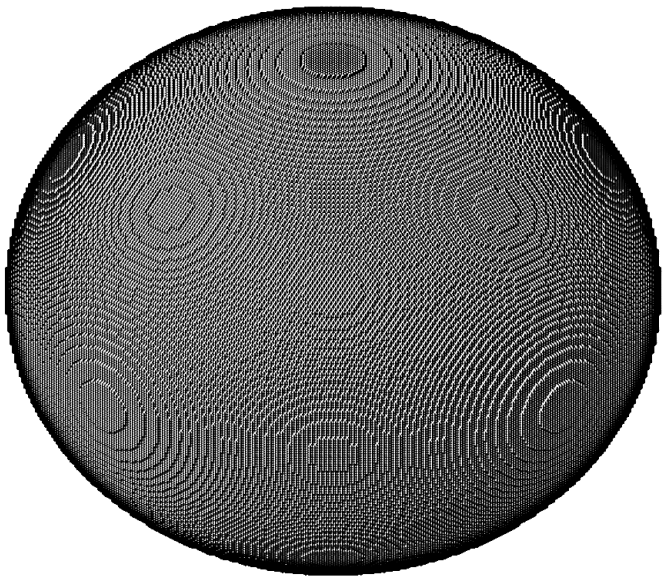


Figure 61: 3D cubic-constituted sphere with a 200 block diameter representing what Fig.60 would roughly result in if reversed through the hologram simulator

A rough example of what the sphere would look like if run through the hologram simulation in reverse based on what we see in Fig.60 would produce a cubic-constituted sphere as seen in Fig.61. This figure was created using a spherical plot generator online using a block diameter of 200 blocks. As can be clearly see once more there are diffraction-like rings formed along the same Red line angles as was previously seen. While the reconstruction appears very similar to a perfect sphere it is in fact only approximately correct with its degree of accuracy proportional to the size of the pixels from which it is

reconstructed. As for why the patterns are detected if they are only illusions as shown in Fig.55, the matching algorithm is likely to be the answer. The algorithm takes pixel values from the reference image to compare to template pixel values. These values are similar to the values the screen uses to illustrate the illusion and so now due to the introduction of the steps once more the output created is affected in a similar way. The pattern which is occurring on the sphere is due to the Zernike polynomial function. And so, the pattern on the hologram image is a variation of this Zernike polynomial function however there is a slight difference between the two.

As is obvious the sphere is Fig.61 can be rotated 90 degrees along any axis around its core and the finishing position will look identical to its starting position. In other words, the rings making up the top of the sphere are identical to those positioned around the sphere half-way down where the axis is. This is the ideal Zernike polynomial pattern. This is not the case in Fig.54 as the blue and red section surrounding rings are not identical to the yellow section main rings which would have been the case if the Zernike pattern were passed down to 2D as well. There is instead a slight stretching effect apparently taking place to the central diffraction rings resulting in similar but different surrounding rings. Due to this slight deviation from identical it may be possible to remove the unwanted surrounding rings using a low pass filter. This filter would remove the higher frequency components of the hologram image which the surrounding ring images may fall under the title of. The results of this are depicted in Fig.62.

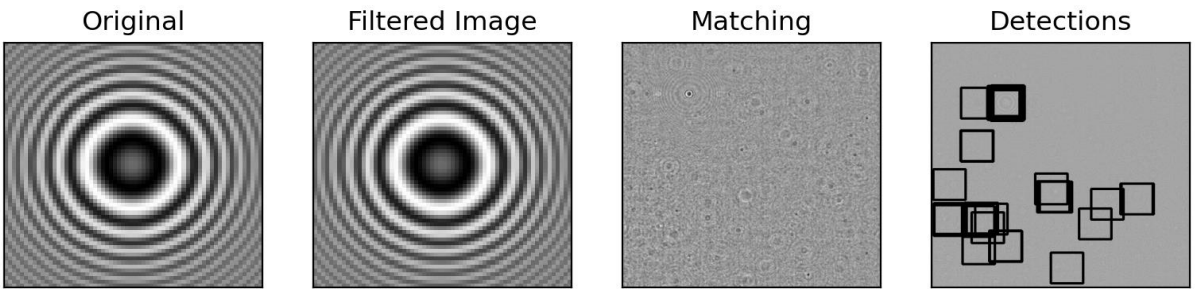


Figure 62: Filter applied to the template before matching is attempted on the diffraction patterns

It can be seen that the filtered image is visibly almost no different to the original template image of the diffraction rings. The matching is done using the TM-CCOEFF method which produces the detections shown when the threshold is set again very low at 0.07. This therefore shows that the filter has been ineffective at improving the detection accuracy. However, for clarity the Original and Filtered images should be scaled up to larger pixel dimensions so as to visibly see the fringe patterns in question and verify if they are still there as expected. This is seen in Fig.63 where the same diffraction pattern has been scaled up to a larger dimension and run through the same low pass filter; again the fringe is still visible.

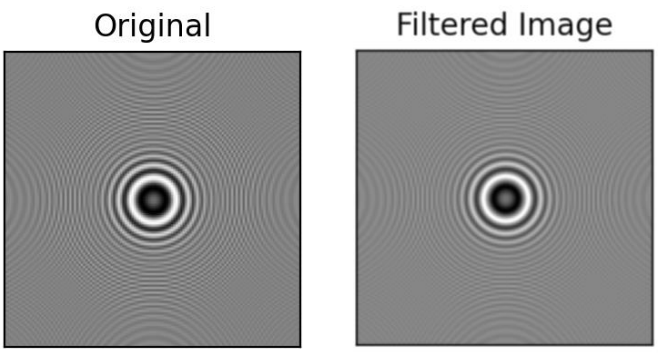


Figure 63: Larger pixel dimensions of diffraction pattern with filter

Because of this it appears as if the fringe patterns are not able to be removed from the image and are naturally supposed to be there as this is the correct representation of the spherical hologram. The low threshold value being used by the matcher may however still be able to be improved by considering the central region being matched. In Fig.62 the template is mostly black around the very central region and then towards the outer region it forms the more gradual diffraction patterns. This black section takes up a fair percentage of the actual template being matched, but this is not visible in the reference image it is trying to match to. This causes the matcher to have difficulty in trying to match the black central region when it is not present. Secondly, if the outer fringe patterns are natural and to be expected then scaling up the image to that in Fig.63 and then attempting to match would remove the detection of so many extra fringe patterns as they are taken as part of the same object in a larger scaled image centred on the actual bacterium. Finally, the central region of the templates is dependent on how the simulation of the diffraction is performed. A differently sized sphere forming the patterns would result in different central regions but very similar outer regions. As well as the refractive index of the sphere in question. Altering this and comparing results could be important to improving the design of the template matcher.

- Re-simulating the diffraction patterns

Considering the points made concerning the sizes of the spheres forming the diffraction patterns in the templates, a range of values should be set out and compared. The simulation is performed in holopy found at "Python\holopy\Coding\Template matching\Programming\...Library creator.py". The results of the different diffraction rings are shown in Fig.64.

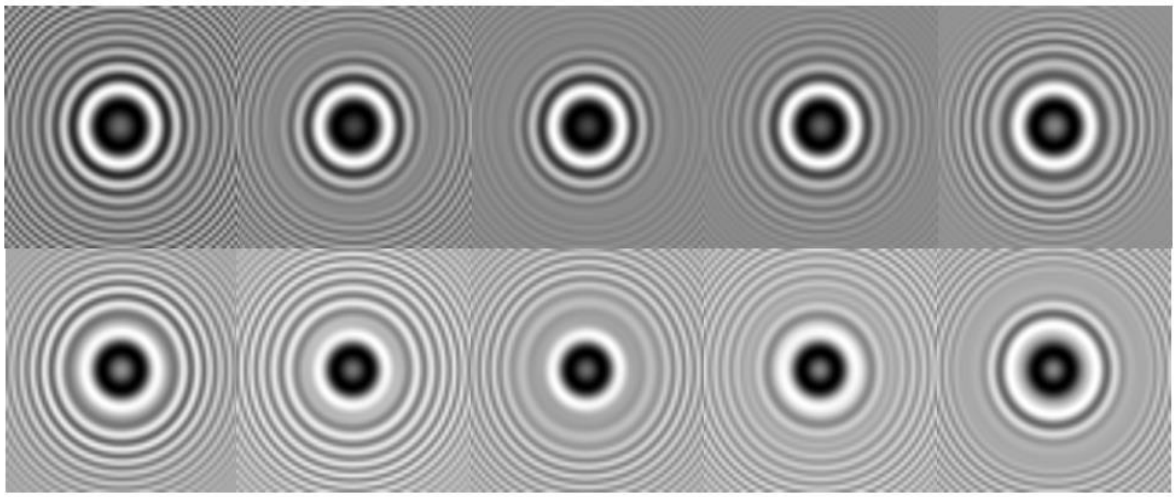


Figure 64: Diffraction patterns simulated for different sphere sizes and refractive indices. This is for n=1 going from left to right with increasing sphere radius starting at 1 in steps of 0.2 reading across down to 2.8

As can be seen the outer regions of the diffraction patterns are very similar to each other in every case. However, the inner regions around the black areas are very different in every case only due to the radius of the sphere simulating it.

Performing the same radii simulations but for a different refractive index may yield different results and so this is tested and shown in Fig.65 for comparison.

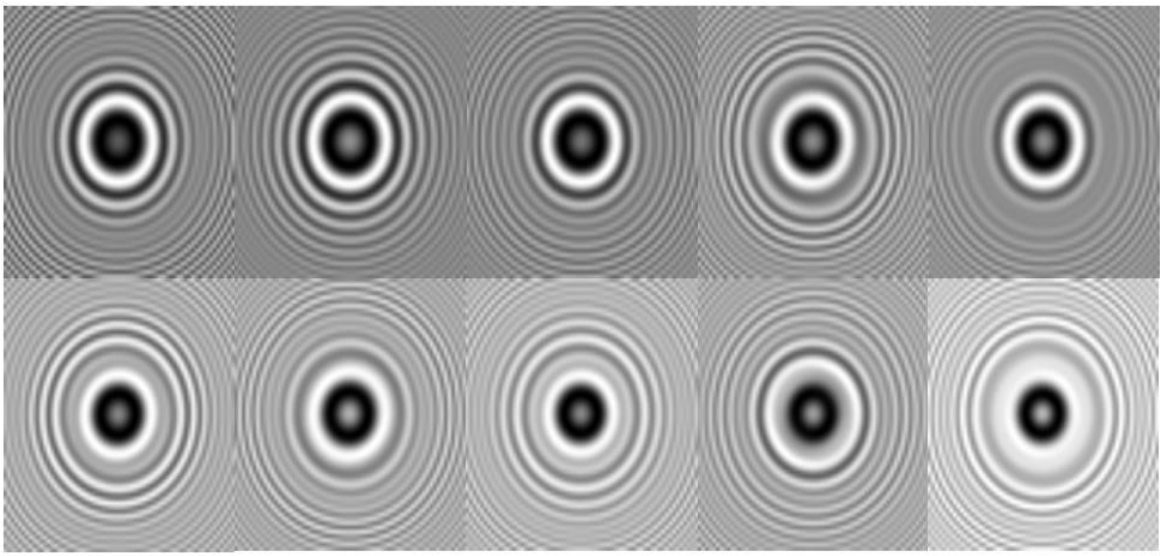


Figure 65: Diffraction patterns simulated for different sphere sizes and refractive indices. This is for n=2 going from left to right with increasing sphere radius starting at 1 in steps of 0.2 reading across down to 2.8, as seen in Fig.64

Again, clearly the inner regions are different in each of the radii being simulated. Because of the clear differences seen here it would be useful to run these images through the template matcher and see what detections are made in each case. Not only seeing the comparison of the radii in each case but also a cross comparison of the refractive indices to see if any visible patterns are taking place. This is seen in Fig.66 where only a chosen few have been taken due to the complexity of each of the detection zones.

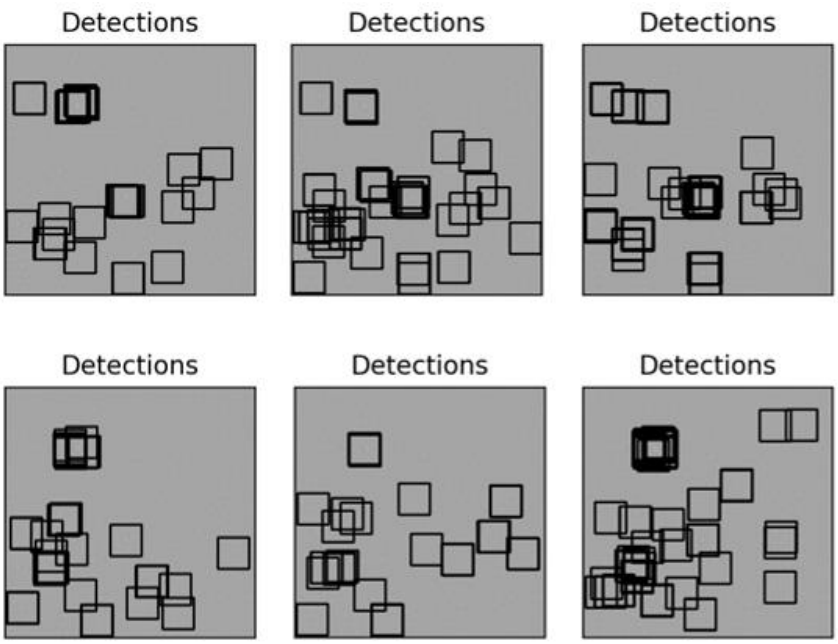


Figure 66: Detection zones found when the different templates are compared to the same reference. The top row is for n=1 and the bottom is for n=1.4. Reading left to right the radii are 1, 1.8 and 2.8 for both the top and bottom rows

The complexity is obvious due to the high number of detections made at the threshold value of 0.06 however this large number of detections makes it easier to spot similarities if any exist. There appears to be clusters of detections in every case. The differences in each case however are significant. Enough to say that there is likely no patterns present but rather each of the detections could be for

bacteria of different sizes and possibly even different refractive indices due to the severity of the differences. If this is correct it suggests a considerably more complex matching system is necessary for picking up all the bacteria in the reference image. It implies that there are bacteria with different dimensions and refractive properties to the extent that different templates are necessary to detect these different sized bacteria even though they may be at the same depth. Considering the idea that the central region is responsible for the dimensions of the bacteria and only the outer rings are responsible for the depths of the bacteria it is necessary to compare two different sets of depths with the same set of radii in each case as the control.

- Comparison of various patterns

Taking the above results into consideration leads to comparing the diffraction patterns shown in Fig.67 and altering them to arrive at Fig.68. These figures are shown below and created by simply covering the central dark rings with white circles.

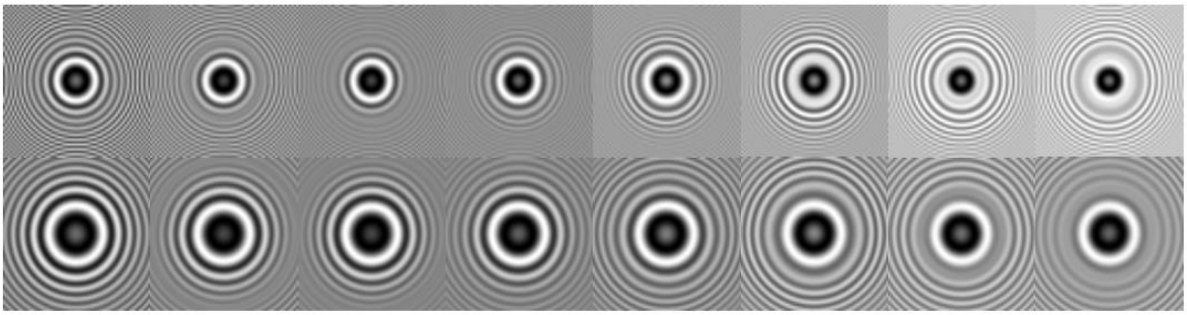


Figure 67: Diffraction patterns arranged in depth order from Z=10 microns to Z=100 microns

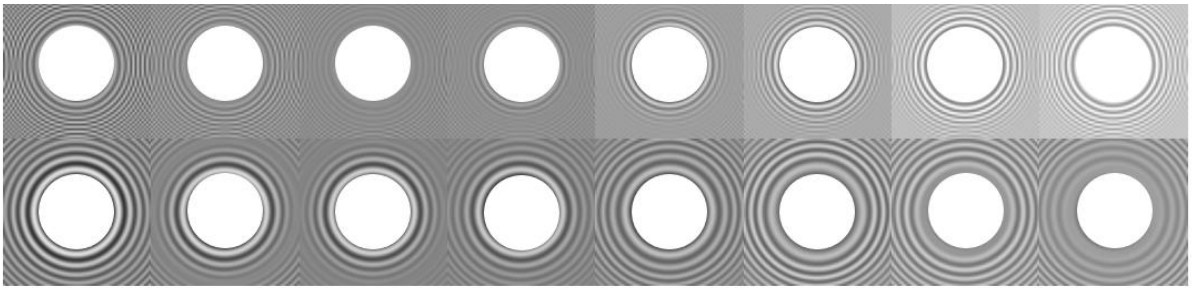


Figure 68: Diffraction rings altered by removing the central dark rings leaving only the outer fringes of the patterns from Fig.67 showing only the outer fringes are needed for matching

By visually looking at Fig.68 it can be seen that the outer fringes of the diffraction patterns alter with the depth of the object. It may be the case that only the outer fringes of the rigs are necessary for matching to determine the object depth without needing the central dark patches. Removing the dark patches seen in Fig.67 means that the matcher will be able to work at a higher threshold value than before improving accuracy. It also suggests that the very outer fringe patterns which have been seen previously are again not necessary for the matcher to locate the depth of the objects in question.

Video Tracking

09/02/2021

- How the video tracker works

The image tracker designed so far takes a library of synthetic images, created using holopy, of diffraction patterns at different depths. It then attempts to scan the provided image for a match in the Fourier domain between the templates and the reference image. Once the template is picked up over a certain provided threshold value the detector is used to map out the square region in which the detection was made and overlay this on the original reference image. Since a video is simply a collection of images taking place in time order, the same design can be used to check each individual frame of the video in the same way.

To do this the video must be first able to be split into individual frames and these frames divided by the background image then saved in order under specific recallable names. Secondly once the frames are saved, they need to be called back to the matcher to be matched with the library of images one at a time to produce detections if any exist. Thirdly the produced overlayed detection images must then be saved to a second location under similar recallable names again in time order. And finally, these resulting detection images must be compiled back into a video to create a stream of detections moving as the particle in question moves.

- What the results show for first tests

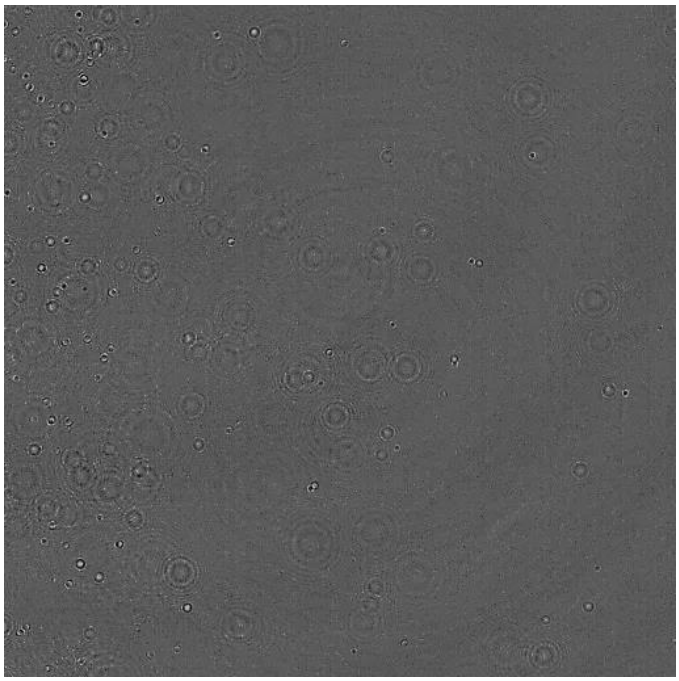


Figure 69: A single DIFF image showing frame 101 of the video divided by the background image leaving just the diffraction patterns

A video of archaea in a volume was provided for testing the detector on. The steps mentioned above were performed in sequence resulting in the file found at "Python\Video Tracker\Programming\.DIFF-frame and vid creator.py". This takes the first 100 frames of the video named .video.avi and creates a set of divided images in the folder DIFF images. From here the images are then called back to the program and compiled back into a video to see the video once the frames have been just divided by the background making the diffraction rings visible again. Specifically, the image arrays have to be multiplied by 255 to improve brightness levels and

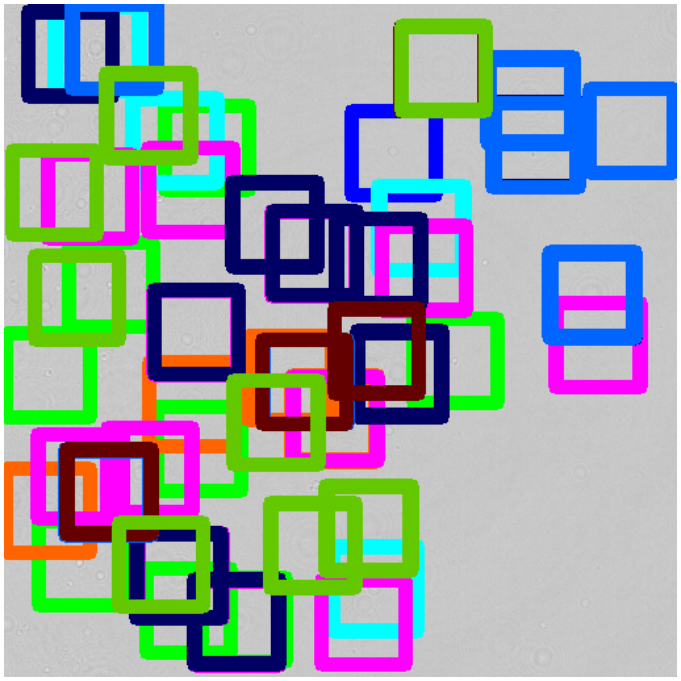


Figure 70: Frame 1 of the video matched using 10 templates

bring the pixel values back up to the maximum so they can be made out in the video. In Fig.69 a single DIFF image can be seen, this is frame number 101 in the video. Several 10s of diffraction rings can be seen mostly congregated towards the left side of the image.

This shows that the frame creator and the complier both work as well as the frame divider to create the images shown. From here the frames must now be matched before being compiled back into video form. This is shown in Fig.70 where 10 templates in steps of 10 microns have been matched with each of the 100 frames created previously. Here frame 1 has been matched and as can be seen there are several matches denoted by the different colours where there is a different depth detected. The image is however very congested and cluttered due to the high number of detections made. Also, the shape of the detections is not best, seen as the rings are circular but the detection is showing a square. Altering this detection shape to a circle with a thinner boarder would better label future detections.

While watching the created video found at "Python\Video Tracker\videos\.VID MATCHER.avi" there are several things seen which could not have been so easily seen going through the frames one by one by hand. There are places where detections being made suddenly disappear and then reappear later on in the video. This shows the detector has failed to pick up the detection which should have obviously been there. There also appears to be overlapping detections however this could be just the detector changing from one depth to the next depth. This needs to be checked and can be done so using a 3D matcher rather than a 2D tracker. It was known before that the size of the library would have to be larger than just 10 images and as discussed it may need to involve several different properties such as n values and r values of the synthetic diffraction rings. Taking the knowledge that the image gets very congested quickly it will be necessary to create a system whereby specific types of templates can be chosen to be matched at a time rather than attempting to match all templates simultaneously as the result will be too complex to analyse.

- Improvements to make to the system

Taking the above results into mind and focusing on the missing detections seen in the video it would be good to create a synthetic video and see how the diffraction rings here move through the video not only in 2D but also in the 3D simulation. From this it may become

apparent how to better improve the system design to account for missing detections. The videos created showing the synthetic created object movement are found at "Python\Video Tracker\videos\synthetic X movement.avi" and "Python\Video Tracker\videos\synthetic X+Z movement.avi". Something unapparent can be seen in these simulations whereby the outer fringe patterns are changing as the object traverses the medium. This suggests that they are indeed created by mirror reflections and may not be directly due to a 3D representation. In order to further determine if this is the case the deeper programming of holopy must be looked into to find how each of the pixel values are calculated. From that the understanding of how the images are formed will show what is causing the outer fringe patterns. This is necessary to improve the system as can clearly be seen they are altering significantly what the expected output would be in the video.

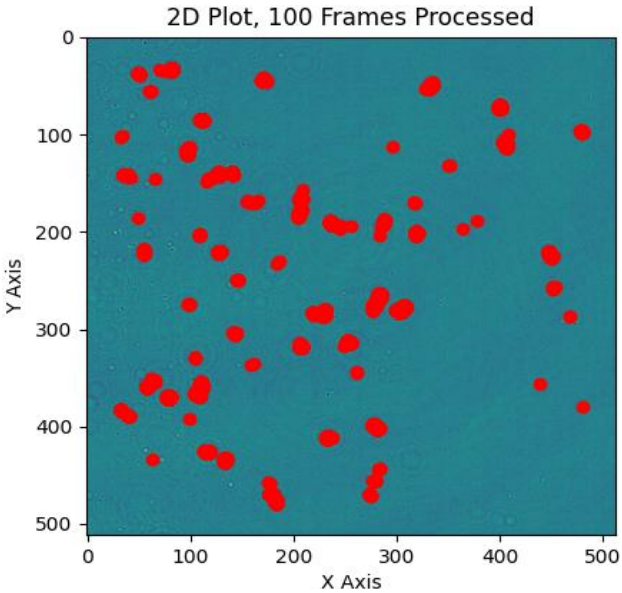


Figure 71: 2D plot of 10 templates matched over 100 frames

A considerably larger template library has to be constructed taking Z depths from 1 to 100 microns in each case of an n and r value ranging from 1 to 3 in steps of 0.2 each. This totals approximately 10,000 images in the library to be compared to the video frames which should create a considerably better set of detections. Finally, the program needs to be improved in some way to allow changeable choices to be made for which set of templates and any other variables which may want to be altered.

- Results of improvements taken to the system

The detector has been altered to show clearer detections of what is happening in each frame as discussed previously. Furthermore, due to this being just a 2D projection of detections the 3D colour scheme has been removed just to show 2D traversal of the objects in question. Finally, the resulting output of the program found at "Python\Video Tracker\Programming\2D plot Matcher.py" is actually a plot rather than a video showing the movement of the object in the medium. This better shows overall movement taken by the object in question. This is done by taking the detection made, locating the centre coordinates of this detection for every template matched over a threshold value and then putting these coordinates into an array which is then forecast to a scatter plot overlayed on the original starting frame for better visualisation purposes. One such simulated output is shown in Fig.71. The larger looking red patches seen are due to the bacteria moving around in that area. Some places it can be seen there is very limited movement. This could be due to the bacteria not moving or again due to the lack of correct detections. Further template matching needs to be performed yet to improve this.

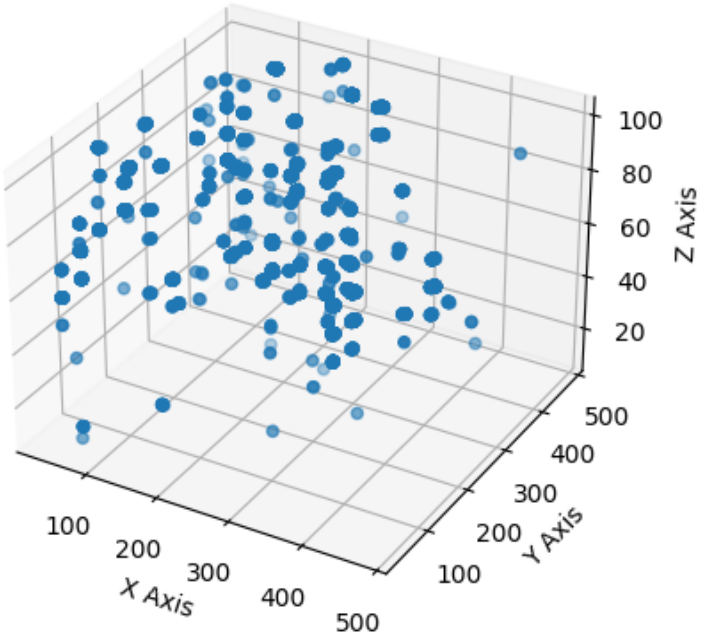


Figure 72: 3D plot of 100 frames from video matched against 10 templates

In Fig.72 the same set of parameters are entered however this time into a 3D tracking system. This is done by creating another array mapping the templates which were matched with their respective coordinates in 2D. This creates a 3D set of coordinates which can be plotted as seen. Upon inspection of this plot however it can be determined that several templates are matching on the same bacteria resulting in lines along the Z axis. Increasing the threshold value may remove the number of multiple incorrect detections made and improve accuracy.

These plots however only have 1 alterable input currently which is the number of frames to be processed from the original video. The files found at "Python\Video Tracker\Programming\improved 2D plot Matcher.py" and "Python\Video Tracker\Programming\improved 3D plot Matcher.py" have been redesigned to enable considerably more inputs to be altered as the program runs. A fully interactive data generator. Not only is it fully interactive but it also takes data from the considerably large template library generated using holopy and the file "Python\Video Tracker\Programming\template creator.py" edited each time to generate the correct set of synthetic images. An example of how much can be altered in the improved 2D program is shown in Fig.73.

```
al\improved 2D plot Matcher.py'
Number of frames to process? 100
Template n value to use? 1
Template r value to use (, for .)?  1,6

File chosen: n=1, r=1,6

Threshold value to use? 0.12
Templates in steps of? 5
```

Figure 73: Example of improved 2D plot matcher showing the level of alterable inputs

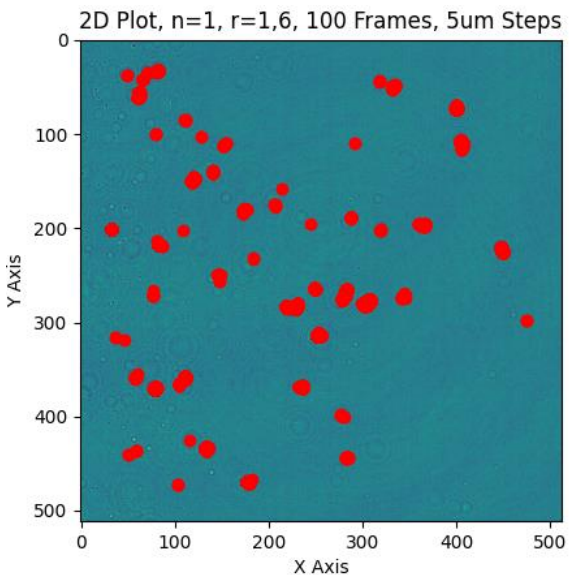


Figure 74: 2D plot generated using the inputs shown in Fig.73

This produces the plot shown in Fig.74, there is very little movement seen once again even with the larger threshold value. From the overlay image of the first frame, it can still be seen there are a lot of diffraction patterns not being picked up at all mostly seen again towards the left side of the image. It is likely these patterns are actually created by a different sized bacterium and so when using templates in steps of 5 microns which are

synthetically formed using n=1 and r=1.6 only bacteria which fall in this category will be highlighted.

In Fig.75 a typical resulting matched detector image is shown, and it can be more clearly seen that the detections are less cluttered and better highlighted using the thinner circular method previously mentioned. There are areas where several overlaps can be seen. These are unwanted and are likely to be overlaps of different templates incorrectly detecting the ring at their depth. These incorrect overlaps lead to incorrect data plotting and inaccuracy. However, this is only a 2D plot and so different depths being located in this scenario are not an issue as depth is not being recorded in 2D. For the 3D improved results see below.

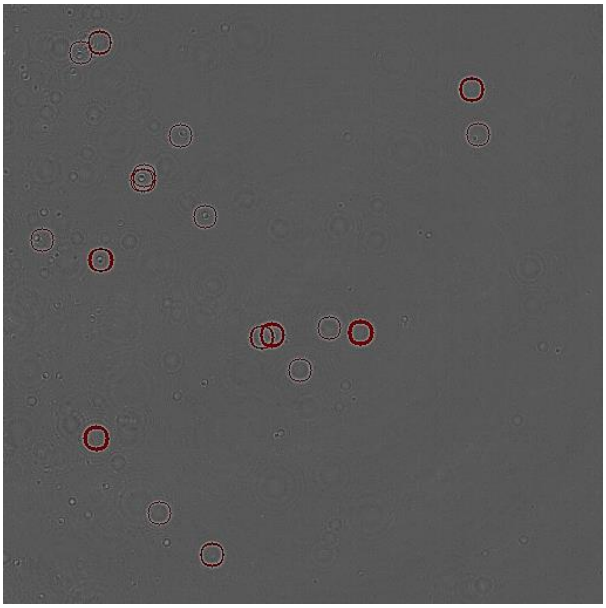


Figure 75: Matched detection shown for improved detector

In Fig.76 the improved 3D for the same parameters as in Fig.75 is shown. It is clear that there are lines forming parallel to the Z axis which is due to incorrect template matches at wrong depths. Increasing the threshold value further may remove these lines, also increasing the template steps may also reduce the lines being seen. Larger steps means that the rings being seen will have to be forced into an average Z depth which while removing rings and increasing clarity, reduces accuracy in turn.

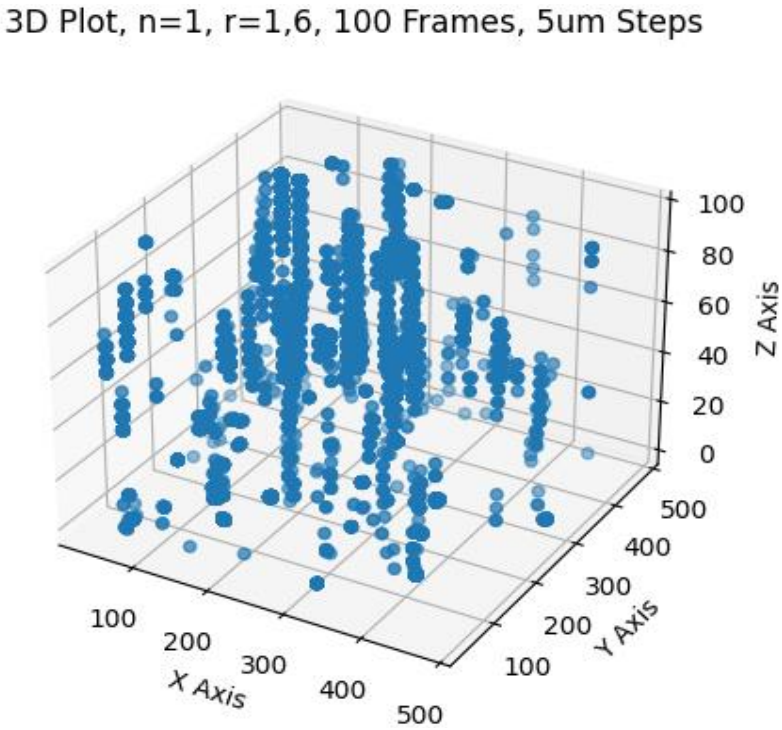


Figure 76: Improved 3D plot showing the data created from the inputs in Fig.73

Another possible method of improving the system further is to take the best matches in every case rather than just the entire collection of matches. While the threshold value allows the matcher to determine what is an acceptable level of match, it does not allow the matcher to decide which template matches best for a particular object. For example, in Fig.76 on the far left there is a set of 4 detections for different Z depth templates. If instead the best match of these was chosen rather than plotting all of them then the plot will only

show the best of the best matches. An issue with this is these 4 detections may not have been taken on the same frames. Some may have been taken on later frames meaning the matcher will struggle to choose the best of the best without some Artificial Intelligence input deciphering this.

It should also be noted in order to check the functioning of the improved matchers a tester program had to be constructed due to the complexity of the improved matchers processing several thousands of bits of data at a time. This tester program can be found at "Python\Video Tracking\Programming\tester .py" and its main purpose is to show that the program is indeed checking every frame with the correct number of templates each time and not just the first frame with the correct number of templates and then altering this later down the processing line.

- Discussion of areas yet to improve

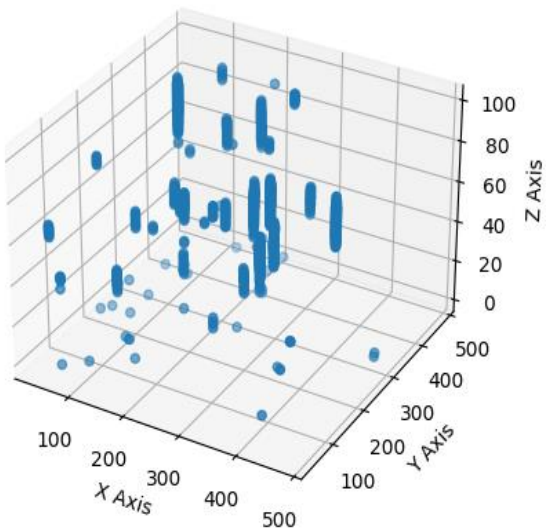


Figure 77: Improved 3D plot showing a single frame9

Taking a single frame using the improved matcher with the highest number templates to be matched produces the image in Fig.77. Using this as a comparison against the original holopy 3D reconstruction system to see how well the two methods are correlated.

Comparing both of the created plots, the same line features along the Z axis can actually be seen. In holopy this is because it is taking a threshold value for intensity for which to map. This is a different reason to that explained for Fig.77 however the same effect is occurring. The 2D placement along the X and Y axes are however very accurate. Taking the array which produces the plot in Fig.77 and analysing it for the range of Z values to determine the error in the Z dimension could be useful to determine when an improvement in accuracy occurs.

The Hologram Analysis System (THAS) 16/02/2021

- What THAS is designed to do

The purpose of THAS is to enable the detailed analysis of holographic data in various ways looked into thus far. The system should be able to take any holographic-microscopic video input of objects and once provided with the background image (used to create clearer frames for analysis), the production of several sets of data should be possible. The sets of data produced will be produced based on specific custom settings put into the system.

There should also eventually be an Autorun setting whereby the system performs self-analysis of the data provided and chooses its own settings to perform data production. These settings would be designed to maximise wanted effects on the data in order to create more accurate data.

From looking back at the recent endeavours into holographic analysis and parameters required, a list of parameters necessary for THAS can be produced. This includes; a background image, a video, the number of frames in the video to analyse, the steps between the frames being processed (necessary for slow moving objects to greater emphasize movement in three dimensional plots), the refractive index of the synthetic templates being used by the matching process, the synthetic spherical radius of the templates being used by the matching process, the steps between the templates (used to speed up data production for a basic overview of the situation if necessary), the threshold value being used by the matching process. To add to these 2 further parameters of contrast and sharpness can be used whereby the parameters alter their respective properties for the video frames themselves in an attempt to better improve matching results, or at least analyse the effect of altering such parameters.

The plots being produced, based on previous results looked into, should be; a two dimensional plot of the X axis against the Y axis for attempting to 2D-track the bacteria, a three dimensional plot of the X, Y and Z axis for attempting to 3D-track the bacteria along with the addition of a colour bar to highlight the sections in the 3D plot where the matching value is at its highest (this is to better show the 3D tracking loci of the bacteria), a plot of the Z axis against the Matching value (this is a new plot in order to monitor for associations between the depth and how well the templates respective to those depths are matching), a plot of SSIM against the Z axis (another new plot leading back to the very first stages of analysis into images at the start of this project. The attempt is to monitor again the templates themselves against the depths), a plot of MSE against the Z axis (another plot attempting to analyse the templates relative to the actual images for those depths again), and finally a plot of MSE against the Matching value (to monitor how these values alter in proportion to each other).

The hopes of this system are that once finished, any example video could be run in the system either under specific parameters or under Autorun settings. Under Custom settings the video is processed and analysed for the respective plots described above. And in Autorun settings the video is checked against several operations leading to specific settings to be chosen to process the video under. Once processed the same plots are produced along with all the useful data outputs in text for clarity.

- Resulting data produced

The previous sets of extensive python coding for every individual set of plots and tests were compiled into a single python script. Then utilising the PyQt library, the input parameters were altered from a terminal input seen previously in Fig.73 into the more user-friendly

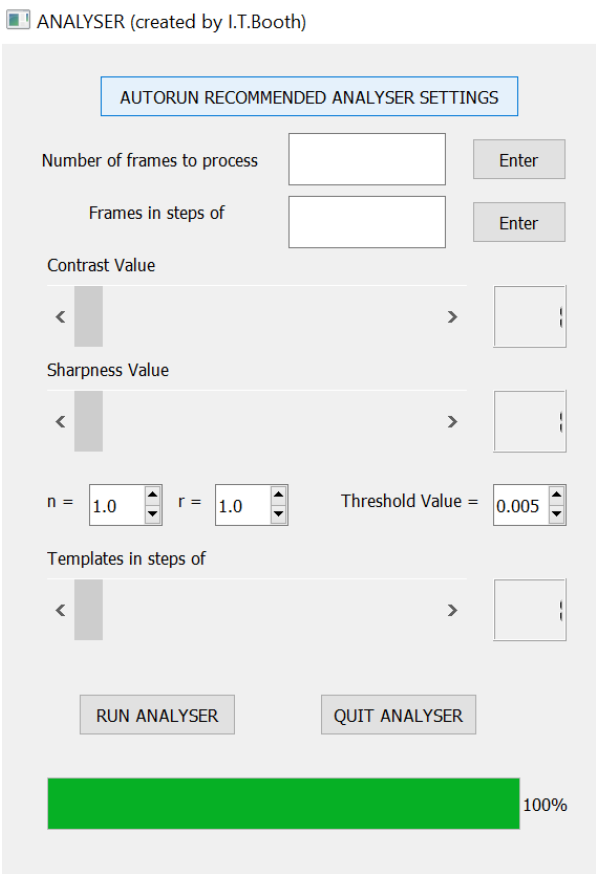


Figure 78: The Hologram Analysis System user-friendly inputs

input parameters seen in Fig.78. Various input-specific settings were chosen, such as the contrast value increases in steps of 1 between a minimum of 1 and maximum of 20 based on testing to decide what appropriate limits would be. Likewise, with the sharpness settings. The n and r values both start at 1.0 and increase in steps of 0.2 as these are the steps taken when producing the synthetic template library currently consisting of several thousand images for referencing as each set of n and r values has 200 templates associated with them. The threshold value has a minimum of 0.05 and currently a maximum of 0.2 in steps of 0.05 however this may need to be increased or at least altered upon future testing and producing better matching circumstances. The template steps start at 1 and increase in steps of 1 up to a maximum of 20. The 'RUN ANALYSER' button takes the custom settings entered and attempts to run analysis on the video and background provided. These settings currently are not needed to be altered as only 1 video is available for analysis however in future use these input choices would be available. The 'QUIT ANALYSER' brings up a message box to check if this is the wanted action, upon confirmation the application is closed. Finally, the 'Number of frames to process' and the 'Frames in steps of' inputs take integer values upon pressing the 'Enter' button. Currently the Autorun settings are set to produce a fast loading set of results for ensuring the program is functioning as anticipated as considerable testing to get this system working was necessary and so saving time was of the utmost importance. There is also a loading bar present which was useful to visualise what the system is doing in real-time. Along with this the text output produced with the graphs, just off the right side of the image in Fig.78, shows real time estimates of time remaining for the analyser to complete its processes.

Following the completion of the processing, the text output produces various numerical data as seen in Fig.79 along with the 6 separate plots accessable in their respective tabs including the 3D Plot which is updated from th eprevious attempt with the Match Level now represented with a colour bar. As well as the entire 3D plot being interactive whereby the plot can be moved around like a cube in a hand by swiping over it when left clicked on it. Including a zoom function where right clicking and swiping up or down will zoom in or out respectively. In Fig.79 the 3D Plot seen is the outcome of the 'Autorun' setting currently set to create a fast loading set of data for testing purposes. A more detailed 3D Plot is seen in Fig.80 where considerably more frames have been taken into account, across a much larger

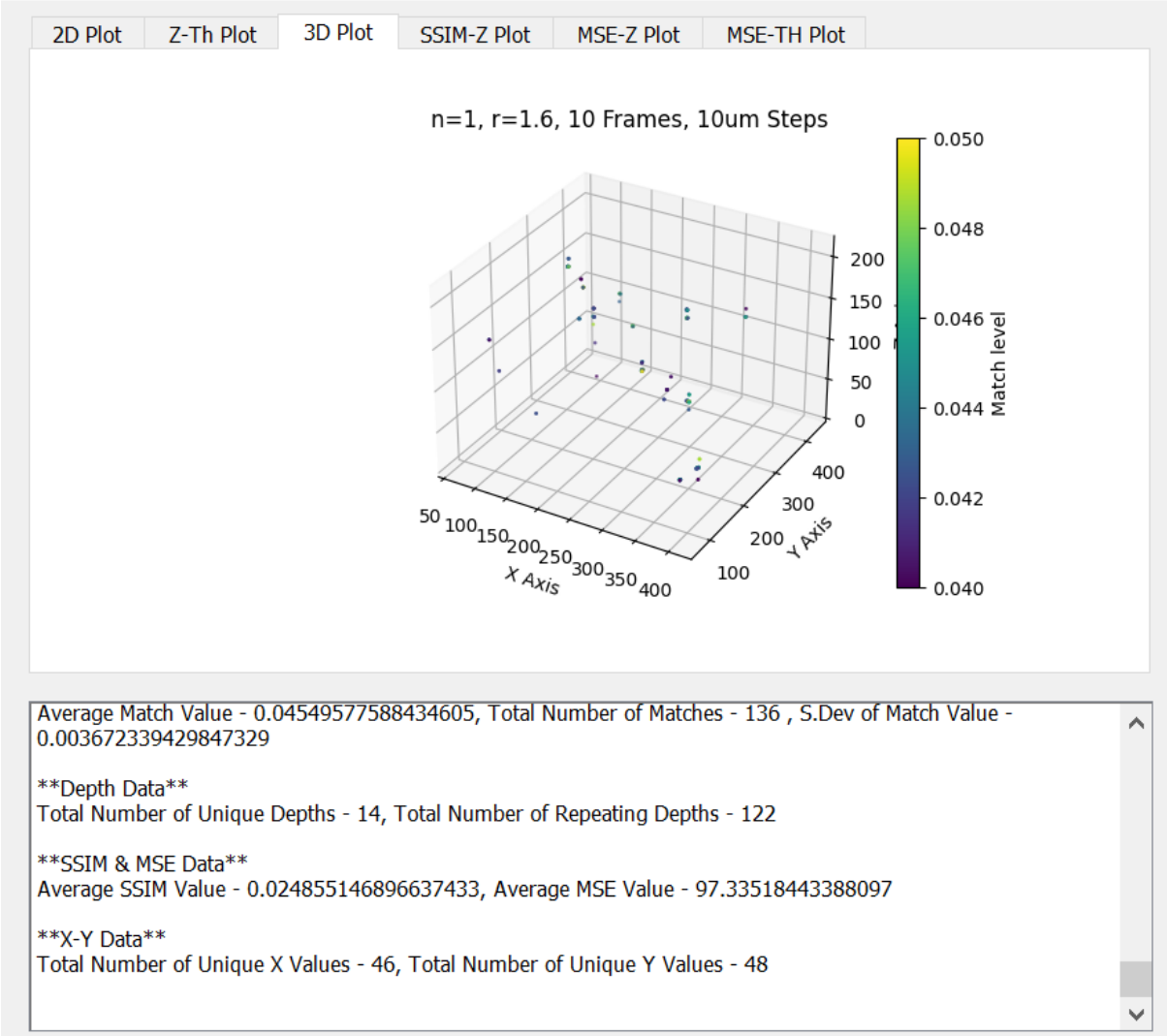


Figure 79: Autorun setting outputs showing the 3D plot for testing, the text outputs and the various accessible plots

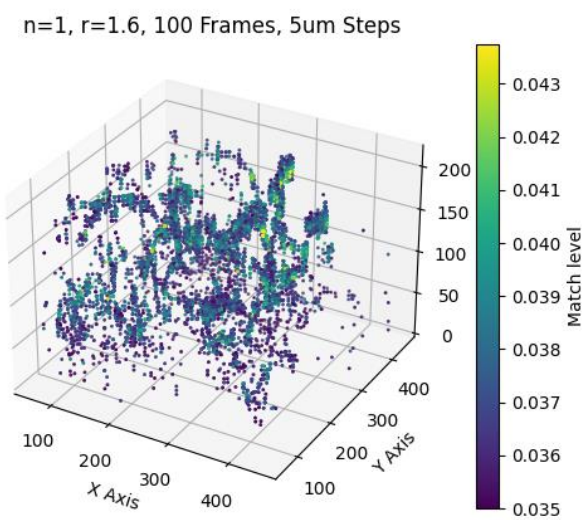


Figure 80: 3D Plot for a considerably larger set of data

range compared to previously where all the frames were taken in order. In Fig.80 the entire 10,000 frames of the sample video have been looked at in steps of 100 to emphasize movement in the archaea.

While the plot looks very congested and unusable, upon interacting with the plot and adjusting to other positions, various clearer patterns of movement can be made out. There is still improvement yet to be made regarding removing the cluttering singular results which are not moving and are likely different dimensionally sized archaea being picked up incorrectly as they happen to pass the low threshold value. The threshold value itself has been further reduced due to the increase in the template dimensions from 64-pixel widths to 128-pixel widths.

This was necessary to show the larger diffraction patterns exhibited by depths of over 100 up to 200 microns. Further analysis into the depth's verses matching values should reveal information regarding this alteration and its affects.

In Fig.81 the Z-Th plot can be seen, which is the plot of the aforementioned depth verses matching value. The average matching value is shown to be approximately 0.0456 in this plot. From this it can be seen that the lower Z depths up to approximately 100 pixels are all below that average matching value. A clear increase in matching value can be seen from the lower depths up to the higher depths where upon approximately 125 pixels the average per depth drops. Further such plots with various settings will need to be performed for comparison to better understand this latter drop however it is clear the lower depths which have had an increase in pixel dimensions have suffered with their matching values. Returning the pixel dimensions to their previous sizes for the lower depths could return higher matching values for the lower depths once more.

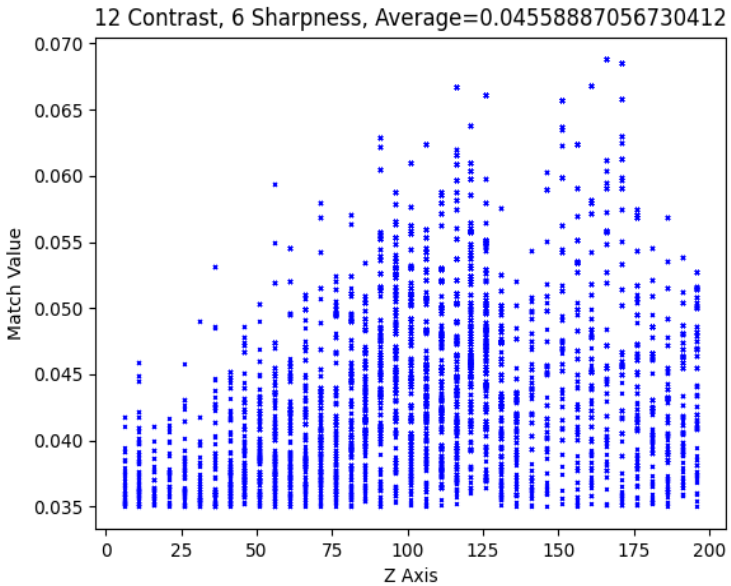


Figure 81: Z-Th Plot showing the matching values against their depths

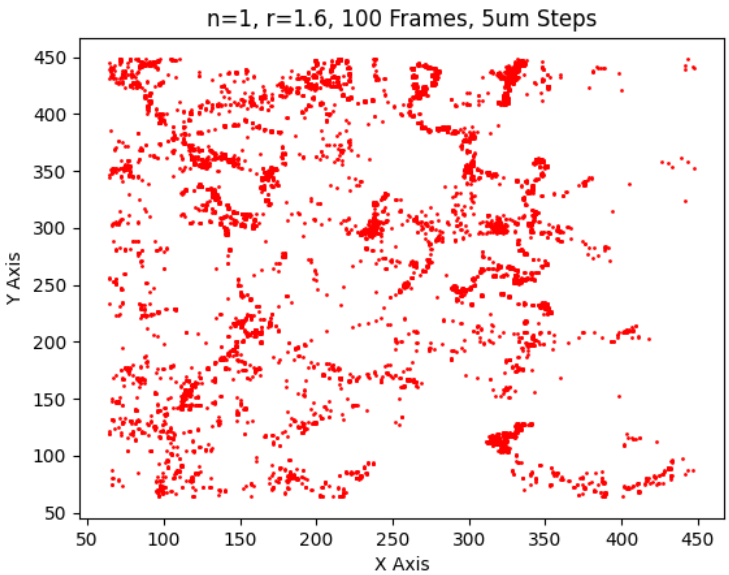


Figure 82: 2D Plot of data showing paths of archaea have taken over the entire 10,000 frame sample video in steps of 100 frames at a time

The 2D Plot is shown in Fig.82 where clearer paths of movement can be made out now due to the addition of the frame skipper system. Particularly around the (250, 400) coordinates where a significant loop can be seen to take place. This same loop can be seen in the 3D Plot once the plot has been interacted with to give a clearer view of it happening. Again, there are clear places where only a few detections have been made which are likely to be other dimensionally sized archaea incorrectly detected here. The removal of these incorrect detections while it would be useful for clarity it would be very difficult to do with the current programming as technically no mistakes have taken place in detecting the archaea. Rather the number of detections is undesirable and programming to

detect this would require a deep learning software to detect between different sets of frames and then remove the unwanted results which do not correspond to other positions detected in the other frames. An alternative method would be to improve the method of detection in the hope these stray results are minimised rather than removed.

Returning to the very first few tests taken place during this project, the Structural Similarity Index (SSIM) was one of them. This value compares two images of the same dimensions and produces a value based on several parameters. This test was again conducted in the latest Hologram Analyser System where the matching location of the template is cropped from the original frame itself and saved to a folder under its frame number and template match name. This cropped image is then compared to the template which it matched using the same dimensions. The value is then saved to an empty array ready for plotting later in the processing line. In Fig.83 the plot of the SSIM against the Z depth can be seen. As was seen before the lower matching (or in this case SSIM) values occur at the lower Z depths, again below the average matching value of 0.0254 this time for the SSIM value, up to the depth of 100 pixels. However, unlike before at the 125 pixels mark the SSIM value does not drop but rather continues a slight increase with the Z depth increasing to a value of approximately 175 where it levels out and maintains a rough SSIM value of 0.032 from there on. As before plotting a series of these under different input parameters should reveal more information regarding how best to evolve the system to take these values into account for a better match.

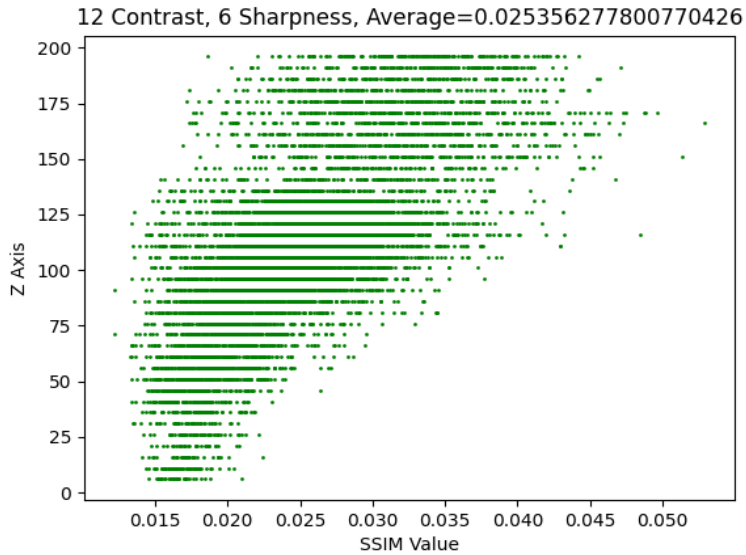


Figure 83: Plot of SSIM value against Z depth with the same input parameters as was seen for the previous plots in Fig.80, 81 and 82

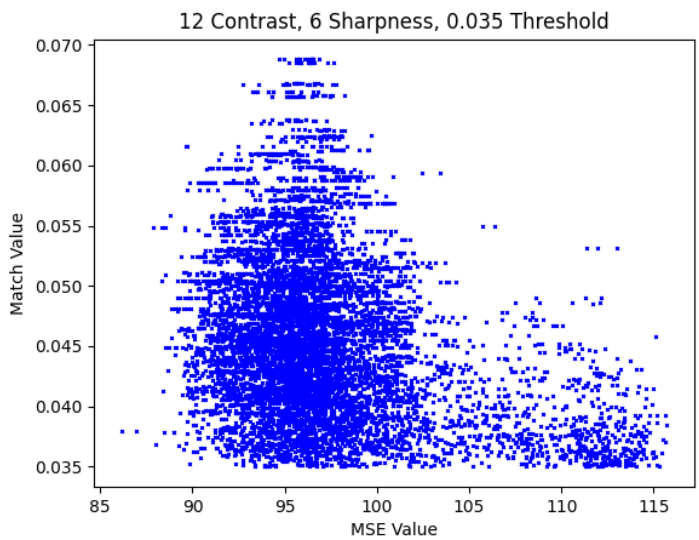


Figure 84: MSE-Th Plot showing the MSE value for the matches made against the matching value which made it

Another test performed early on in this project was the MSE value comparison for 2 different images. This is also performed in THAS and the plot of the MSE Value against the Matching Values is shown in Fig.84. The majority of the data is clearly positioned around the MSE value of 95 throughout all of the Matching values. However, there is a significant section towards the lower matching values where the MSE reaches much higher values of 115. This suggests that the higher

matching values produce lower MSE values than the lower matching values, but this does not mean the lower values are useless as the data still is in a majority around the 95 MSE mark. And so, there is some data being matched at the lower matching value which is correctly matched but there is also incorrectly matched data coming in there which could be removed based on this plot. Using this to evolve THAS to only select matched results below 100 would result in a clean-up of lower matching values with more Mean Square Error.

The final plot produced by THAS is the MSE value against the Z depth which is shown in Fig.85. An interesting curve-like structure takes place in this plot. The lower Z depths produce the highest MSE values, but these quickly decrease as the Z depths increase down to approximately the average MSE value of 97.1. However, after a pixel depth of 125 the MSE values begin increasing again almost as quickly as they came down in the latter drop. Based on this it suggests that all of the MSE values of over 100 occur at the lower Z depths and so removing these from the matches would remove all of the lower Z depth detections and hence is not a viable evolution for THAS. Instead, further testing of various input parameters should take place to compare these plots and determine how the MSE values alter under different parameters. Specifically focused on the lower Z depths as these are clearly producing the largest Mean Square Errors.

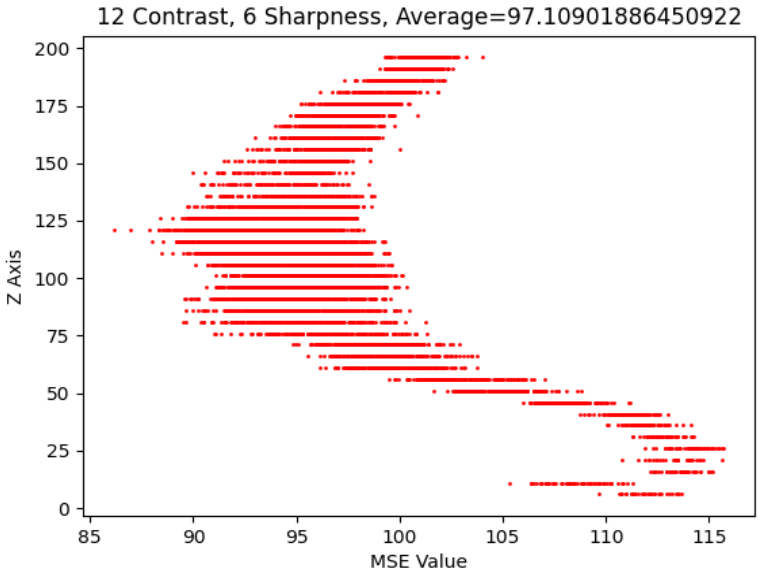


Figure 85: MSE Value against Z depth for the pervious input parameters

- Improvements to make to the system

Currently THAS is taking in a lot of data for processing. From analysis of the plots created in THAS it has become evident where specific data can be utilised to improve the system. Specifically, upon plotting the SSIM plots it became apparent that the lower Z depths are not as well matched as the higher Z depths. This is likely to be due to the template dimensions themselves, and so by decreasing the template dimensions back to 64 pixels would bring the SSIM value up. By looking at the 2D plots created it is obvious that the actual paths of movement being tracked are not able to be distinguished from each other. By using a colour bar and setting the colour to correspond to the Z depths should better show the 2D plots and enable distinguishing capabilities between paths.

By altering the contrast and sharpness values it becomes apparent that the higher contrast and sharpness allows the threshold value to be slightly increased as the templates can be better matched. Therefore, using contrast and sharpness values of 20 for all preceding plots

should also allow the threshold value to be increased. Specifically looking at the Z-Th plot it can be seen that the majority of the data is above the threshold value of 0.04. Using this value as a foundation for all preceding plots should simplify the tracking process. However, upon comparing this data to the MSE-Z plots as aforementioned, the detections below 0.04 correspond to the lower Z depths and also the higher MSE values. It is likely however that these lower Z depths are actually artifacts in the volume that are being detected. Removing these lower depths therefore should not create a negative impact on the tracking.

The plot of MSE-Z is useful for seeing the range of MSE values for each depth however it would also be useful to plot the MSE-Th values with the Z axis as a colour bar to combine several plots into a singular plot for better comparison. Furthermore, after several different parameter inputs being plotted it occurred that the MSE values could be cut off at a maximum and a minimum to better highlight the actual volume to be tracked. Allowing this to be a featured input to THAS could enable considerably more accurate plots when it comes to the 3D plotting. Finally, in terms of the 3D plotting, currently there are 2 versions of the 3D plot however neither of these versions show which direction the objects are travelling over the course of the tracking. Enabling this directional tracking and showing it in the 3D plotting would not only better highlight the different tracks but also allow directional analysis of the data.

The Hologram Analysis System 2 (THAS2) 23/02/2021

- Improved system results

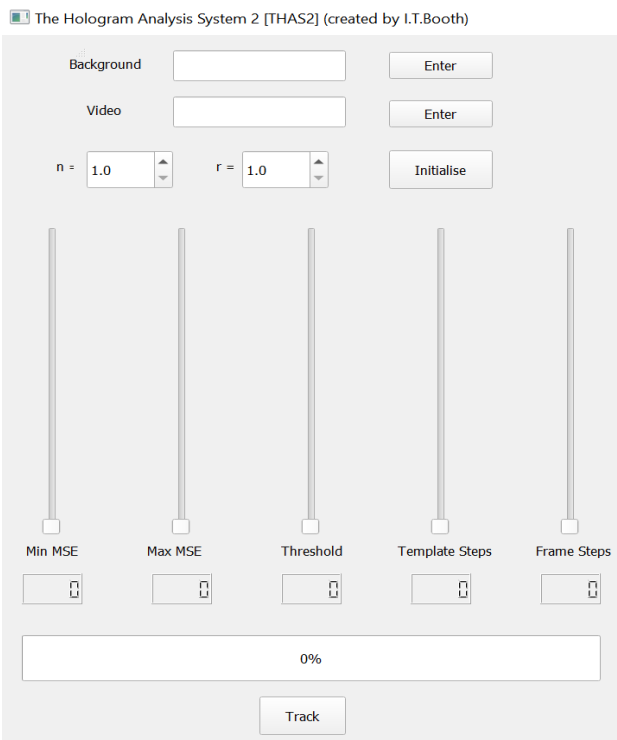


Figure 86: THAS2 parameter inputs upon start-up

The original THAS plotted the entire data set from the input parameters, and it would take approximately 10 minutes according to the estimated time feature. However, this meant that the user would have to wait 10 minutes before seeing how good or bad the input parameters were. This is a bad approach to the system in terms of taking as little time as possible to achieve good plotting results. Instead, performing an initialisation stage to quickly plot a rough outline of the data to then choose the rest of the parameters based on this would greatly speed up the data acquisition process. In THAS2 this has been implemented. The new and updated input system can be seen in Fig.86 on the left where some inputs such as the

background, video, refractive index, and sphere radius have been maintained as well as the loading bar.

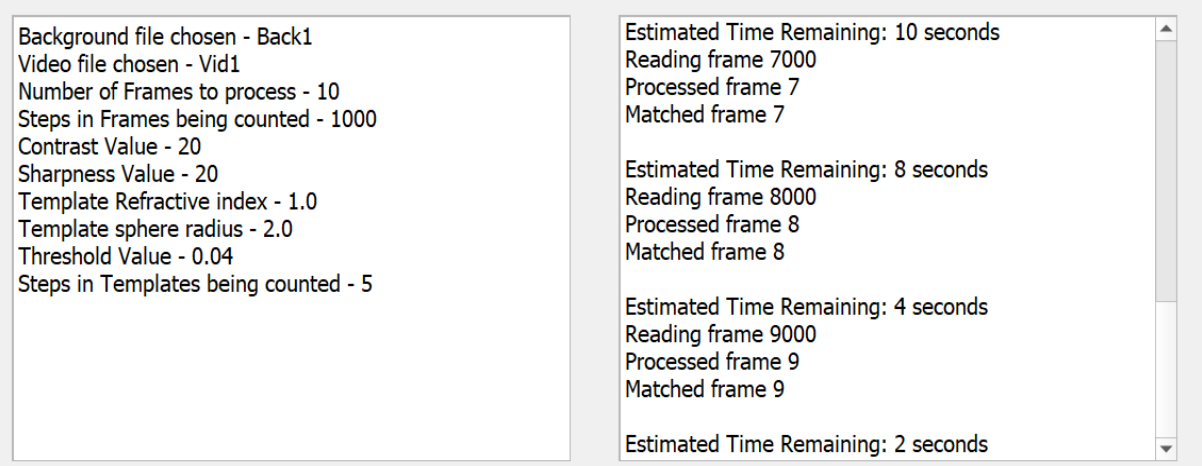


Figure 87: Output texts upon starting initialisation stage; on the left is the input specifics and the right is the loading text

As can be seen in Fig.87, the new text output system has been divided into two sections for better usage. Previously the input specifics were put at the start of the loading text and then of course the system loads, and the input data is left at the top of a large chunk of text which must be scrolled through to check the inputs are correct. Now the inputs are kept separate from the loading text as well as the final plot specifics which, once the loading has finished, are also placed on the left side.

Notably there are some new plot specifics being calculated and printed in the left shown in Fig.88. The addition of 'Recommended' settings have been implemented. This was based on the previous aims of THAS such that the system would have an autorun implemented. It was found that to perform such a feat an Artificial Intelligence protocol would be required. However, by using an approximation method to recommend parameter settings, the human brain (substituting the AI protocol) can instead use the recommendations and then select the parameters accordingly. Specifically, the threshold value (Match value), the Minimum MSE (Min MSE), and Maximum MSE (Max MSE) values have been recommended based on the percentile data of the arrays containing their information. Such that the 10th percentile for the threshold and the minimum MSE have been taken as well as the 90th percentile for the maximum MSE.

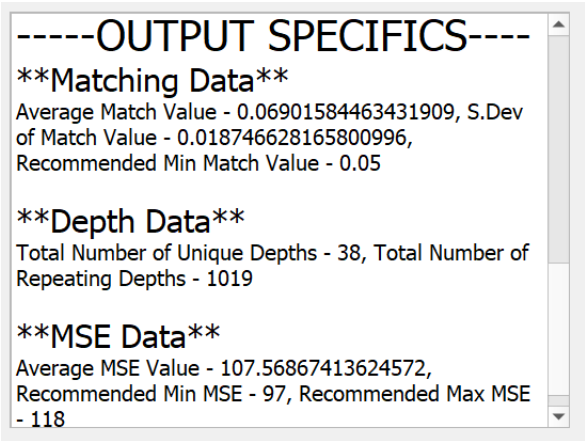


Figure 88: Output specifics to initialisation stage

In addition to the recommendations the actual plots themselves have been improved to correspond to the recommendations. The point of the recommendations for the parameter inputs is that they essentially crop out data points which have been detected which are not as well matching as the rest of the data. This not only speed up loading times for the final tracking stage but also removes unwanted matches such that the essential plots (that being

those of the 2D and 3D plots) will be less cluttered with irregularities leading to clearer interpretations of the actual tracking data overall.

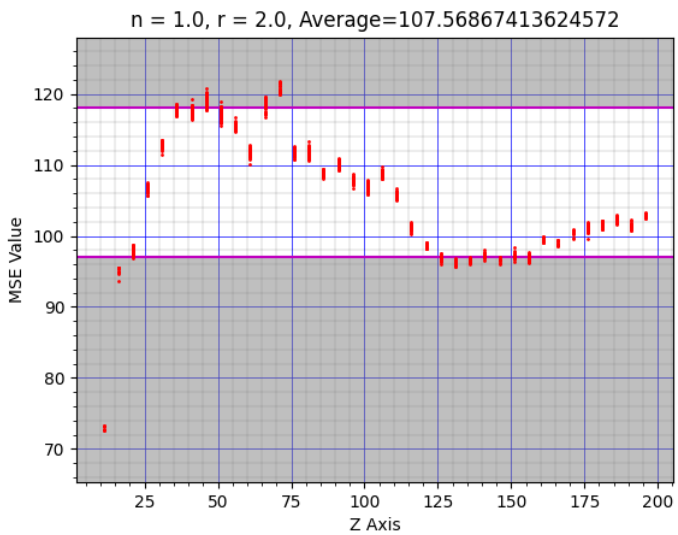


Figure 89: MSE-Z plot for the initialisation stage showing the recommendations in shaded areas

In Fig.89 is the MSE-Z output of the initialisation stage showing the recommended cut off points on the plot itself. The shaded area is left shaded so that the user (AI protocol) can use the recommendations and adjust them up or down according to the data which is going to be cropped. For example, in this figure the recommended Min MSE is 97 however for the Z depths of 135 micrometres it can be seen the shaded area crops out usable values. And so perhaps changing the recommendation to 96 instead could be a correct interpretation of the data. To further aid the choice of parameters the Average MSE value has been added to the graph title. Having minimum and maximum values roughly symmetrical to this average is a good guide to parameter setting.

The updated plot of MSE-Th is shown in Fig.90, and it should be noted the previous plot in Fig.89 is still useful despite the addition of the Z colour bar here as it is clearer which Z values are actually being omitted. Here the colour bar is more of a guide. As before the recommendations have been shaded in and highlighted with the lines including the minimum match value as well. By looking at the colour bar and seeing the distribution of these colours in the plot, it becomes apparent that while the 'yellow section' (for example) covers matching values from 0.04 up to almost 0.07, that the other colours corresponding to lower Z depths do not go below 0.05. And so, by using the minimum value recommended of 0.05 the extra matching yellow depths can be removed and the rest of the data for all Z depths should now match up equally. This ensures that there is not an abundance of certain Z depths in the 3D plot and so there is less clustering to be expected. As before with the latter recommendations for the MSE values, the threshold (match) value would need an Artificial Intelligence protocol to learn where is best to cut off the data

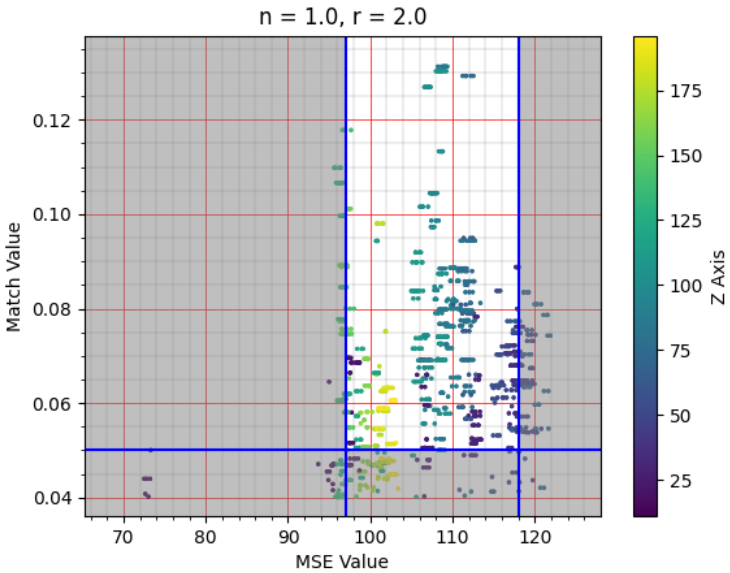


Figure 90: MSE-Th plot showing the MSE level recommendations as well as the new Z colour bar and the recommended minimum threshold value

points. While it seems that the 10th and 90th percentiles are good estimates, there are plots where it is evident that cutting off at these values would be a terrible idea as it removes a large section of data for certain depths which would lead to a bad 3D plot with large gaps of tracking. And so, these are only recommendations and not as good as an actual AI protocol would be once it has learnt what it must do.

Figure 91: The input parameters based on the initialisation stage, ready for the full tracking stage to commence

Now that the initialisation stage is complete, which took approximately 30 seconds (a drastic improvement on the previous 10 minutes of loading), the parameters can be entered for the tracking stage to commence as seen in Fig.91. Here the Recommended Min MSE value has been reduced by 1 to 96 to include the Z depths taking place at approximately 135 microns. The recommended Max MSE has been kept as well as the Threshold value (Min Match Value). The final two parameters are user preference. The lower the Template Steps, the more templates are matched against the frames of the video leading to a longer processing time however it also leads to more matches. And the Frame steps is the gaps between each frame being checked against the templates. This is also an update on the previous system where the Number of frames also had to be input. Here the total number of frames in the video is considered and from that the wanted Frame steps, the number of frames to be checked, can be calculated. This ensures that the entire video is always scanned no matter what frame step settings are chosen. For example, in Fig.87 is shows the initialisation stage settings where the frame steps are at 1000 and therefore the number of frames being checked is 10 (as the total number of frames in the video is 10,000). As with decreasing the Template Steps to a lower value increases loading time, so does the Frame Steps value. And likewise, it also provides more plotting data upon completion.

Once the tracking begins it wipes all the previous data collected during the initialisation stage from the collection arrays and begins creating data to fill these arrays based only on the input parameters newly set. For the above parameters it took approximately 8 minutes to reach completion and for visual purposes these parameter settings seem good enough as will be seen in the upcoming figures. If an in-depth plot is required, then clearly the Template Steps and Frame Steps can be reduced and upon testing this the program estimated it would take nearly 12 hours to process all of that data. Not only is this a ridiculously long time to process it also will produce a very congested set of plots to the point it is unlikely to have been worth that wait. Secondly, the way the processing takes place means it should be possible to combine two different script's results which ran

concurrently to half the processing time. By setting the first script to take data from the first half of the video and the second script to take data from the second half and then combining the saved array information to plot should half processing times and so on if more scripts are utilised.

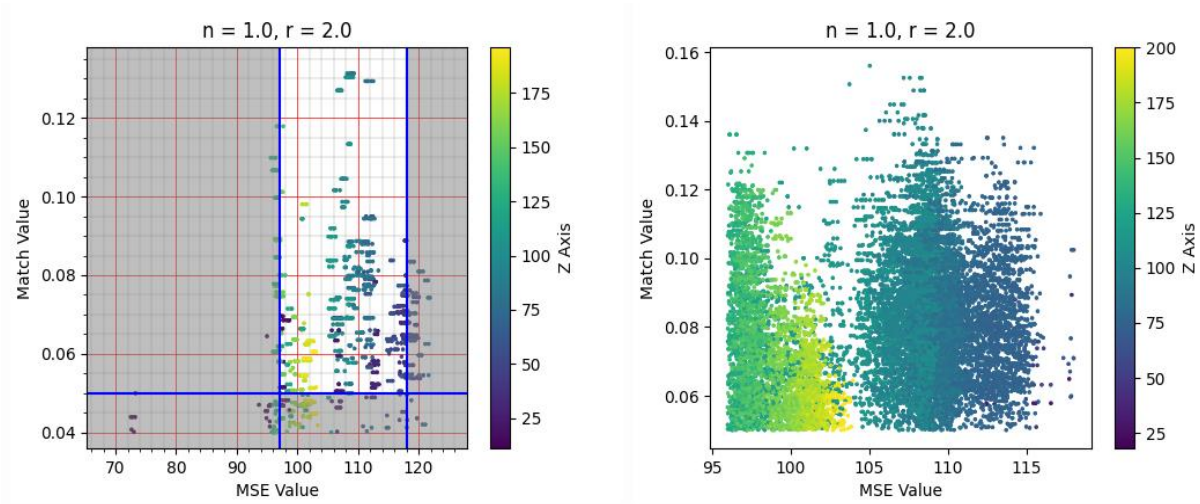


Figure 92: Plots of the MSE-Th values comparing the initialisation stage (left) the Tracking stage (right)

The first of the plots outputted is the MSE-Th plot of the 'cropped' area from the initialisation stage shown on the left to produce the more in-depth version from the tracking stage on the right. As can be seen in Fig.92, the colour distribution in the plot is very smooth for the Z axis corresponding to the MSE value. This indicates a good plot has taken place. There are versions where the distribution is very badly scattered, and it always leads to a bad scattering in the 3D and 2D plots as well. It is likely due to the choice of refractive index and sphere radius being a bad selection leading to bad templates to be matched and thus a badly represented Z depth in the preceding plots. Judging from this tracking plot the Max MSE value could have been dropped even lower to approximately 115 as there is barely any data occurring between 115 and the 118 values. As well as that the minimum threshold value seems a good match but could maybe have been better as looking towards the MSE values of 110, the data occurring there has the majority of it taking place a little higher up the Match Value axis as there are apparent gaps in the scattering towards the very base of the colour clusters. By increasing the minimum match value to approximately 0.055 instead could have levelled off the lack of values seen there and produce a more even distribution of results as well as speeding up the overall processing time.

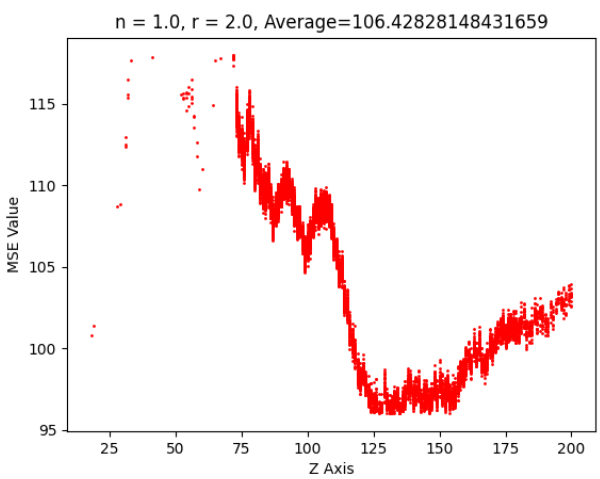


Figure 93: Plots of the MSE-Z values for the Tracking stage

The next of the plots created by the tracking stage is that of the MSE-Z values seen in Fig.93. As was highlighted in Fig.92, the lower Z depths are indeed lacking compared to the higher Z depths in terms of detections. Considerably lacking below the Z depths of 70. Due to the lowering of the Min MSE value to 96 instead of 97 it can be

seen the Z depths around 135 are now indeed just included in the plot which would have otherwise been omitted. Leading to a large gap in the tracking from a Z depth of 125 up to a Z depth of 150 microns. From this data the following 2D and 3D plots were created using the updated methods.

In Fig.94 is the 2D plot for THAS2, showing the considerably clearer tracks for the object compared to before with the removal of artifacts which were previously dotted around the plot. Along with the improved detections there is also the considerably improved distinguishing between the different object paths due to the addition of the Z axis colour bar. It is evident that the error in the X and Y dimensions is very minimal due to the clearness of the paths visible. There are also noticeable gaps in the paths in places where the archaea have moved a considerable distance between the frames being checked. In order to remedy these gaps, smaller frame disparities should be taken to detect the objects when they have moved a shorter distance. This is done by reducing the Frame Steps; however, this also leads to longer processing times as well.

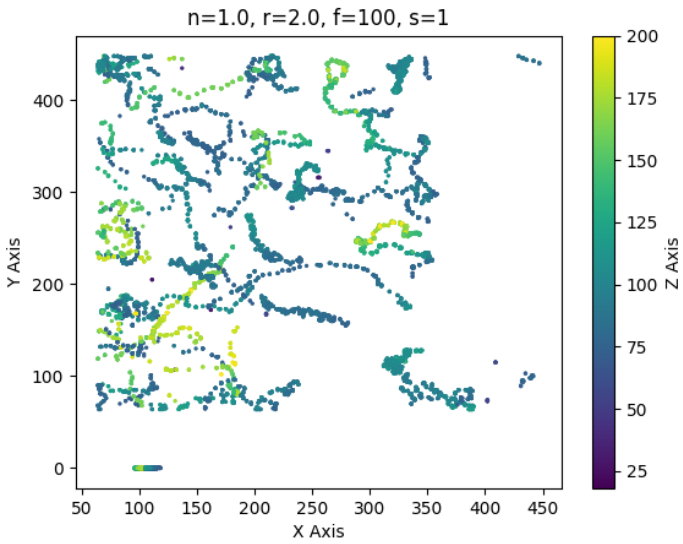


Figure 94: 2D plot of the tracking stage showing Z depths as colour

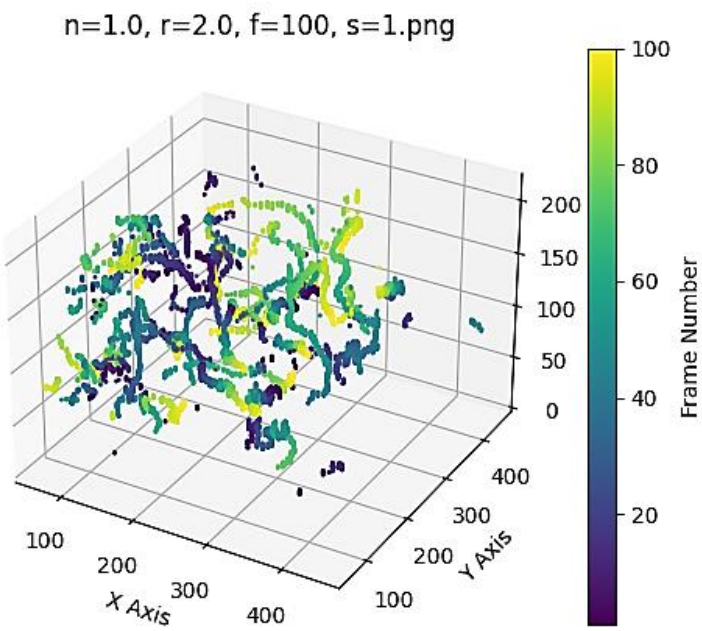


Figure 95: 3D plot of the tracking stage showing frame number as colour

Finally, the 3D plot shown in Fig.95 is upgraded such that the colour distribution shows a clean transition over the course of every tracked path. This transition denotes the traversal direction of the corresponding objects being detected by mapping their position with a colour relative to frame number.

When comparing THAS2 plot for Fig.95 against the previous 3D plot for THAS, in terms of the colour distribution it is apparent that THAS2 is considerably more useful than the predecessor. A new level of detail is now available to the user not seen before. Regarding the alterations which have occurred due to the use of the new threshold parameter inputs, it can be seen that the plots are much cleaner looking with less clustering of singular points which were thought to be

artifacts, leaving clearer plots. When comparing back to the original plots for the first tests on this, the level of improvement in terms of Z error alone has gone from a rough error of 30 microns down to only 3 or so microns.

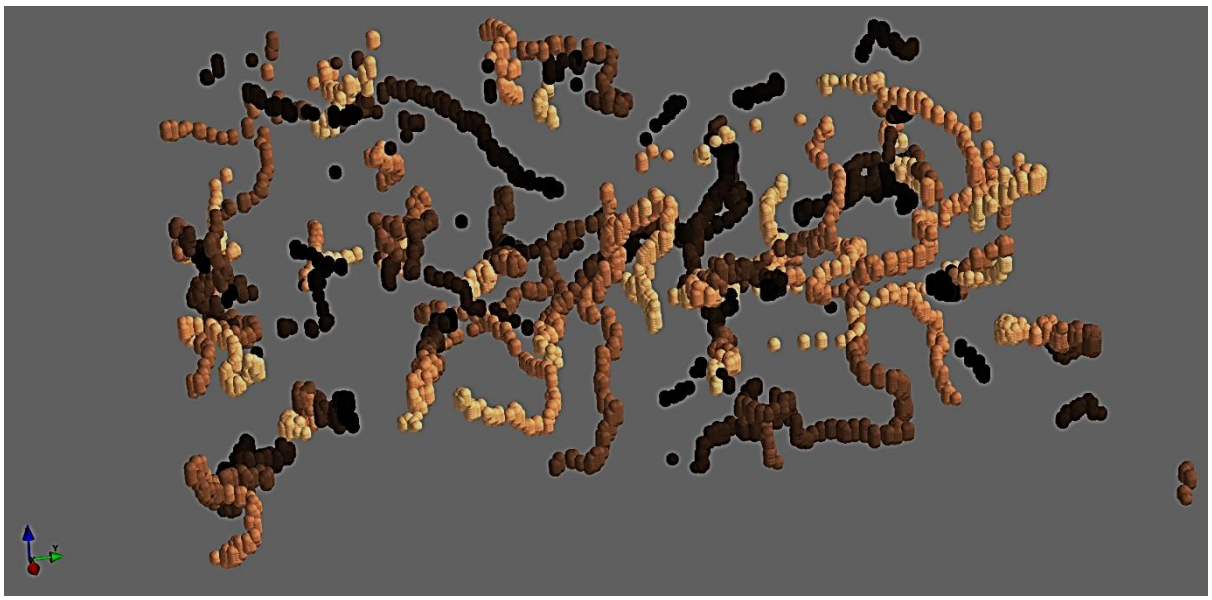


Figure 96: Mayavi2 plot for the 3D data created by THAS2 producing a much more interactive setting for analysis

While the 3D plot created by THAS2 is more visually adept, there is considerable lag in attempting to rotate the plot due to there being usually over 10,000 points plotted which the Matlibplot library struggles with. There is also the issue that the plot created only appears in the tab which is small and cannot be enlarged. This leads to the idea of using the mayavi2 library again which is better at processing 3D plots of the same sort with less lag and could be enlarged if it were cast in another window. In Fig.96 is the mayavi2 plot for the same data using copper colouring to denote frame colour as was done before. This is considerably more interactive than the previous 3D plot in Matlibplot to the point the volume can be entered and moved not only rotationally but shifted along axes allowing the user to explore every track in careful detail up close.

This specific set up requires the user to enter the Mayavi pipeline after launching the 'Interactive 3D' plotter. From there enter the 'Scalar Scatter', then 'Colors and Legends', followed by entering 'Glyph'. When in Glyph it is necessary to navigate to 'Actor', 'More options …', and tick checking 'Render points as spheres' box. Finally altering 'Point size' to whatever suits the preference; here it was set to 18.

- Discussion of the improved system

For the sake of clarity, the updated code for THAS2 has been included in Appx.6 along with in-depth explanations of every section of the code where more explanation is necessary for how every feat is performed. In order to discuss the improvements of THAS2, comparisons will be drawn on existing tracking data from an alternative tracking method performed on the same archaea video, as well as comparing various plot outputs of THAS2.

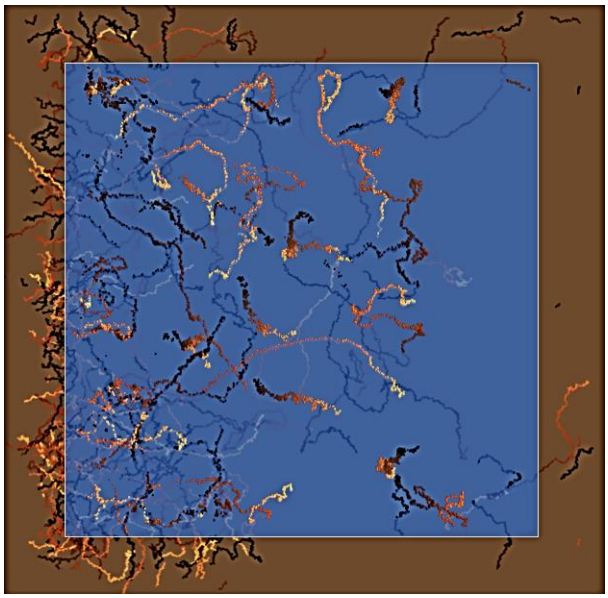


Figure 97: Overlay of the 2D THAS2 plot against the alternative tracking method

The nature of how the template matcher works by taking a 128-pixel width template and matching it against the video at all positions means that the closest to the boarder of the full video width of 512-pixels is 64-pixels on either side. This leads to a loss of information around these boarder edges as seen in Fig.97. The central blue region of this figure is the plot from THAS2 in 2 dimensions. The outer region overlapping the faded central region is the alternative tracking method. As can be seen there is a significant level of overlap, with the X and Y errors being very minimal visually, to the point they would have to be measured numerically to note any difference at all. As expected, there are areas where detections are failed to be picked up using THAS2 due to the choice of templates being used to attempt detection. By increasing the number of templates being used to attempt detection, the missing detections may be picked up and further match that of the alternative tracking method. Also, further testing of the choice of templates could render better or worse matching results as the archaea being picked up may match templates with larger or smaller radii or different refractive indices.

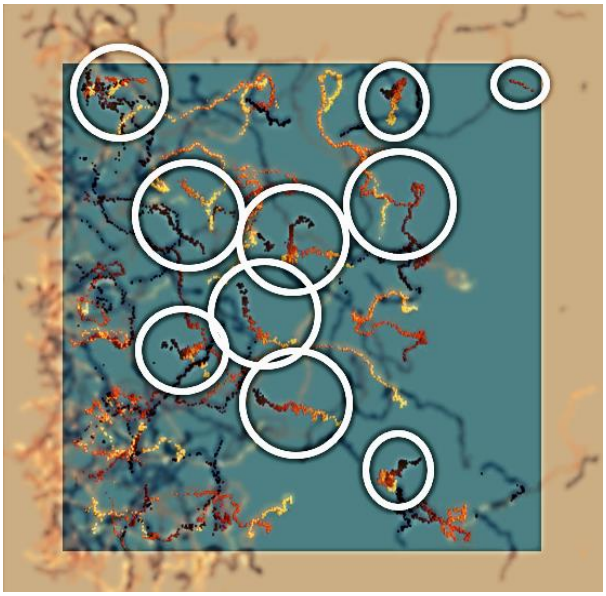


Figure 98: Highlighted extra detections picked up by THAS2

Unexpectedly however, there is also detections picked up by THAS2 which were not picked up by the alternative tracking method. Several cases of such detections picked up in this single plot alone are shown in Fig.98. When checked in 3D it is clear these are not anomalies and are continued detections of matching tracks. Furthermore, the alternative tracking method produces resulting Z depths deeper than THAS2 can. These lower Z depths correspond to lower SSIM values seen before when using THAS. If the template sizes are altered from 128-pixel widths down to 64-pixels in width, this could counteract the lack of detections found using THAS2. This is because THAS2 utilises a Threshold value which means lower Z depths are going to be omitted.

The amount of omitted Z depths can be seen more clearly by the overlay of the THAS2 plots for X and Z dimensions against the alternative tracking method seen in Fig.99 below. There can also be seen a fair degree of overlap however, the Z error seen on THAS2 is much larger

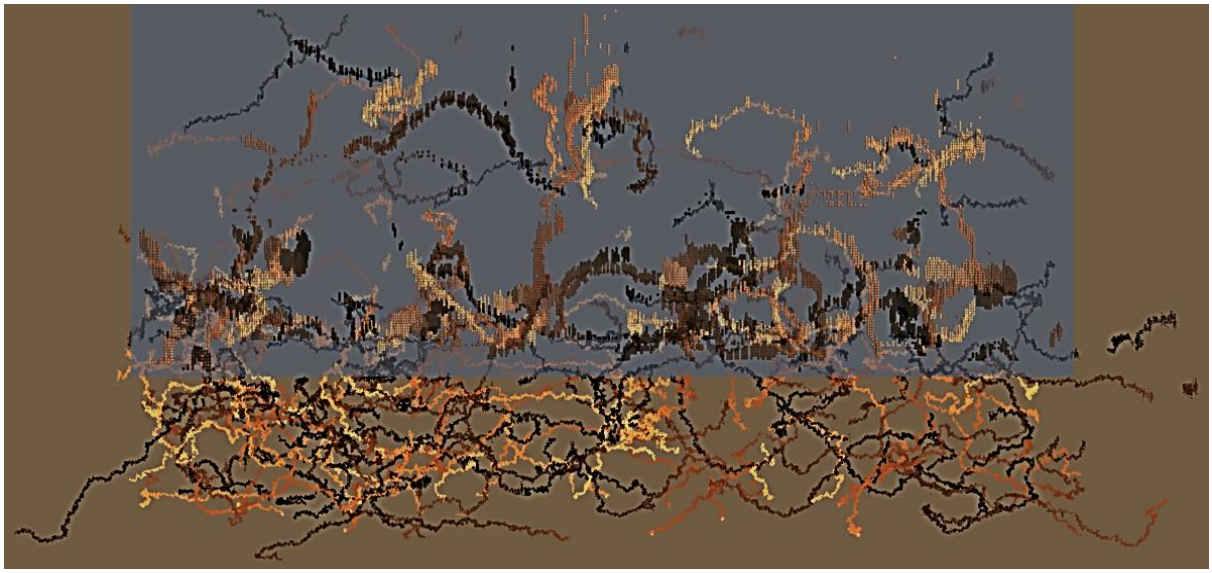


Figure 99: the overlay of THAS2 X against Z dimensions, seen in blue, over the alternative tracking method

than that of the alternative tracking method. These errors could be smoothed out using the AI protocol discussed previously. An alternative method would be to take out each of the tracks individually and then average out the X, Y, and Z coordinates at equal steps such that a line of best fit is created in 3D for each individual track. These individual tracks could then be recombined back into a full array of resulting coordinates and plotted as below in Fig.100, as was done using the alternative tracking method.

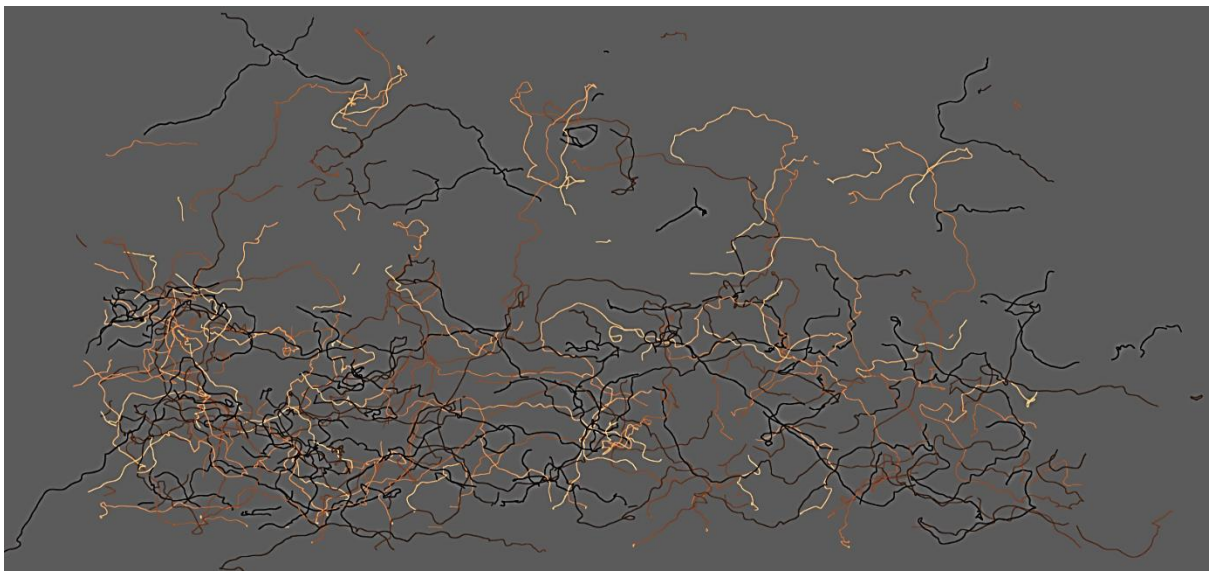


Figure 100: Smoothed plot of the entire 3D array by combining individually averaged plots back into a full system plot

There is clearly considerably less error in Fig.100 for the Z depth than that seen in Fig.99 for the same data before smoothing. While this is more visually useful there is still technically an error associated with this method due to the fact the actual path is not necessarily following exactly the perfect average path. This error is partially due to experimental uncertainty in equipment being used to collect the data as well as the lack of a perfect match occuring during the template matching process.

Computational Analysis of THAS2

02/03/2021

- Adjustments to THAS2

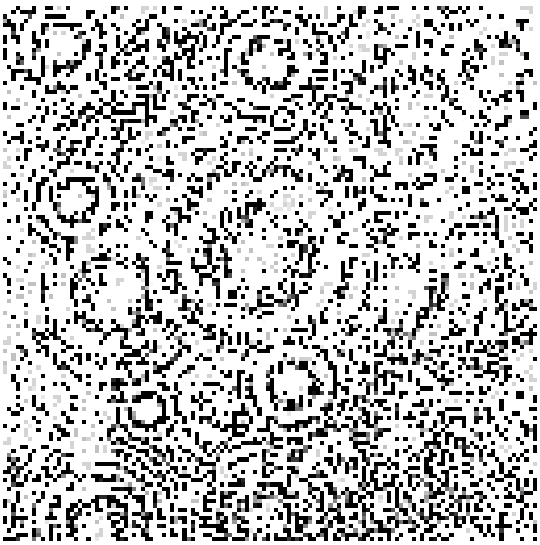


Figure 101: an example of a Cropped frame matched by THAS2 and archived for later analysis

The processing speed of THAS2 is currently comparatively slow, and this is likely due to the lack of adapting the process to be faster. As previously mentioned the processing times might be able to be halved if the script were split into multiple parts and each of those parts were to run simultaneously. Further adjustments to the coding could allow the process to go even faster. Currently THAS2 archives all of its findings into designated folders so that any specific sets of data can be looked at again and the actual process that THAS2 undertakes on every frame and every match can be scrutinised. Archiving this amount of data comes at a very expensive time cost as well as a processing cost. It may not always be necessary to archive all of the data for every test and so the addition of a 'Fast Plot' button disabling the archiving could drastically improve loading times. In some cases the archives from a single plot can be in excess of 29,000 images, specifically when recording every cropped match of a template on every frame in the 'cropped frames' folder. An example of a cropped frame match can be seen in Fig.101. This is the archive containing exactly what THAS2 is seeing when it finds a match as well as where it finds the match.

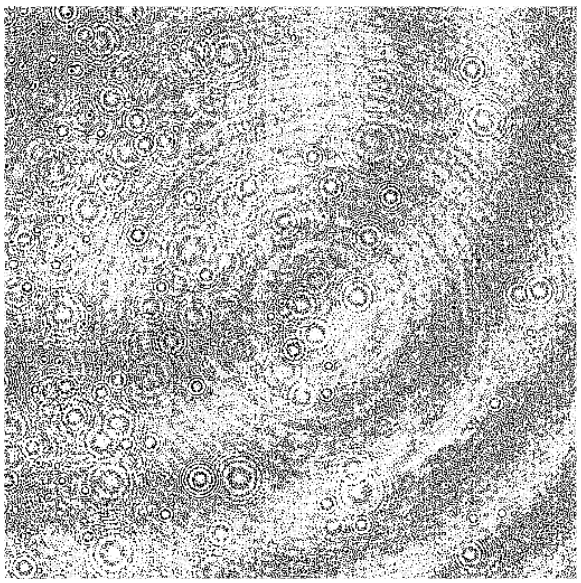


Figure 102: an example image of an enhanced frame created using THAS2 ready for cross-correlation against the template libraries

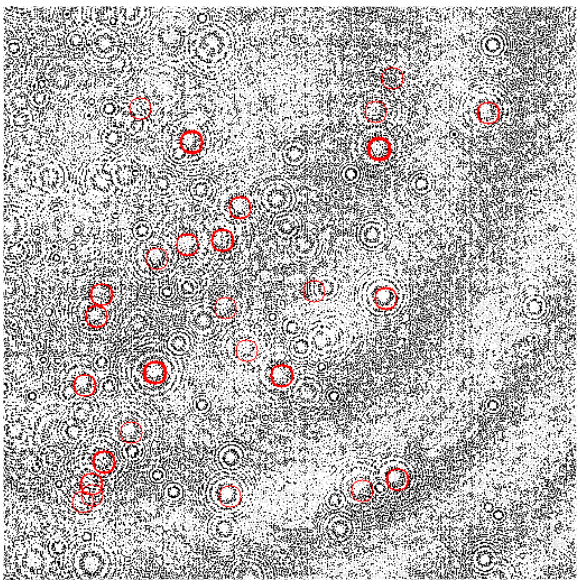


Figure 103: an example image of an enhanced frame being matched against templates with resulting matches being highlighted with red circles by THAS2

Other bulky archive folders include the 'enhanced frames' folder which contains every frame that THAS2 is cross-correlating after it has undergone parameter-set enhancements. An example of this can be seen in Fig.102 alongside Fig.103 which is of course the 'matched frames' folder which contains images which highlight every match detected on every frame as a whole.

- Resulting CPU and Memory data

Overall, THAS2 has shown its ability to detect tracks of archaea in 2D very effectively with minimal error, and in 3D with approximately 3 microns of error for the plots shown in Figs.95-100. Not only was THAS2 able to detect 3D tracks but it was able to detect tracks which were not previously known about using alternative tracking methods. However, THAS2 still suffers from a lack of ability to detect lower Z depths most likely due to the template size of these lower Z depths which could be amended. Further testing should be conducted on the actual computation taking place while THAS2 is actively running. Looking specifically into the CPU usage and levels of memory intensitive activity. If THAS2 loads using a more CPU intensive approach than memory intensive then when implemented in an optical computer this should be counteracted and the entire process speeded up considerably. However, if THAS2 uses a more memory intensive approach then the move to optical computing will have less of a drastic effect on the time it takes to process the data.

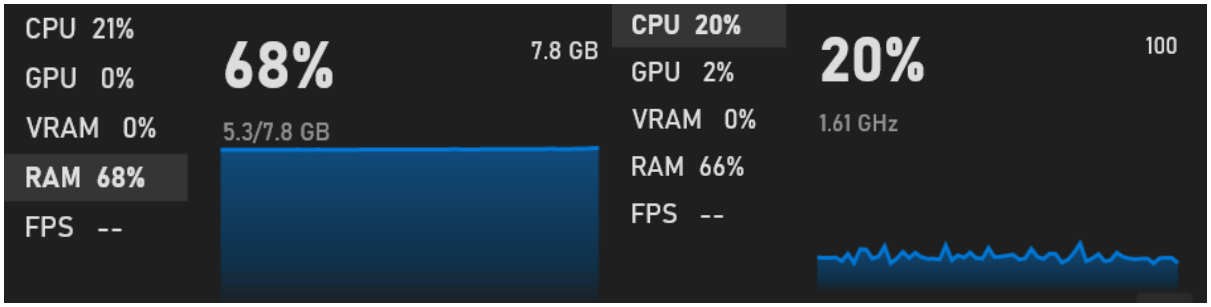


Figure 104: RAM (left) and CPU (right) measurements during the running of THAS2 when in the Tracking Stage

In Fig.104 is the aforementioned results of running THAS2 during its processing. It can be seen that the CPU percentage is up around the 20% mark for a considerable time throughout the entire process. In fact, there was a recorded high of 23% and a recorded low of 14% while running the process, but once finished the CPU percentage took a drastic drop down to approximately 2% with a maximum of 3% and a minimum of 1%. This clearly suggests a very CPU intensive approach for THAS2 to process the data and therefore suggests that once incorporated into an optical computer the processing speed should be considerably faster. In comparison, the RAM percentage was a steady (67 ± 1)% throughout the tracking stage. Once finished however the RAM did not drop but maintained its value of around 67%. This suggests that the process performed by THAS2 did not require any substantial amount of RAM in order to process the data. These tests were conducted on just a laptop rather than a heavy duty Desktop Computer. The laptop specifications are 8GB RAM, 1.6GHz i5-10th Gen CPU, with a 64-bit Windows OS. A future test on a heavy duty Desktop Computer running the same data should result in similar rises in CPU usage and

maintain its RAM. The tests will be performed on a Desktop Computer with 16GB RAM, 3.0GHz i5-7th Gen CPU, with a 64-bit Windows OS.

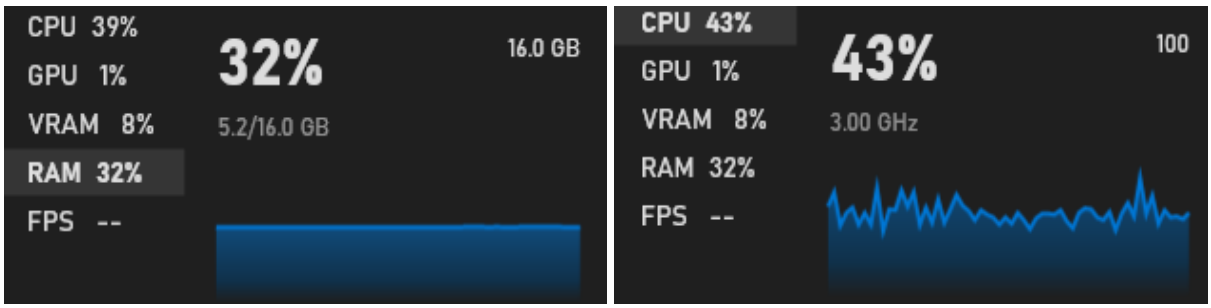


Figure 105: RAM (left) and CPU (right) measurements during the running of THAS2 when in the Tracking Stage for the Desktop Computer running the script

As can be seen in Fig.105, the overall usage of RAM effectively does not change as it is still utilising around 5.2GB of RAM, which in the case of the Desktop computer comprises of 32% of its capabilities. This again is held throughout the entire process of THAS2 as was seen using the laptop with a maximum of 8GB of RAM. However, when looking at the overall CPU usage it is apparent that the usage of the laptop is considerably less than that of the desktop despite the faster capabilities of the desktop processor. The desktop has an i5-7th Gen CPU which clocks 3.0GHz processing base frequencies, compared to the laptop which contains the i5-10th Gen CPU which only clocks 1.6GHz processing base frequencies. And so, despite the larger number of operations within a given time being passed by the desktop, the overall usage of the CPU is up compared to the laptop. Therefore, more specifics must be looked into to determine why the laptop processor is outperforming the desktop processor in terms of percentage usage.

The i5-10210U CPU contains 4 cores and 8 threads compared to the i5-7400 CPU which contains 4 cores and only 4 threads. Furthermore, the i5-10210U contains the latest intel hyperthreading technology. This is the ability to perform simultaneous multithreading (SMT), used to improve parallelisation of computations specifically on x86 microprocessors which python programming scripts fall under. Combining this technology with the threads which allow the processor to perform multiple tasks at the same time making the tasks faster, it becomes apparent that the i5-10th Gen CPU will greatly surpass that of the i5-7th Gen CPU in terms of running THAS2. This also suggests that a faster processor, as seen in that of the desktop, is not necessarily going to outperform a processor with more multitasking capabilities as seen in that of the laptop. Therefore, suggesting if a multithread CPU with the latest intel hyperthreading technology was used to run the split-script routine, it should greatly decrease processing times by simultaneously running different sections of the script in parallel. Such as the i9-10850K CPU which contains a considerably larger 20 threads capability with the intel hyperthreading technology. A CPU such as this one should in theory be capable of running the split-script routine and therefore greatly decrease running times while maintaining a moderately low CPU usage. The use of the split-script however may increase the overall CPU percentage but it is doutbful the usage increase in the 20 cores would have a drastic effect.

The Hologram Analysis System 3 (THAS3) 09/03/2021

- What needs to be improved and how

As was seen by exploring the results of THAS2, there are several areas still yet to explore which could lead to a system capable of not only more accurate three dimensional tracking of archaea traversing through a volume, but also a system where lower Z depths can be picked up amongst the deeper ones creating a full set of tracks rather than cutting off sections as seen previously. Taking each aim in turn and improving their respective parameter attributes should, in theory, lead to the improvement of THAS2 and create THAS3.

- Aims of THAS3

- ### Improved accuracy of detections

  As seen previously, THAS2 has considerably better accuracy than its predecessor THAS in terms of the Z depths. Comparatively speaking, the X and Y dimensions are of approximately equal accuracy in both cases of THAS and THAS2. However, the Z depths could still be further improved as seen when THAS2 is compared to the alternative tracking method. The reason for the inaccuracies seen here is due to the number of matches being produced for a singular set of diffraction rings. Multiple templates are being matched at roughly the correct Z depth leading to the 'blur' or 'thick-line' effect seen. If this number of repeated matches is reduced then so is the 'blur' effect. In order to reduce the number of repeated matches taking place on a singular diffraction pattern a set of rules must be adopted which allows for only certain parameters to be fulfilled and cancels any which may lead on to produce the undesired effects seen previously.

  After deeper analysis of the actual matches captured by THAS2 by searching through the archive created, it becomes apparent that a few things are happening which could be altered by applying a correct set of rules. Taking a look at Fig.106 which is of the matches picked up by THAS2 in frame 13 of the tracking stage. Which corresponds to frame 1300 of the video from which THAS2 is attempting tracking. A number of details can be seen which leads to forming a rule-set. Towards the right hand side it can be seen that there is a thin, single circle around an obvious set of diffraction rings. This is what looks like a single detection by THAS2 for a template in frame 13. However, upon comparison with the left ring it can be seen that this ring is considerably thicker. This

Figure 106: Frame 13 of tracking stage captured by THAS2 in the archive showing red circles for matches of templates

is clearly a multitude of detections picked up by THAS2 all in very close proximity as was previously seen in THAS. This is occuring due to THAS2 picking up multiple detections all with slight deviations from each other in X and Y dimensions. Therefore, the first set of rules becomes obvious, that after a detection is made in the frame, that any detections within a set radius of the primary detection must be void. In doing so the blur effect will be negated when the final 3D plot is created. However, it is not necessarily the case that all these detections were from the same template and are likely multiple different templates in close proximity to each other. And so adapting the rule to: after a detection is made in the frame, that any detections within a set radius of the primary detection must be averaged for the Z depth detected. This ensures that the information from every detection is kept. This could be further improved such that the MSE for every detection, as well as the Matching value for every detection is used as weights in calculating the average. This would be better suited for the aforementioned AI protocol however, which specialises in correctly choosing weighted values to predict correct outputs.

Furthermore, the apparent single detection seen on the right may actually also be a multitude of detections for the Z depth however they all happen to be place perfectly on top of each other. This would therefore appear as a single ring however multiple rings are infact being complied into one. A set of tests for this can be carried out, and if it shows that this is the case then the same rule applies as before, that any detections within the specified radius are void/averaged for Z depths.

Finally, it is clear that some diffraction patterns have not been detected at all in Fig.106 where at least 4 rings have not been picked up by the template system. This could be due to them not matching the particular radius being tested for by the templates or perhaps the refractive index does not match. Future testing for this frame with other template libraries may result in these diffraction rings being detected, however if it does not then it may be necessary to alter the original background frame again by changing its contrast and sharpness parameters to allow these rings to be picked up.

- Larger number of detections

While detecting a larger number of detections will obviously mean more data is collected, given the previous aim of improving the accuracy by limiting the number of repeated detections it can be seen a difficulty arises. A method for decreasing repeated results but increasing the number of previously undetected results must be acquired to result in THAS3 being an improved version of its predecessor.

A method of approximately checking how the alterations to THAS2 improve or deplete its capabilities is to count by hand the total number of visible diffraction patterns on a single frame, and compare that number to the actual number of detections found on the same frame. This method can be taken further such that the

total number of frames being checked can be multiplied by the hand counted number to give the total expected number of detections over the entire video. This can then be compared to the final number of detections made by THAS2 to see how they equate.

It should be noted however, that even though the numbers may, at some point, match up apparently very well after parameter alterations. The number of detections may include repeated results rather than new previously undetected results. To ensure this is limited as much as possible it is necessary to check on a single frame at a time and monitor how alterations effect the detections. It may also be useful to attempt to calculate how many incorrect detections are being picked up by THAS2 to have a better idea of what overall percentage of all detections are actually correct.

- Results of adaptions to THAS2

Firstly, the templates themselves must be compared for their outputted data such that the optimum refractive index and radius can be found as well as determining the accuracy of the different templates. To do this, every possible refractive index along with the range of radii must be plotted by THAS2 in the initialisation stage to give not only an overview of the plots but to give specific data outputs in the text box shown previously. This numerical data can then be plotted for each of the refractive indices in order to determine if any trends appear in the lines. From that, the optimum values can attempt to be chosen for future testing in THAS3.

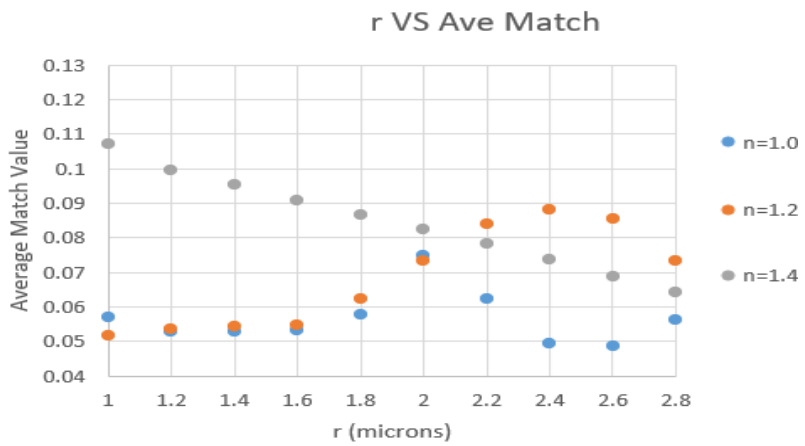


Figure 107: Combination of data taken from THAS2 for every template library showing the radius, refractive indices, and Average Match values for every plot

The first sets of plots seen in Fig.107 are those of the average match values from each individual library plot as a whole against the respective radii. By plotting this data it enables the ability to determine if there is a peak value for any set of libraries at which the chosen index and radius produce the best matches as a whole.

Previously, all plots were plotted at the 2 micron level regardless of index and as can be seen this is where the plots all converge giving similar results. However, in order to extract the best results it can be seen a maxima appears for the 1.2 refractive index at 2.4 microns radius. Using these parameter values in THAS3 tests should therefore result in better matches available. This means the threshold can be increased, reducing the number of

repeats and therefore overall improving the accuracy of the detections, achieving the first aim of THAS3.

Also of interest is the clear line produced by the 1.4 index which goes off the edge of the graph. This suggests that lower radii for this index could result in even higher match values. Future testing of these values should be conducted to determine if this could be the case.

Another comparison of interest is the Mean Square Error (MSE), against the radii of the libraries. This would show the expected error values of each parameter and may show an optimum set of values to utilise to produce matches with minimum error.

The plots in Fig.108 shows the radius against Average MSE values for every library. Unlike the previous plot in Fig.107, there are no peaks seen for any of the indices. However, the plots of 1.4 and 1.2 refractive indices do still produce clear polynomial functions.

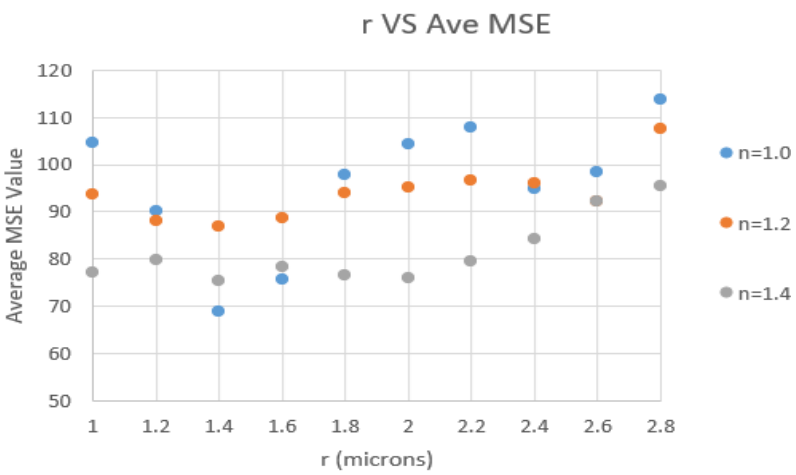


Figure 108: Combination of data taken from THAS2 for every template library showing the radius, refractive indices, and Average MSE values for every plot

Reasoning behind why such polynomials are being produced could be due to the holopy template generating system. As seen previously, specific steps should be taken between Z depths such that no resonant frequencies are produced correlating to the wavelength of the light source creating the diffraction patterns. The templates created in the libraries being plotted now are in simple steps of 1 micron and not in specific Z steps accounting for this phenomenon. Therefore, polynomial-like effects where the templates are coming into and out of phase with the light source producing them may result in different MSE values due to the different intensities being produced on each template. Because of this a system which attempts to average out the shading effects on the templates could be adapted similar to the suggestion on ***Page 48 – Fringe Patterns, Comparison of various patterns***. By taking the darker patches seen which are wanted to be removed to improve matching efficiency and replacing them with the average shade for that particular template could resolve both issues in a single step. Producing more effective matching templates and averaging the MSE values over the whole plot for that library.

While this alteration is taking place, the previous suggestion, on ***Page 68 – The Hologram Analysis System 2 (THAS2), Discussion of the improved system***, of altering the template dimensions such that they are smaller could be adopted. This allows the lower depths to be detected better, resulting in not only an increase in detections omitted as seen in Fig.106, but also the extra depths missed in Fig.99 now being detected.

The final plot created from analysis of the entire template library is that of the radius against the number of repeated values. The number of unique values was very consistent in every case throughout the testing and so was not plotted.

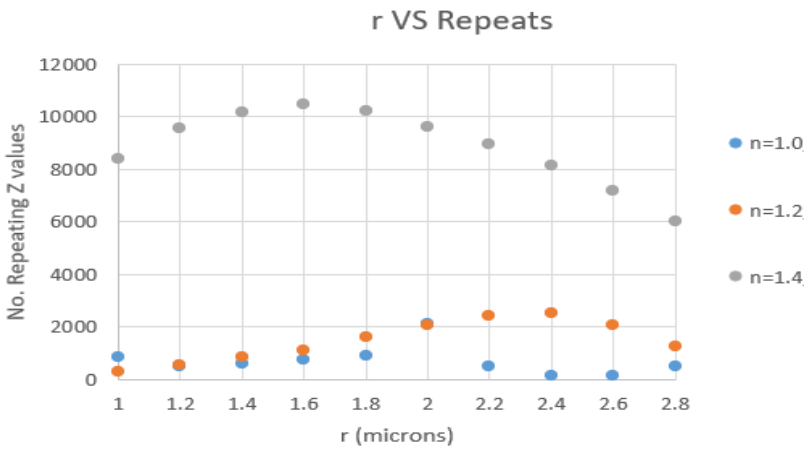


Figure 109: Combination of data taken from THAS2 for every template library showing the radius, refractive indices, and number of repeated values for every plot

In Fig.109 the previously mentioned sets of plots have been created. The refractive index of 1.2 along with the radius of 2.4 microns has coincidentally created another peak as seen in Fig.107 for these values. This further suggests these parameter values are the ideal choice for THAS3 to continue testing with. Along with these the index of 1.4 has now a peak value for the first time seen at a radius of 1.6. Testing around these parameters should also be conducted given the enormous number of detections in comparison to those picked up by the index of 1.2.

While a larger number of repeated values (at the same Z depth) can be a sign that the system is detecting a lot of useful results which it missed before. It could also be the case that these repeated values are for the same archaea which is not useful. Only testing in 3D can show which is the case by visually seeing how blurred the produced paths are.

n	r	Ave MSE	Min MSE	Max MSE	Ave Match	S.Dev Match	Rec Match	Repeats
1.2	2.4	96.042	59	111	0.0881	0.0262	0.06	2529
1.2	2.4	84.371	50	111	0.0824	0.0296	0.06	2918
1.2	2.4	93.762	55	111	0.0895	0.0312	0.05	3061
1.2	2.4	96.324	60	111	0.0959	0.0291	0.06	2697
1.2	2.4	95.782	56	112	0.1004	0.0312	0.06	3363
1.2	2.4	95.211	71	107	0.133	0.0414	0.08	9028
1.2	2.4	92.665	68	106	0.1324	0.041	0.08	14564
1.2	2.4	91.624	72	105	0.1409	0.0429	0.08	27952

Table 5: THAS3 test parameters and plot results, attempting to locate the optimum settings such that the number of repeats (detections) is at a high while also ensuring the Average Match value is kept high.

By taking the previous coding of THAS2 and performing alterations as suggests according to the previous figure analysis, the information in Table.5 was produced. The choice of index and radius was 1.2 and 2.4 microns repsectively due to the previously seen properties these exhibit. The data in the table following those values are found from the actual plots themselves.

The top row is for the original unaltered plot in THAS2. The second row is after the contrast and sharpness was increased to a relatively larger magnitude of 70 each, while alos increasing the threshold value to 0.05. This was to test how these specific parameter values for index and radius react under higher frame manipulation.

The third row is just the sharpness being left at 70 while returning the contrast back to 20. This was to see the effect on the MSE and number of repeats. The fourth row is the first implementation of the template alteration system such that the first 100 templates are shade altered. The python program for this can be found at "Python \THAS3\Alter\.template alteration.py". There can be seen a slight increase in average matching values as predicted however the MSE value increased and the number of detections dropped. Seen as the contrast value of 70 produced a lower MSE value it was reimplemented in row five as well as further splitting the template alteration. Now the alteration takes place in quarters such that the average shade for the first quarter of the library is used to replace the darker patches. Then the average shade for the second quarter of the library is used to replace the second quarter's dark patches and so on. Now the number of repeats increases as predicted while keeping the MSE value low, at the same time the average matching value increases even further.

The adoption of the altered template dimensions is then taken up in the sixth row. Here the templates are shrunk up to the half way point. Again as predicted there is a drastic increase in detections by more than three-fold on the original. It is likely these are the lower depths now being detected, but it could be the case they are incorrect detections which cause the blurring effect. At this point the continuation of adaptions to THAS2 will continue and then testing to find which is the case will be conducted later.

The seventh row, like the fifth row, the alteration changes from half way to quarterly such that the lowest depths have the smallest dimensions for the first quarter of the library, followed by a larger set of dimensions for the second quarter and so on. Once again the number of detections drastically increases likely to be even more detections being picked up at lower depths, which is now six-fold on the original number of repeats. However, there is a slight drop in matching values seen in this step.

This results in the final row adaptions such that the size of the quarterly templates are altered slightly. Keeping the quarterly steps but now with less of a step change. This final step creates an enormous leap in repeated detections to over ten-fold on the original number of detections. While achieving the wanted higher average match value. Given that the number of repeats is now at such an unexpectedly high value of near 28,000, the alterations are halted and full testing of THAS3 can now occur to see the effects of these implementations on the tracking system.

In order to best see the effects of the alterations, a comparison will be drawn at all points between the previous plots of THAS2, to the new plots of THAS3. The main areas attempting to be checked are that the alterations have had the desired affects. Those being, that the matching values will be increased for the templates attempting to be matched. This allows a higher threshold value to be set meaning the number of incorrect/false detections can be reduced achieving the first aim of THAS3. Secondly, the number of repeats overall is wanted to be increased. This could be either through detecting the lower Z depths which were thought to be omitted due to the increased template dimensions, or by detecting diffraction patterns which were otherwise not picked up when they were expected to be picked up. This was thought to be because of Mean Square Error values being inconsistent.

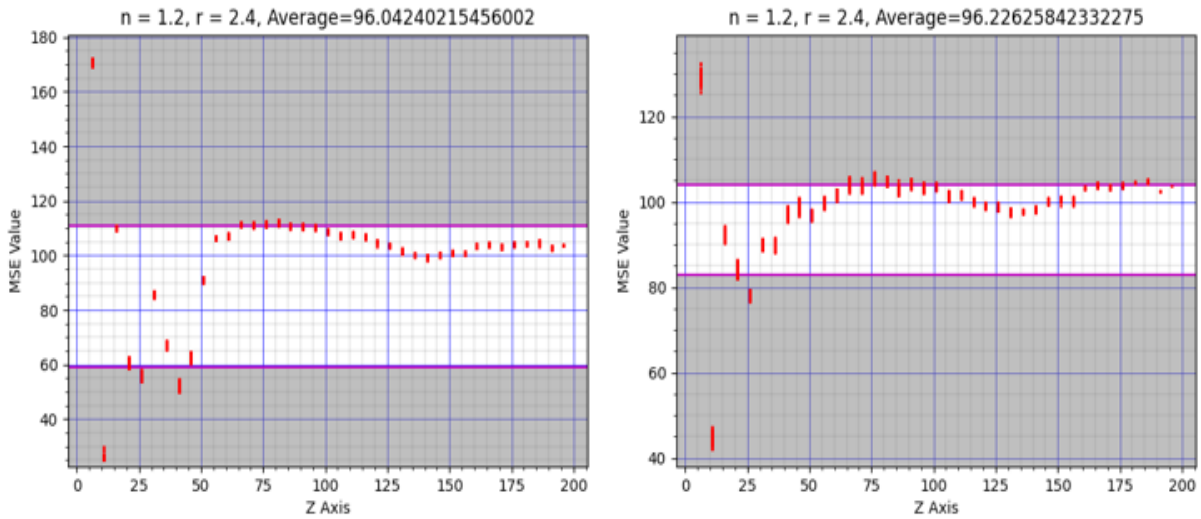


Figure 110: First THAS3 comparison of complete alterations from THAS2 of the original THAS2 plot (left) against the new THAS3 plot (right) for the MSE values against the Z depths

Upon first look, the graphs in Fig.110 appear to have undergone very little change. The depths from 50 up to 200 appear to have exhibited exactly the same results in both THAS2 and THAS3. Below 50 however it becomes apparent that the detections have become more consistent with those of the higher depths. They are exhibitting MSE values comparative to the higher values, so much so that the recommended minimum MSE cut off has been raised from approximately 60 up to approximately 83.

This was partially predicted previously such that the MSE values through all the library detections be more consistent by altering the shading in an attempt to correct for complex resonance issues. The average MSE value for the entire THAS3 plot is almost identical to that of the previous THAS2 plot.

Finally, the threshold value for THAS3 was increased to 0.07 for the initialisation stage rather than 0.04 given the data collected in the table. Given this fact, it becomes apparent just how much better THAS3 is already. Not only achieving any detections above 0.07, but to actually detect them such that the overall plots look near identical implies that the full tracking data is likely to be correct only this time with much more accuracy than before. Hence achieving the first of THAS3's aims by this stage, but for clarity this will be checked as it was before in Fig.106.

Now that the initialisation stage has confirmed that the data being collected by THAS3 is likely to exhibit excellent data, the tracking stage can now be executed. It should be noted the recommended cut off points for the MSE values are not going to be accepted for the first run. Instead the maximum cut off value will be set to 106 so as to include the tip of the data around the 75 micron depth. The lower MSE cut off value however will be accepted at 83. And the extraordinarily high threshold value that was recommended was at 0.12. Nothing this high has been seen at any point during testing of THAS2 or THAS, in fact it is more than double the usual recommended threshold value. Given deeper analysis of the MSE-Th (Z) plot however, the value was moderated to 0.1 so as to include possible useful data which would otherwise be omitted.

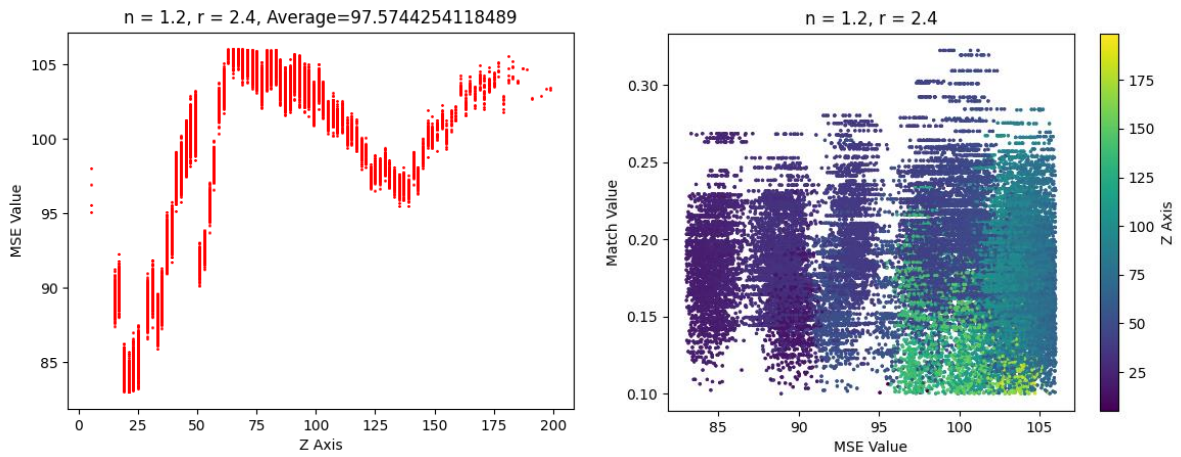


Figure 111: THAS3 tracking stage resulting plots for the MSE against Z depth and Match Value against MSE Value

As was hoped for, the MSE against Z depth plot in Fig.111 shows for the first time considerable data being collected at depths below 75, all the way down to approximately 10 microns. All of which at very reasonable MSE values which are in close proximity to those seen by the higher Z depths. There is the slight issue seen between 75 and 50 microns in depth where there is an apparent lapse in data format. This is likely due to a bad choice of template dimensions for this quarter. An amendment to the alteration system could be to further split the library up into smaller sections such that the effects seen here are limited.

As can be seen on the Match Value against MSE Value plot, there is a considerable number of detections recorded at the lower depths as was aimed for. The main point to take from this plot however is not the number of detections, but the immense magnitude of the matching values being obtained. Previous THAS2 plots would reach a maximum of around 0.12 where there were very limited results being obtained at that level. In THAS3, not only has the value of 0.12 been surpassed but it has been completely left behind. There can be seen matching values of over 0.32 in magnitude. In comparison to the usual minimum threshold value set by THAS2, the values of 0.32 and higher are more than eight-fold larger.

It should also be mentioned, not only has the matching value for the lower depths reached such enormous heights, but it has surpassed even that of the higher Z depths now. This was the reason for the lowering of the recommended Match Value as otherwise they would have been omitted altogether. This also explains the sudden rise from around 2500 results to almost 28,000 results as the majority of these results are now obviously the lower Z depths.

To counter this, it may be necessary to attempt to improve the higher Z depths detections so that they do not get left behind. However, that is not the aim of THAS3 and so should only be looked at later. This can be determined shortly upon viewing the 3D data plots and seeing how the higher Z depths are effected. It may be the case that the number of detections being picked up for the higher Z depths is still enough such that the tracks can be clearly seen and analysed, therefore there would be no need to attempt to detect more of the higher depths as the results would be adequate.

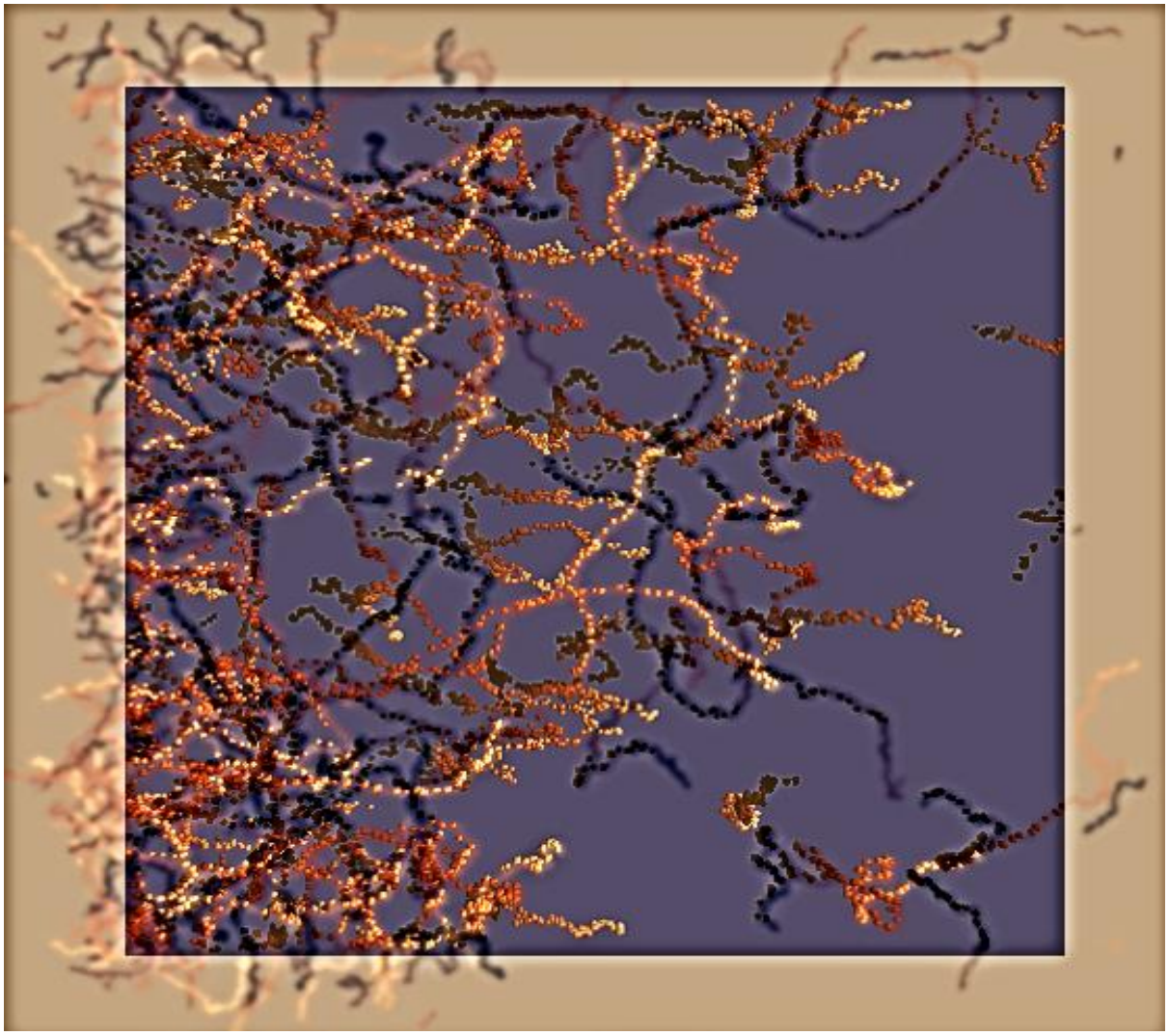


Figure 112: The 2D plot of THAS3 data overlayed on the alternative tracking method's 2D plot for comparison.

Previously, THAS2 data was compared to an alternative tracking method in terms of its 2D plotting so as to highlight areas which were detected and other areas which were not detected by either method. As can be seen in Fig.112, almost every single path detected by the alternative tracking method has now been picked up by THAS3. It can be said every track has at some point been detected but not in its entirety. As was mentioned before, by taking smaller frame steps as well as smaller template steps this is likely to be countered resulting in the entire data overlap.

As was the case with THAS2, there are still detections picked up by THAS3 which have not been detected at all by the alternative tracking method. However, upon deeper analysis, the alternative tracking method is said to utilise a removal system using statistical analysis of the tracks such that any tracks deemed to be made by non-living entities have been omitted. This would explain the majority of the tracks being picked up by THAS3 and not by the alternative tracking method, but not all of them. There are several tracks which are continuations of tracks found by the alternative tracking method and so these must be made by living entities and have simply not been picked up in their entirety by the alternative tracking method.

For comparative purposes the previous THAS2 2D plot overlayed on the alternative tracking method data has been presented next to that of THAS3 below in Fig.113.

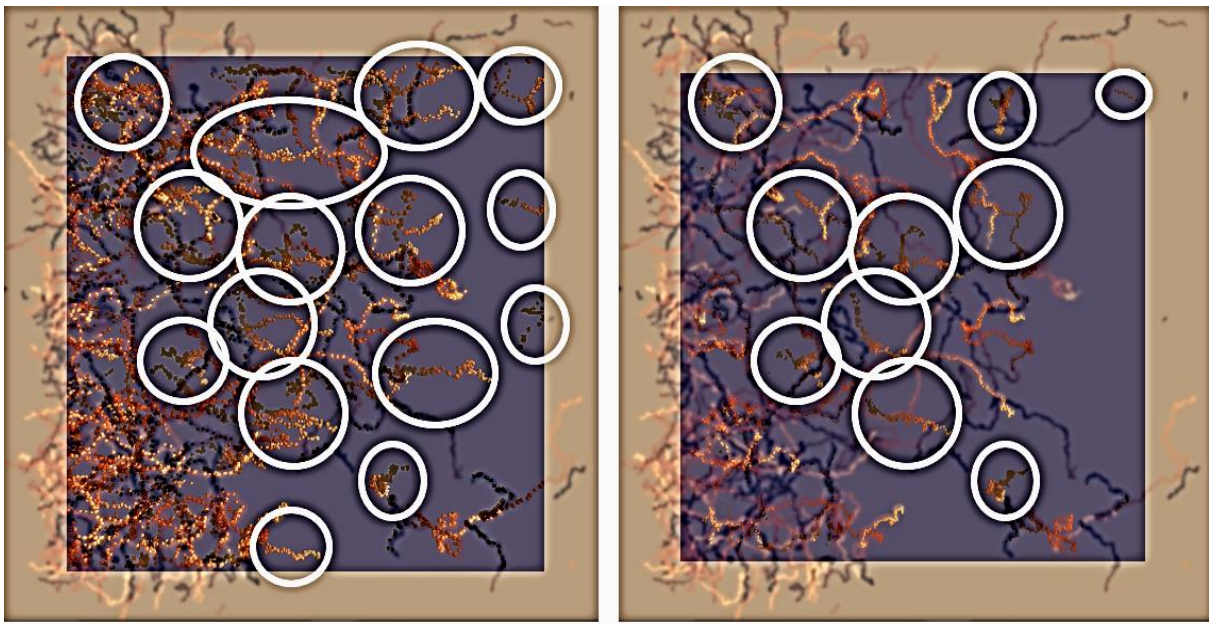


Figure 113: The comparison of the overlayed 2D plots for THAS3 (left) against THAS2 (right). Highlighted in white is the detections not picked up by the alternative tracking method which is being overlayed on

The white highlighted rings for THAS2 are indeed still present in every case of THAS3. In addition to these rings, there have been some alterations due to the new lower depth data some of the tracks highlighted in THAS2 are now larger and so have been indicated as such in THAS3 with larger rings. There is also clearly many more rings present in THAS3 shown where detections have been made which have not so by the alternative tracking method. There is likely to be many more not highlighted specifically in the lower left region where it becomes too difficult to visually decipher paths which are present or not.

There can also be noticed the slight extra depletion of the boarder around the overlays in THAS3. This is due to the use of the smaller template dimensions at lower depths, as it allows detections in a closer proximity to the outer boarder of the 512 pixel-wide frame.

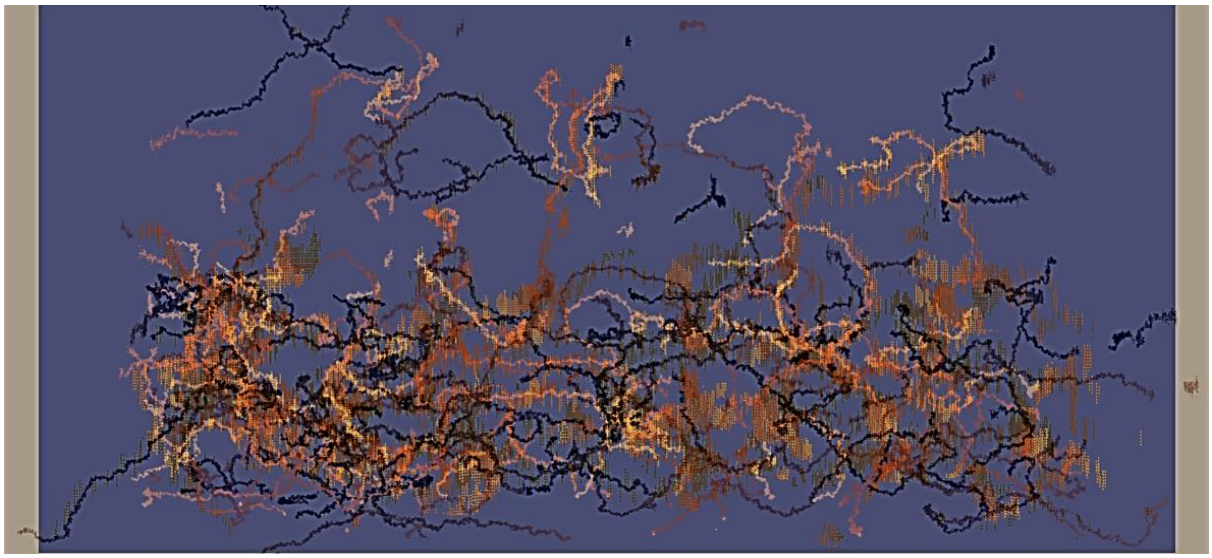


Figure 114: Cross-sectional view of the overlayed THAS3 volume against the alternative tracking method plot.

Above in Fig.114 is the cross-sectional view of the overlayed THAS3 volume data against that of the alternative tracking method plot. This shows that THAS3 now completely overlaps the alternative tracking method plot in terms of the Z depths. There also appears to be less blurring around the tracks however there is still some notable difference between the two methods.The approximate Z depth error of THAS3 is still around the 3 micron error, however taking into account the radius of the entities being detected is supposed to be approximately 2.4 microns, the value staying at around 3 microns should be very acceptable.

Following the plotting of the n=1.2 and r=2.4 template data, the next set of templates to be matched was that of the n=1.4 and r=1.6 set. While this is expected to be less effective compared to that of the previous data, it is still useful to plot and check what is produced.

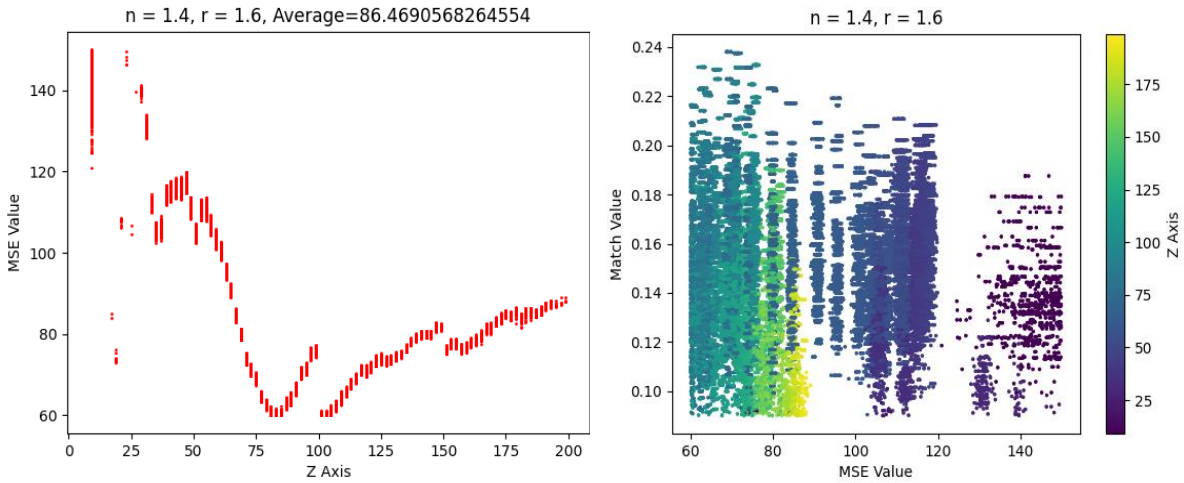


Figure 115: THAS3 tracking stage resulting plots for the MSE against Z depth and Match Value against MSE Value for the refractive index of 1.4 and the radius of 1.6 microns for the template library

Fig.115 contains the aforementioned plots. Similar affects on the MSE value against Z depths can be seen occuring quarterly due to the alteration system taking step changes. As before, reducing these steps should negate this issue and produce smoother plot results. However, unlike that of the previous plot, the values of 75 and under are more spread out in terms of the consistency of the MSE values over the plot data. While still in an acceptable range, these values are quickly approaching unacceptable.

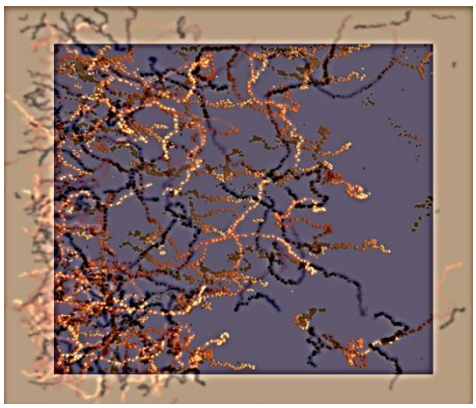


Figure 116: THAS3 2D overlayed for comparison of n=1.4, r=1.6

The matching values of the detections shown in the MSE against MSE value plot can be seen to not reach as extraordinary heights as previously seen but still considerably high, notably more than that of THAS2 still. The matching values across the plot in terms of Z depths (denoted by the colour bar) can be seen to be much more even compared to those seen previously where higher Z depths were almost omitted due to low matching values.

Taking a look at Fig.116 will show that the 2D data created by THAS3 for the latest template library is

almost identical to that of the previous 2D plot by THAS3 for the previous template library. This suggests that both sets of data are very good, with perhaps the slight hand to the previous with less gaps being seen in the tracks. This is likely due to the higher matching values seen previously. There is not any notible blurring difference between this 3D plot and the latter 3D plot when compared visually.

Also seen previously was the notable increasing matching values towards the lower radii of the refractive index of 1.4. To test this area the full track of the n=1.4 and r=1.0 data has been looked at.

As can be seen in Fig.117, this is the cleanest looking plot of MSE against Z depth produced so far with very steady and very clear patterns created. The values are all fairly low in terms of the MSE magnitude throughout the data selection reaching a peak low value at around 60 microns Z depth.

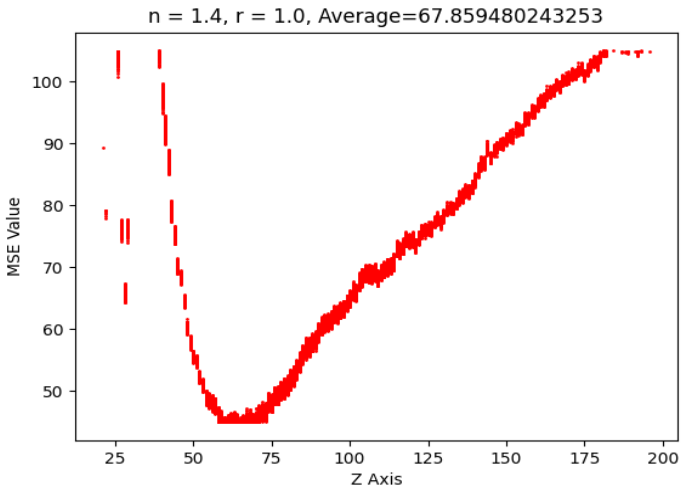


Figure 117: MSE Value against Z depths for the refractive index of 1.4 and the radius of 1.0

However, despite the seemingly good data and the apparently great low average MSE value, the actual 3D plot contains an extremely blurred set of tracks much like those seen back in the testing of the original THAS. This was unexpected based on the data seen in Fig.117 and so an explanation as to what is going on is critical at this point. Given that the only apparent change in the data found in this plot compared to the previous plots is the considerably lower MSE values, it must be these values which are impacting the blurred phenomenon.

It was previously thought that the lower the MSE, the better the resulting plot would be assuming less error in the matches leads to better matches. However, it is now apparent this is not the case. Upon reflection, a possible explanation as to why this may be occurs.

The system which is picking up high match values but producing low MSE values is effectively only matching at such high values because the images which are being matched are producing low MSE values. An image with a low MSE value has a low value because the template and the set of rings being compared to the template look very similar pixel-wise. It therefore could also be the case that the templates being compared are not of great quality, with very little feature definition.

This would result in a low MSE value being produced after the match has been made. And the match was made at a high matching level because there were very few features to match. However, because of the lack of features the MSE value is then also reported to be very low. This means that despite the low error and the high match, what has actually taken place is an ineffective set of templates has been produced with limited features leading to

'good matches' occuring which in turn produce low MSE values because of their lack of features.

It is a vicious cycle which would in effect keep repeating itself onwards and upwards in terms of the matching values as has been seen in Fig.107. In otherwords, the templates being created by holopy for the refractive index of 1.4 are ineffective for matching using THAS3. And they will only present data which appears to be good matches with low error but really, the matches are matching well for templates which are not actually a good representation of the diffraction rings created by the entities in the volume.

It is likely that as the refractive index increased still further that similar vicious cycles are witnessed with ever increasing matching values at lower radii. Of course with lower radii, the rings created on the templates are smaller leaving less features to be matched and so this matches the latter mentioned explanation perfectly. As to why this has occurred at a refractive index of 1.4, it is likely to be due to the previously mentioned intensity flip witnessed at the point where the refractive index of the synthetic entities surpasses that of the medium they are in. This phenomenon would have led onto drastic differences in the features being presented on the templates and is therefore likely to be the cause of these feature issues.

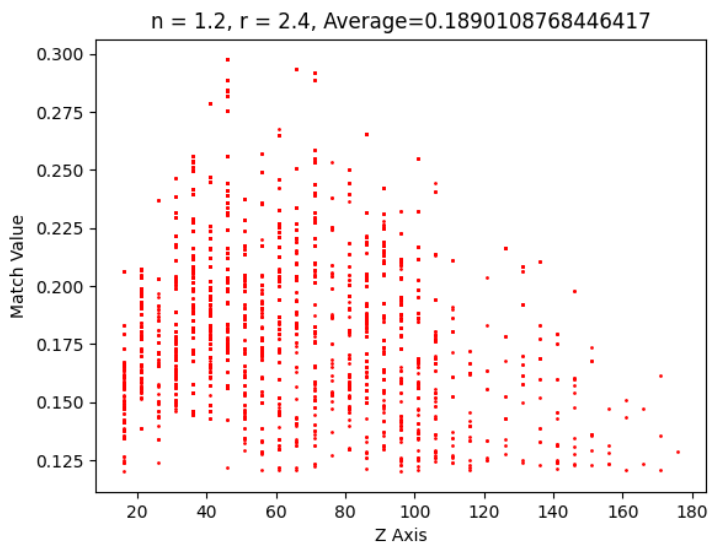


Figure 118: Reimplemented Match Value against Z depths in THAS3

Given these previous discoveries, the reimplementation of the Match Value against the Z depths was brought into THAS3 as seen in Fig.118. It was taken out previously because of the addition of the colour bar to the Match value against MSE value plot showing the Z depths. However, while this feature is useful, it does not show clearly the distribution wanted in cases where certain MSE values correspond to multiple Z depths as in these cases the colours overlap.

As seen in the latter few plots created by THAS3, the MSE values tend to be very consistent leading to many overlapping colours and so the reintroduction of the plot in Fig.118 is necessary to better display how different depths are matching.

Based on these plots it may be possible to create a synthetic frankenstein of archaea diffraction rings from multiple refractive indices and radii, such that an archive consisting of only the very best matching templates is designed. However, it should be noted that this synthetic frankenstein archive may only be the best for a particular volume at a time. Under different conditions an alternative 'best set of templates' may occur. Regardless of this, the addition of the Match Value against Z depth plot can still be utilised to find these 'best templates' in whatever situation THAS3 is given.

- Discussion of THAS3 capabilities

The aims of THAS3 were to not only improve the accuracy of detections such that multiple repeated values are limited, but to also increase the number of detections such that the number of missing detections were decreased. After considerable analysis of the templates themselves, as opposed to the frames as was previously attempted to be perfected in THAS2, several methods of alteration were decided upon and instigated. After analysing each alteration in detail to decipher if the effects of it were as intended, the adaption from THAS2 was complete. The results of THAS3 were above and beyond expectations, excelling in both the intended aims.

With the considerably increased matching values now being detected by THAS3, the threshold was able to be increased almost three-fold meaning only the most accurate matching detections would be accepted now. Therefore achieving the first aim of THAS3 in that the accuracy of the system has improved. This can be better seen below in Fig.119 where the comparison of the previous Fig.106 and the same set of matches but from THAS3 can be viewed.

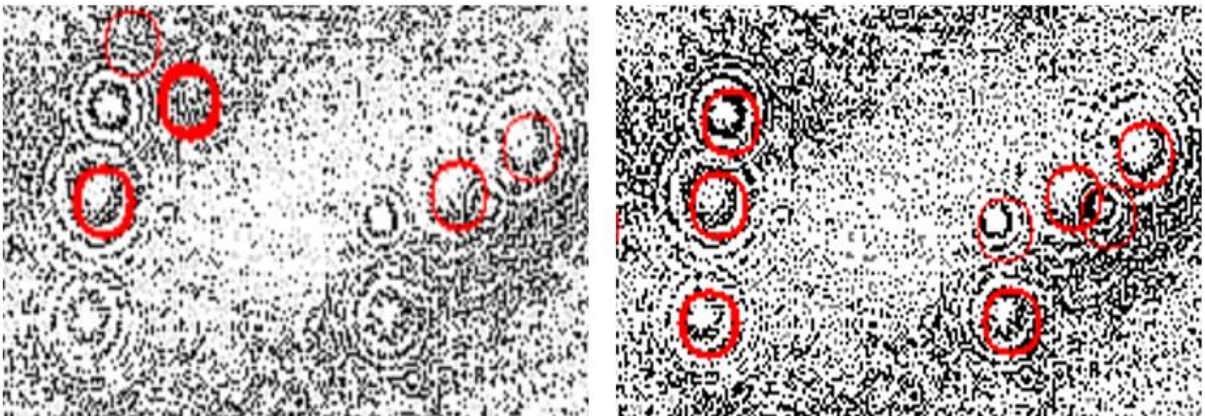


Figure 119: Comparison of detected matches shown in red circle from THAS2 (left) and THAS3 (right)

The accuracy of THAS2 was sufficiently good enough to pick up 3 clearly correct detections being that of the lower three in the left image. The upper 2 detections however do not appear to have any diffraction rings around them and so could in fact be false detections. In comparison to THAS3, the same 3 correct detections are indeed replicated but with less thickness implying that the increased threshold has removed some lesser matches, thus improving accuracy on these. Secondly, the apparent false detections have been completely eradicated in THAS3 suggesting that these were false detections let through the system due to the lower matching value threshold.

In addition to these capabilities, THAS3 has also clearly picked up 5 more detections that the previous THAS2 was completely unaware of. These extra detections are also obviously correct detections as the diffraction rings are visible without a doubt. Therefore, the second aim of THAS3 has also been accomplished in that a larger number of correct detections has been attained. To better see the extent of how many more detections have been picked up, the comparison below between THAS2 and THAS3 in Fig.120 is drawn.

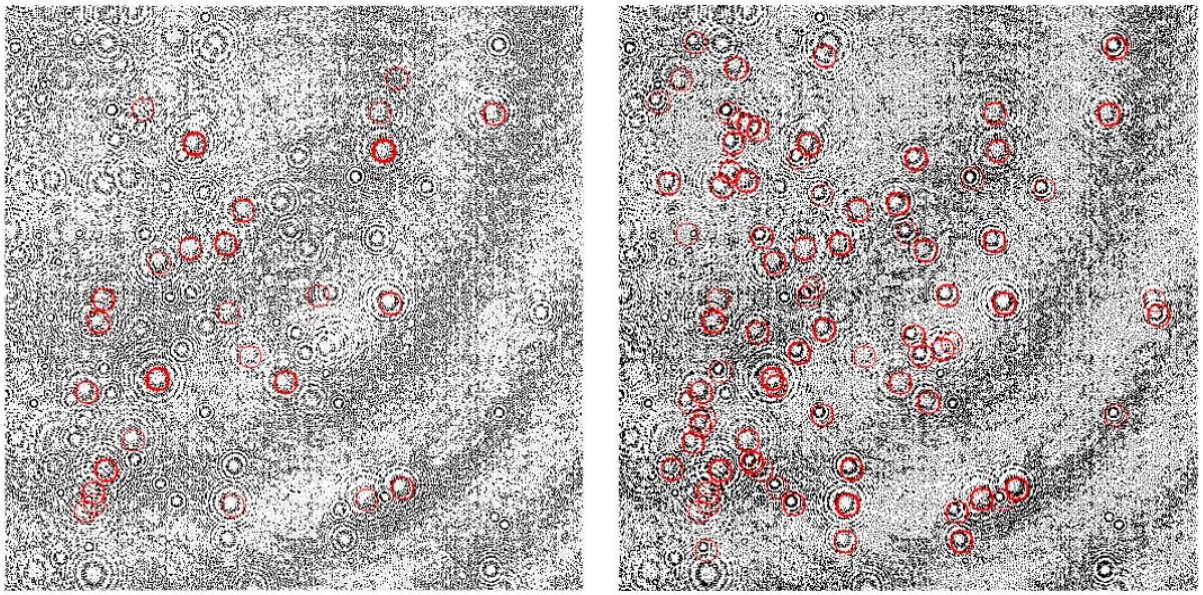


Figure 120: Comparison of THAS2 matches shown in red circle (left) to THAS3 matches (right) for the entire 13th frame

The exact number of detections picked up by THAS3 is 81 compared to just 27 detections in this frame from THAS2. This is also a three-fold increase in the number of detections however, when looking at the detections it can be seen that 3 of the detections from THAS2 are not carried onto THAS3 and all 3 do not appear to have any diffraction rings surrounding them and so it could again be the case these were false detections now removed by THAS3. Furthermore, looking at THAS3 the total number of undetected rings which a human eye could make out were only 7. There were several occasional dots however they are only points and even a human would not be able to determine rings from these dots, so an image matching system is also unlikely to be able to pick up rings which are not visually there to pick up.

Overall, the capabilities of THAS3 are exceptional. Being able to detect matches up to and above the 30% mark, removing false detections, detecting depths previously not seen before in its predecessor system, as well as making detections not picked up by an alternative tracking method. And of course, finally producing an interactive 3-dimensional plot which is cleaner looking than those of the predecessor systems due to the improved accuracy, as well as much fuller looking volumes with the deeper depths now being detected. Finally, the possible addition of a synthetic Frankenstein archive containing the best templates for matching in this particular volume as well as the potential that the archive could achieve similar outstanding results in other volumes being analysed.

Optical Computer Implementing THAS3 16/03/2021

- Optical Computer setup

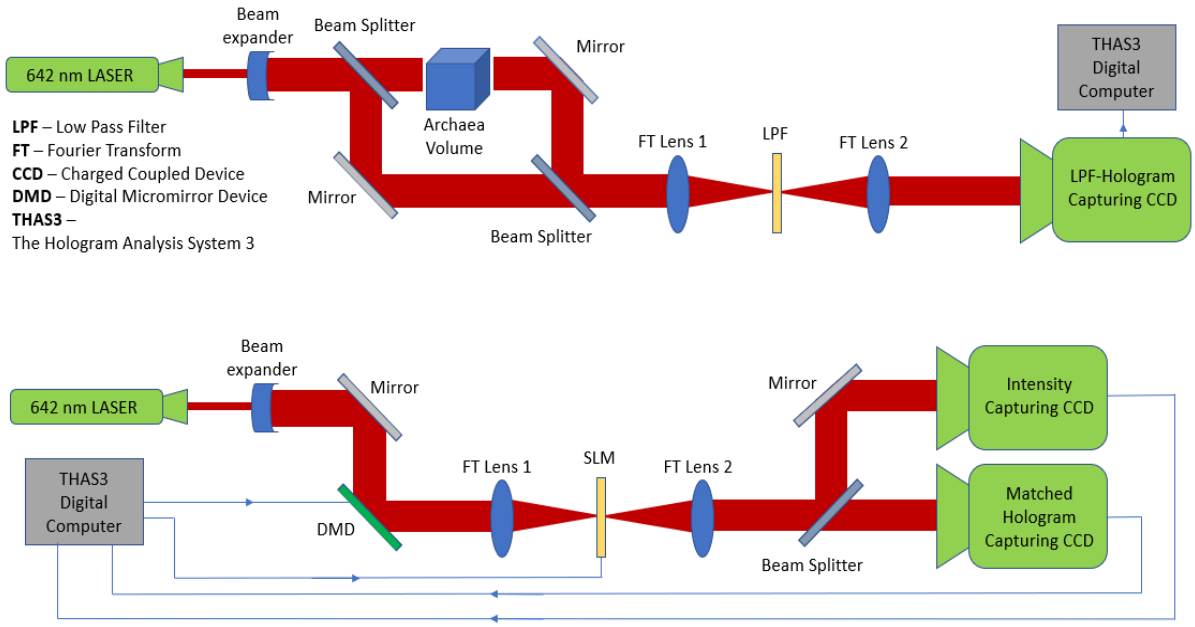


Figure 121: Optical Computer setup; the top diagram is for the hologram creation phase which would run THAS3 as previously seen, the bottom diagram is the optical computer version of the matching stage performed by THAS3

The first half of the optical computer, in Fig.121, is the hologram section which creates the necessary holograms required to be analysed by THAS3. Here simply any bacterial-sized samples can be place in the volume indicated such as the Archaea sample which has been tested previously. From there the hologram system creates two signals being the reference beam and the illumination beam. These are recombined to create the feed which is picked up by the LPF-hologram capturing device. This is after the hologram created has been filtered by a low pass filter to remove any artifacts which may remain in the system. This data is fed into THAS3 where it can be fully run using the original THAS3 script as seen in Appx.7. This gives the tracking results after having run through the digital matching system as was the case in all previous tests.

However, an updated system where the matching is performed using an optical computer is seen in the second half of Fig.121. Here the hologram feed that was picked up has been divided through by the background image to produce the typical hologram data required to be matched against. However, THAS3 now outputs the arrays (of almost any size) to be fed into the DMD such that it becomes cast through the 4F correlator. During this casting of the frames, they are obviously converted into the Fourier domain in the correlator where the SLM performs its task. The SLM produces a filter created by THAS3 by running through its template archive but this archive is not the same template archive as before, but rather the Fourier Transform copy of the archive. This means that the template itself is being scanned for in the image produced by the DMD such that the template which best matches the DMD image will give off a higher intensity result to the CCD capturing intensity readings. The Matching hologram CCD picks up the resulting images to enable a checking ability in case some results are doubted for whatever reason.

Therefore, every single frame of the hologram video is fed through the DMD by THAS3 to be matched and its location indicated in 2D based on the phase properties of Fourier Transforms. Finally, the templates give the third dimension to the data being recorded, and by passing the resulting output back through THAS3 to combine the 2D phase data with the template which produced it, creates the final combined array with all dimensions. The matching process using the Fourier Transform through an optical computer is considerably faster than using a digital computer to attempt to match the same features. The size of the array being matched can also be increased to any wanted size while not also increasing processing time as is the case in a digital computer.

- Design details

- Light source

The chosen light source for the optical computer is a 642 nm laser. This is a red diode laser such as the MDL-C-642. This particular laser is chosen due to the ease of being able to set up a laser whose rays can actually be seen by the naked eye. Also, the previous holopy tests were creating templates according to a laser of this wavelength and so continuity is necessary to validate the templates for future use.

- DMDs

The Digital Micromirror Device to be used for displaying the hologram video feed after its array has been divided through by the background image is the 'DLP® 0.23 HD DMD'. This particular DMD provides a pixel/mirror array which is of adequate dimensions being 960x540, where an array of 512x512 is necessary based on the fact the previous enhanced frames being tested by THAS3 were of this size. The update frequency is also moderate at 60 Hz, which should allow for a fast-processing system capability as a higher update frequency on the mirrors means a faster set of images able to be produced for testing. Bearing in mind the SLM update frequency needs to be higher to allow for the larger number of templates being matched against every frame of the video. The array size may also be further increased in necessary.

- Filters and SLMs

The Low Pass Filter being used to clean up the video feed of the holograms is simply a pin hole allowing only a fraction of the central spectrum to pass. The Spatial Light Modulator being utilised is the HX7318 from Himax using the LCOS technology (Liquid Crystal on Silicon). This modulator is of an adequate size being 1366x768 pixels easily allowing the Fourier Transforms of the templates to be displayed on

them for matching. The update frequency of 360 Hz is also on the high end and clearly higher than that of the DMD producing the images to be matched against.

- CCDs

The Charged Coupled Device responsible for capturing the matched hologram images as well as the LPF-hologram images will be the VL-4018. This particular device has a long focal length relatively speaking which means the objects being picked up in the image will be made to seem larger overall than would seem using a shorter focal length. In addition to this the pixel capture parameters match that of the holopy testing producing captures 1.422 microns per pixel which is useful for continuity of the previous tests.

- Analysis System (running THAS3)

Finally, the computer system running THAS3 consists of an i9-10850K CPU which contains 20 threads as well as the intel hyperthreading technology. The RAM usage was seen to reach approximately 7 GB and so 8 GB to 12 GB of RAM inside the computer would be sufficient to run THAS3 successfully. Particularly using the DDR3 version of RAM cards as seen in both of the test subjects previously.

- Anticipated results of the Computer

The hologram creation system would produce identical results to those which have been tested thus far as it is the setup which produces the hologram video and background image which is the only data THAS3 requires to perform. Therefore, the expected results of the top half of the optical computer are very good, with accurate three-dimensional tracking as well as many custom parameter abilities capable through the THAS3 interface.

Using the Split-code method which was previously discussed THAS3 was able to run the same results in a little over 2 minutes compared to previously near 14 minutes of processing. The RAM usage was up a fraction while the split code was performed, and the CPU usage was drastically up to on average around 80% as can be seen in Fig.122. This code can be found in Appx.8. as well as in the folder named 'Split' which contains all the necessary running parameters and associated files to perform the speed up. This could be even further sped up by using the higher number of threads possible from the i9 CPU previously discussed.

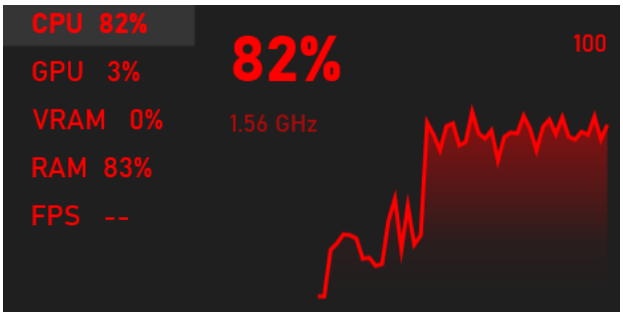


Figure 122: The CPU and RAM usage when running the Split-Code process on THAS3 to improve processing times

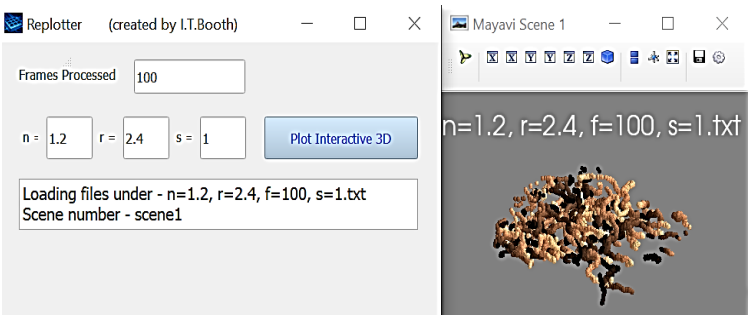


Figure 123: Replotter interface designed to quickly re-plot any saved data files created by THAS3 conveniently rather than rerun the entire THAS3

Along with the Split code itself is the 'Replotter' code. This is specifically written code which takes the 3D data gathered by THAS3 and enables it to be plotted via a user interface as seen in Fig.123. This can be found at "Python\THAS3\Replotter.py".

The second half of the optical computer is so far untested and only theoretical. This system should in theory produce results akin to those seen by the aforementioned system and do so faster. The limitation on this speed is the rate at which templates can be physically given to the optical computer to perform matching on. The 3D detection of points should be of roughly the same accuracy as before. The stage where time could be shortened is in the third-dimension calculation. Using the optical computer and its Fourier Transform abilities should allow for faster matching as not every single pixel position needs to be scanned through and calculated as is the case when using the digital THAS3. Instead, as seen in previous tests, if the feature exists in the image it will be found regardless of how large the array is that contains the template feature as the optical computer does not need increase processing times with larger array dimensions.

A faster set of DMD and SLM technology should allow for faster matching to be performed as more images can be tested against in quick succession. Finally, the rate of capture of the CCD then comes into question as this device is detecting how well every template matches the image being cast based on intensity. If it does not capture clear images and data, then the matching process is disturbed, and the threshold value which would be proportional to intensity would be affected.

The other theoretical section to this second optical computer is that of the intensity itself. It is believed that by creating a filter on the SLM in the form of a template Fourier Transform, that the light being cast by the image of a set of diffraction rings having been Fourier Transformed themselves would best fit through the filter only in the case that the best filter is present. Much like a triangle will only fit into a triangular hole, however it could also be the case that a particular square could slip through the triangular hole, and it is hoped the intensity detector will be able to tell the difference between the correct match and the false match. This is why the secondary CCD is implemented in order to check the results upon capturing to verify what is taking place. As seen with previous tests, with THAS3 having the higher matching threshold value it is hoped that this is carried through from the simulation to the physical computer, which would minimise squares being detected as triangles so to speak.

- Discussion of the Optical Computer's capabilities

Overall, the two optical systems are expected to function very well with at least the first system producing accurate 3D tracking patterns for any volume placed into the hologram system along with THAS3's range of custom parameters to fine tune the video feed and produce useful data as a result. The second half of the optical computer is anticipated to function just as adequately with a little adaption where necessary regarding possible additions of other filters and upgrades to the DMD, SLM, and CCD technology in order to speed up the processing to near real-time video analysis in 3D. And of course the increased range of array sizes which could be passed through the second optical computer with no affect on processing times is a very useful capability given just how much longer it takes THAS3 to process arrays with larger dimensions.

Conclusions

The aim of this project was to design an optical computer. This has been accomplished successfully to a satisfactory degree such that the system proposed will produce readable three-dimensional tracking of bacteria-sized objects in a short period of time with the future possibility to perform in near real-time analysis. The properties of optical computers were looked into to such a degree that they could be utilised to produce a system which focuses on the positives of optical computer attributes, while also drawing on the positives of digital computers in unison. This association between the two enables the possible future design of an optical computer which should, in theory, approach near real-time analysis which would not have been possible with just a digital computer.

The results themselves show on average only a 1 micron error on the X and Y dimensional analysis. And in the Z dimension the typical error seen was as low as 3 microns after having passed through THAS3. While keeping in mind the actual size of the objects being tracked, an error of 3 microns is relatively small when accounting for camera fluctuations as well. In terms of the processing speeds, at best, THAS3 was able to produce: an entire 3D tracking plot denoting the direction of travel, as well as produce a 2D aerial plot, an archive of every single detection cropped to size throughout the entire video feed, an archive set of enhanced video frames, and an archive set of these frames with every detection made highlighted by circling in 2D, and all this is done in just over 2 minutes with the possibility to speed this up even more with the correct computer specifications.

In terms of the optical side, the errors in the images being matched were found to be moderately low in terms of Mean Square error with an average value of around 90. The tested Structural Similarity Index was very low throughout the image testing at around 0.027. With the proposed second optical computer, the array sizes should be able to be increased to almost any size, dependent on components available, while not increasing processing time at all.

Therefore, overall this project was a success with possible areas for improvement as previously mentioned regarding future development of the optical computer system to theoretically near real-time video analysis of bacteria-sized objects to produce three-dimensional tracking, as well as possible synthetic frankenstein archives to produce even better matching results.

References

[1] E.Yu. Zlokazov 2020 *Methods and algorithms for computer synthesis of holographic elements to obtain a complex impulse response of optical information processing systems based on modern spatial light modulators,* Vol.50 pp.643 (Quantum Electronics)

Appendices

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('faces.png',0)
img2 = img.copy()
template = cv2.imread('template.png',0)
w, h = template.shape[::-1]

# All the 6 methods for comparison in a list
methods = ['cv2.TM_CCOEFF', 'cv2.TM_CCOEFF_NORMED', 'cv2.TM_CCORR',
            'cv2.TM_CCORR_NORMED', 'cv2.TM_SQDIFF', 'cv2.TM_SQDIFF_NORMED']

for meth in methods:
    img = img2.copy()
    method = eval(meth)

    # Apply template Matching
    res = cv2.matchTemplate(img,template,method)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)

    # If the method is TM_SQDIFF or TM_SQDIFF_NORMED, take minimum
    if method in [cv2.TM_SQDIFF, cv2.TM_SQDIFF_NORMED]:
        top_left = min_loc
    else:
        top_left = max_loc
    bottom_right = (top_left[0] + w, top_left[1] + h)

    cv2.rectangle(img,top_left, bottom_right, 255, 2)
    plt.subplot(121),plt.imshow(res,cmap = 'gray')
    plt.title('Matching Result'), plt.xticks([]), plt.yticks([])
    plt.subplot(122),plt.imshow(img,cmap = 'gray')
```

```
    plt.title('Detected Point'), plt.xticks([]), plt.yticks([])
    plt.suptitle(meth)

    plt.show()
```

Appendix 1: Matching Template Python code

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img_rgb = cv2.imread('sudoku1.png')
img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)

template1 = cv2.imread('1.png',0)
w, h = template1.shape[::-1]
template2 = cv2.imread('2.png',0)
w, h = template2.shape[::-1]
template3 = cv2.imread('3.png',0)
w, h = template3.shape[::-1]
template4 = cv2.imread('4.png',0)
w, h = template4.shape[::-1]
template5 = cv2.imread('5.png',0)
w, h = template5.shape[::-1]
template6 = cv2.imread('6.png',0)
w, h = template6.shape[::-1]
template7 = cv2.imread('7.png',0)
w, h = template7.shape[::-1]
template8 = cv2.imread('8.png',0)
w, h = template8.shape[::-1]
template9 = cv2.imread('9.png',0)
w, h = template9.shape[::-1]

res = cv2.matchTemplate(img_gray,template1,cv2.TM_CCOEFF_NORMED)
threshold = 0.9
loc = np.where( res >= threshold)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,255), 2)
res = cv2.matchTemplate(img_gray,template2,cv2.TM_CCOEFF_NORMED)
threshold = 0.9
loc = np.where( res >= threshold)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,255,0), 2)
res = cv2.matchTemplate(img_gray,template3,cv2.TM_CCOEFF_NORMED)
threshold = 0.9
loc = np.where( res >= threshold)
```

```
for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (255,0,0), 2)
res = cv2.matchTemplate(img_gray,template4,cv2.TM_CCOEFF_NORMED)
threshold = 0.9
loc = np.where( res >= threshold)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (255,255,0), 2)
res = cv2.matchTemplate(img_gray,template5,cv2.TM_CCOEFF_NORMED)
threshold = 0.9
loc = np.where( res >= threshold)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,100,255), 2)
res = cv2.matchTemplate(img_gray,template6,cv2.TM_CCOEFF_NORMED)
threshold = 0.9
loc = np.where( res >= threshold)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (255,0,255), 2)
res = cv2.matchTemplate(img_gray,template7,cv2.TM_CCOEFF_NORMED)
threshold = 0.9
loc = np.where( res >= threshold)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (100,0,0), 2)
res = cv2.matchTemplate(img_gray,template8,cv2.TM_CCOEFF_NORMED)
threshold = 0.9
loc = np.where( res >= threshold)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (255,100,0), 2)
res = cv2.matchTemplate(img_gray,template9,cv2.TM_CCOEFF_NORMED)
threshold = 0.9
loc = np.where( res >= threshold)
for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,0,100), 2)

plt.subplot(111),plt.imshow(img_rgb,cmap = 'gray')
plt.title('Detections'), plt.xticks([]), plt.yticks([])
plt.show()
```

Appendix 2: Multiple Detection Python Code for Sudoku numbers

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt,exp

img = cv2.imread("mona.png", 0)
original = np.fft.fft2(img)
```

```
center = np.fft.fftshift(original)

def distance(point1,point2):
    return sqrt((point1[0]-point2[0])**2 + (point1[1]-point2[1])**2)

def idealFilterLP(D0,imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            if distance((y,x),center) < D0:
                base[y,x] = 1
    return base

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

plt.subplot(161), plt.imshow(img, cmap = 'gray'), plt.title('Original'), plt.x
ticks([]), plt.yticks([])

plt.subplot(162), plt.imshow(np.log(1+np.abs(center)), "gray"), plt.title("Fou
rier"), plt.xticks([]), plt.yticks([])

LowPass = idealFilterLP(50,img.shape)
plt.subplot(163), plt.imshow(np.abs(LowPass), "gray"), plt.title("LPF"), plt.x
ticks([]), plt.yticks([])

LowPassCenter = center * idealFilterLP(50,img.shape)
plt.subplot(164), plt.imshow(np.log(1+np.abs(LowPassCenter)), "gray"), plt.tit
le("Centered * LPF"), plt.xticks([]), plt.yticks([])

LowPass = np.fft.ifftshift(LowPassCenter)
plt.subplot(165), plt.imshow(np.log(1+np.abs(LowPass)), "gray"), plt.title("De
centralise"), plt.xticks([]), plt.yticks([])

inverse_LowPass = np.fft.ifft2(LowPass)
plt.subplot(166), plt.imshow(np.abs(inverse_LowPass), "gray"), plt.title("Proc
essed Image"), plt.xticks([]), plt.yticks([])

plt.show()
```

Appendix 3: The Python code for the first fully simulated stage progression of the optical computer; specifically the 4F correlator

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
from math import sqrt,exp

from skimage import data, img_as_float
from skimage.metrics import structural_similarity as ssim
from skimage.metrics import mean_squared_error

image = cv2.imread("mona.png", 0)

original = np.fft.fft2(image)
center = np.fft.fftshift(original)
fourier = np.log(np.abs(center))

noise1 = np.ones_like(original) * 0.2 * (original.max() - original.min())
noise1[np.random.random(size=noise1.shape) > 0.001] *= -0.001

noise2 = np.ones_like(original) * 0.5 * (original.max() - original.min())
noise2[np.random.random(size=noise2.shape) > 0.001] *= -0.001

original_n1 = original + noise1
center_n1 = np.fft.fftshift(original_n1)
fourier_n1 = np.log(np.abs(center_n1))

original_n2 = original + noise2
center_n2 = np.fft.fftshift(original_n2)
fourier_n2 = np.log(np.abs(center_n2))

def distance(point1,point2):
    return sqrt((point1[0]-point2[0])**2 + (point1[1]-point2[1])**2)

def idealFilterLP(D0,imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            if distance((y,x),center) < D0:
                base[y,x] = 1
    return base

LowPass = idealFilterLP(50,image.shape)
LowPassCenter = center * idealFilterLP(50,image.shape)
LowPass = np.fft.ifftshift(LowPassCenter)
inverse_LowPass = np.fft.ifft2(LowPass)
Inv = np.abs(inverse_LowPass)

LowPass1 = idealFilterLP(50,image.shape)
LowPassCenter1 = center_n1 * idealFilterLP(50,image.shape)
LowPass1 = np.fft.ifftshift(LowPassCenter1)
```

```
inverse_LowPass1 = np.fft.ifft2(LowPass1)
Inv1 = np.abs(inverse_LowPass1)

LowPass2 = idealFilterLP(50,image.shape)
LowPassCenter2 = center_n2 * idealFilterLP(50,image.shape)
LowPass2 = np.fft.ifftshift(LowPassCenter2)
inverse_LowPass2 = np.fft.ifft2(LowPass2)
Inv2 = np.abs(inverse_LowPass2)

mse_fn1 = mean_squared_error(fourier, fourier_n1)
ssim_fn1 = ssim(fourier, fourier_n1,
                  data_range=fourier_n1.max() - fourier_n1.min())

mse_n1 = mean_squared_error(Inv, Inv1)
ssim_n1 = ssim(Inv, Inv1,
                  data_range=Inv1.max() - Inv1.min())

mse_fn2 = mean_squared_error(fourier, fourier_n2)
ssim_fn2 = ssim(fourier, fourier_n2,
                  data_range=fourier_n2.max() - fourier_n2.min())
mse_n2 = mean_squared_error(Inv, Inv2)
ssim_n2 = ssim(Inv, Inv2,
                  data_range=Inv2.max() - Inv2.min())

mse_fn3 = mean_squared_error(fourier_n1, fourier_n2)
ssim_fn3 = ssim(fourier_n1, fourier_n2,
                  data_range=fourier_n2.max() - fourier_n2.min())
mse_n3 = mean_squared_error(Inv1, Inv2)
ssim_n3 = ssim(Inv1, Inv2,
                  data_range=Inv2.max() - Inv2.min())

print("------Original VS Processed 1------")
print("==================================")
print("MSE of Fourier Noise 1 to Fourier Original: {}".format(mse_fn1))
print("SSIM of Fourier Noise 1 to Fourier Original: {}".format(ssim_fn1))
print("__________________________________")
print("MSE of Noise 1 Output to Original Output: {}".format(mse_n1))
print("SSIM of Noise 1 Output to Original Output: {}".format(ssim_n1))
print("                                  ")
print("                                  ")

print("------Original VS Processed 2------")
print("==================================")
print("MSE of Fourier Noise 2 to Fourier Original: {}".format(mse_fn2))
print("SSIM of Fourier Noise 2 to Fourier Original: {}".format(ssim_fn2))
print("__________________________________")
print("MSE of Noise 2 Output to Original Output: {}".format(mse_n2))
print("SSIM of Noise 2 Output to Original Output: {}".format(ssim_n2))
```

```
print("                                    ")
print("                                    ")

print("-----Processed 1 VS Processed 2----")
print("==================================")
print("MSE of Fourier Noise 1 to Fourier Noise 2: {}".format(mse_fn3))
print("SSIM of Fourier Noise 1 to Fourier Noise 2: {}".format(ssim_fn3))
print("__________________________________")
print("MSE of Noise 1 Output to Noise 2 Output: {}".format(mse_n3))
print("SSIM of Noise 1 Output to Noise 2 Output: {}".format(ssim_n3))

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

plt.subplot(331), plt.imshow(image, cmap = 'gray'), plt.title('Original'), plt
.xticks([]), plt.yticks([])
plt.subplot(332), plt.imshow(fourier, "gray"), plt.title("Fourier Original"),
plt.xticks([]), plt.yticks([])
plt.subplot(333), plt.imshow(Inv, "gray"), plt.title("Processed Original"), pl
t.xticks([]), plt.yticks([])

plt.subplot(334), plt.imshow(image, cmap = 'gray'), plt.title('Original'), plt
.xticks([]), plt.yticks([])
plt.subplot(335), plt.imshow(fourier_n1, "gray"), plt.title("Fourier Noise 1")
, plt.xticks([]), plt.yticks([])
plt.subplot(336), plt.imshow(Inv1, "gray"), plt.title("Processed Noise 1"), pl
t.xticks([]), plt.yticks([])

plt.subplot(337), plt.imshow(image, cmap = 'gray'), plt.title('Original'), plt
.xticks([]), plt.yticks([])
plt.subplot(338), plt.imshow(fourier_n2, "gray"), plt.title("Fourier Noise 2")
, plt.xticks([]), plt.yticks([])
plt.subplot(339), plt.imshow(Inv2, "gray"), plt.title("Processed Noise 2"), pl
t.xticks([]), plt.yticks([])

plt.show()
```

Appendix 4: The Python code for the implementation of defects into the Fourier stage, specifically for the Mona Lisa image.

```
from prysm import FringeZernike, PSF, MTF, PixelAperture, OLPF
from matplotlib import pyplot as plt
%matplotlib inline
plt.style.use('bmh')

# data from a wavefront sensor, optical design program, etc...
coefficients = [0, 0, 0, 0, 0.1, 0.1, -0.025, -0.025, 0.1]
```

```
# a circular aperture inscribed in a square 10mm on a side with 50mm EFL
# note the default mask is a circle, so the kwarg is somewhat redundant here.
pupil = FringeZernike(coefficients, dia=10, transmission='circle', z_unit='um'
, norm=True)
psf = PSF.from_pupil(pupil, efl=40)  # F/2

pupil.y

fig, ax = pupil.plot2d(interpolation='nearest')
ax.grid(False)
ax.set_title('Wavefront')

fig, ax = psf.plot2d(xlim=20, power=1/2)  # 1/2 stretch, colorbar scales as we
ll.
ax.grid(False)
ax.set_title('PSF');

mtf = MTF.from_psf(psf)
mtf.slices().plot(['x', 'y', 'azavg'], xlim=(0,200))

pixel_pitch = 5  # 5 micron diameter pixels
aa_filter = OLPF(pixel_pitch*0.66)
pixel = PixelAperture(pixel_pitch)
sys_psf = psf.conv(aa_filter).conv(pixel).renorm()  # renorm so max=1

sys_psf.plot2d(xlim=20, interpolation='lanczos', power=1/2)  # sys_psf is a Co
nvolvable, not a PSF.
plt.grid(False)

sys_mtf = MTF.from_psf(sys_psf)
sys_mtf.slices().plot(['x', 'y', 'azavg'], xlim=(0,200))
```

Appendix 5: The Jupyter code for modelling the MTF for a full optical system through a low pass filter and a lens being picked up by a sensor

```
#
       #
#
       #
#                                Created by I.T.Booth
       #
# ---------------------------------------------------------------------------
------ #
#
       #
#                           The Hologram Analysis System 2
       #
```

```
# ---------------------------------------------------------------------------
------ #
#                                    (THAS2)
       #
#
       #
# Attempts to match video frames against an extensive library of synthetic tem
plates #
# Specific properties about the templates and the frames themselves can be cho
sen    #
# Once chosen, the program performs several operations producing plots of inte
rest   #
# Displaying and saving the created DATA for later analysis by user if necessa
ry     #
#
       #
#
       #



########## Program Import Requirements ###########
import sys
import time
from PyQt5 import (QtCore, QtGui, QtWidgets)
from PyQt5.QtCore import QProcess
from PyQt5.QtWidgets import (QWidget, QMessageBox, QLabel, QHBoxLayout, QAppli
cation)
import numpy as np
from numpy import arange
import PIL
from PIL import Image, ImageEnhance, ImageFilter
import matplotlib
from matplotlib.patches import Rectangle
from matplotlib import pyplot
import matplotlib.pylab as plt
from matplotlib.backends.backend_qt5agg import FigureCanvas
from matplotlib.backends.backend_qt5agg import NavigationToolbar2QT as Navigat
ionToolbar
from math import sqrt,exp
import cv2
from mpl_toolkits.mplot3d import Axes3D
import array as arr
import keyboard
from pandas import read_csv
from scipy.optimize import curve_fit
matplotlib.use('QT5Agg')
```

```
########## Autorun callable settings ##########
background = "Back1" # Background image file name
video = "Vid1" # Video file name
count = 1 ## NON ALTERABLE ##
end = 11 # Number of frames to process (10) +1
nth_frame = 1000 # Taking frames every nth step (x/(end-
1)) where x=total frames
contrast = 20 # Enhancing frame contrast
sharpness = 20 # Enhancing frame sharpness
brightness = 250 # Sets pixel array multiplier
tempcount = 1 # Starting template
tempend = 201 # Ending template
tempsteps = 5 # Taking templates every nth step
n = 1.0 # Synthetic template refractive index
r = 2.0 # Synthetic template radius of sphere
Location = ("n={}, r={}" .format(n, r)) ## NON ALTERABLE ##
threshold = 0.04 # Threshold value for template matching
MinMSE = 0 # Sets minimum MSE value acceptable
MaxMSE = 0 # Sets maximum MSE value acceptable

PlotX = [] ## NON ALTERABLE ##
PlotY = [] ## NON ALTERABLE ##
PlotZ = [] ## NON ALTERABLE ##
PlotTH = [] ## NON ALTERABLE ##
PlotMSE = [] ## NON ALTERABLE ##
PlotCOUNT = [] ## NON ALTERABLE ##




########## Main Setup ###########
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("ANALYSER")
        MainWindow.resize(1900, 950) # Initial Window Dimensions
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")



        ########## Labels ##########
        self.BackgroundLabel = QtWidgets.QLabel(self.centralwidget)
        self.BackgroundLabel.setGeometry(QtCore.QRect(80, 27, 100, 20))
        self.BackgroundLabel.setObjectName("BackgroundLabel")
        self.VideoLabel = QtWidgets.QLabel(self.centralwidget)
        self.VideoLabel.setGeometry(QtCore.QRect(100, 87, 100, 20))
        self.VideoLabel.setObjectName("VideoLabel")
        self.nLabel = QtWidgets.QLabel(self.centralwidget)
        self.nLabel.setGeometry(QtCore.QRect(65, 160, 20, 20))
```

```
        self.nLabel.setObjectName("nLabel")
        self.rLabel = QtWidgets.QLabel(self.centralwidget)
        self.rLabel.setGeometry(QtCore.QRect(250, 160, 20, 20))
        self.rLabel.setObjectName("rLabel")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(30, 660, 100, 20))
        self.label_3.setObjectName("Min MSE")
        self.label_4 = QtWidgets.QLabel(self.centralwidget)
        self.label_4.setGeometry(QtCore.QRect(170, 660, 100, 20))
        self.label_4.setObjectName("Max MSE")
        self.label_5 = QtWidgets.QLabel(self.centralwidget)
        self.label_5.setGeometry(QtCore.QRect(325, 660, 100, 20))
        self.label_5.setObjectName("Threshold")
        self.label_6 = QtWidgets.QLabel(self.centralwidget)
        self.label_6.setGeometry(QtCore.QRect(460, 660, 120, 20))
        self.label_6.setObjectName("Template Steps")
        self.label_7 = QtWidgets.QLabel(self.centralwidget)
        self.label_7.setGeometry(QtCore.QRect(620, 660, 120, 20))
        self.label_7.setObjectName("Frame Steps")



        ########## Loading Bar ##########
        self.progressBar = QtWidgets.QProgressBar(self.centralwidget)
        self.progressBar.setGeometry(QtCore.QRect(25, 780, 670, 60))
        self.progressBar.setProperty("value", 0)
        self.progressBar.setObjectName("progressBar")



        ########## LCDs ##########
        self.MinMSElcd = QtWidgets.QLCDNumber(self.centralwidget)
        self.MinMSElcd.setGeometry(QtCore.QRect(25, 700, 70, 40))
        self.MinMSElcd.display(0)
        self.MinMSElcd.setObjectName("MinMSElcd")
        self.MaxMSElcd = QtWidgets.QLCDNumber(self.centralwidget)
        self.MaxMSElcd.setGeometry(QtCore.QRect(175, 700, 70, 40))
        self.MaxMSElcd.display(0)
        self.MaxMSElcd.setObjectName("MaxMSElcd")
        self.Thlcd = QtWidgets.QLCDNumber(self.centralwidget)
        self.Thlcd.setGeometry(QtCore.QRect(325, 700, 70, 40))
        self.Thlcd.display(0)
        self.Thlcd.setObjectName("Thlcd")
        self.Templcd = QtWidgets.QLCDNumber(self.centralwidget)
        self.Templcd.setGeometry(QtCore.QRect(475, 700, 70, 40))
        self.Templcd.display(0)
        self.Templcd.setObjectName("Templcd")
        self.Framelcd = QtWidgets.QLCDNumber(self.centralwidget)
        self.Framelcd.setGeometry(QtCore.QRect(625, 700, 70, 40))
```

```
        self.Framelcd.display(0)
        self.Framelcd.setObjectName("Framelcd")



        ########## Sliders ##########
        self.MinMSESlider = QtWidgets.QSlider(self.centralwidget)
        self.MinMSESlider.setGeometry(QtCore.QRect(30, 250, 60, 400))
        self.MinMSESlider.setMinimum(0)
        self.MinMSESlider.setMaximum(150)
        self.MinMSESlider.setPageStep(1)
        self.MinMSESlider.setOrientation(QtCore.Qt.Vertical)
        self.MinMSESlider.setObjectName("MinMSESlider")
        self.MinMSESlider.valueChanged.connect(self.MinMSElcdUpdate)
        self.MaxMSESlider = QtWidgets.QSlider(self.centralwidget)
        self.MaxMSESlider.setGeometry(QtCore.QRect(180, 250, 60, 400))
        self.MaxMSESlider.setMinimum(0)
        self.MaxMSESlider.setMaximum(150)
        self.MaxMSESlider.setPageStep(1)
        self.MaxMSESlider.setOrientation(QtCore.Qt.Vertical)
        self.MaxMSESlider.setObjectName("MaxMSESlider")
        self.MaxMSESlider.valueChanged.connect(self.MaxMSElcdUpdate)
        self.ThSlider = QtWidgets.QSlider(self.centralwidget)
        self.ThSlider.setGeometry(QtCore.QRect(330, 250, 60, 400))
        self.ThSlider.setMinimum(0)
        self.ThSlider.setMaximum(30)
        self.ThSlider.setPageStep(1)
        self.ThSlider.setOrientation(QtCore.Qt.Vertical)
        self.ThSlider.setObjectName("ThSlider")
        self.ThSlider.valueChanged.connect(self.ThlcdUpdate)
        self.TempSlider = QtWidgets.QSlider(self.centralwidget)
        self.TempSlider.setGeometry(QtCore.QRect(480, 250, 60, 400))
        self.TempSlider.setMinimum(0)
        self.TempSlider.setMaximum(20)
        self.TempSlider.setPageStep(1)
        self.TempSlider.setOrientation(QtCore.Qt.Vertical)
        self.TempSlider.setObjectName("TempSlider")
        self.TempSlider.valueChanged.connect(self.TemplcdUpdate)
        self.FrameSlider = QtWidgets.QSlider(self.centralwidget)
        self.FrameSlider.setGeometry(QtCore.QRect(630, 250, 60, 400))
        self.FrameSlider.setMinimum(0)
        self.FrameSlider.setMaximum(200)
        self.FrameSlider.setPageStep(1)
        self.FrameSlider.setOrientation(QtCore.Qt.Vertical)
        self.FrameSlider.setObjectName("FrameSlider")
        self.FrameSlider.valueChanged.connect(self.FramelcdUpdate)
```

```
        ########## Spinners ##########
        self.nSpinner = QtWidgets.QDoubleSpinBox(self.centralwidget)
        self.nSpinner.setGeometry(QtCore.QRect(100, 150, 100, 50))
        self.nSpinner.setDecimals(1)
        self.nSpinner.setMinimum(1.0)
        self.nSpinner.setMaximum(3.0)
        self.nSpinner.setSingleStep(0.2)
        self.nSpinner.setObjectName("nSpinner")
        self.nSpinner.valueChanged.connect(self.nVal)
        self.rSpinner = QtWidgets.QDoubleSpinBox(self.centralwidget)
        self.rSpinner.setGeometry(QtCore.QRect(280, 150, 100, 50))
        self.rSpinner.setDecimals(1)
        self.rSpinner.setMinimum(1.0)
        self.rSpinner.setMaximum(4.0)
        self.rSpinner.setSingleStep(0.2)
        self.rSpinner.setObjectName("rSpinner")
        self.rSpinner.valueChanged.connect(self.rVal)



        ########## Buttons ##########
        self.RunButton = QtWidgets.QPushButton(self.centralwidget)
        self.RunButton.setGeometry(QtCore.QRect(450, 150, 120, 50))
        self.RunButton.setObjectName("RunButton")
        self.RunButton.clicked.connect(self.Run)
        self.BackgroundButton = QtWidgets.QPushButton(self.centralwidget)
        self.BackgroundButton.setGeometry(QtCore.QRect(450, 22, 120, 35))
        self.BackgroundButton.setObjectName("BackgroundButton")
        self.BackgroundButton.clicked.connect(self.Enter1)
        self.VideoButton = QtWidgets.QPushButton(self.centralwidget)
        self.VideoButton.setGeometry(QtCore.QRect(450, 85, 120, 35))
        self.VideoButton.setObjectName("VideoButton")
        self.VideoButton.clicked.connect(self.Enter2)
        self.TrackButton = QtWidgets.QPushButton(self.centralwidget)
        self.TrackButton.setGeometry(QtCore.QRect(300, 860, 100, 50))
        self.TrackButton.setObjectName("TrackButton")
        self.TrackButton.clicked.connect(self.Track)



        ########## Text Inputs ##########
        self.BackgroundInput = QtWidgets.QLineEdit(self.centralwidget)
        self.BackgroundInput.setGeometry(QtCore.QRect(200, 20, 200, 40))
        self.BackgroundInput.setObjectName("BackgroundInput")
        self.VideoInput = QtWidgets.QLineEdit(self.centralwidget)
        self.VideoInput.setGeometry(QtCore.QRect(200, 80, 200, 40))
        self.VideoInput.setObjectName("VideoInput")
```

```
        ########## Text Outputs ##########
        self.scrollArea = QtWidgets.QScrollArea(MainWindow)
        self.scrollArea.setGeometry(QtCore.QRect(1290, 530, 525, 390)) # Locat
ion and Dimensions
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setObjectName("scrollArea")
        self.scrollAreaWidgetContents = QtWidgets.QWidget()
        self.scrollAreaWidgetContents.setGeometry(QtCore.QRect(0, 0, 525, 390)
) # Location and Dimensions
        self.scrollAreaWidgetContents.setObjectName("scrollAreaWidgetContents"
)
        self.LoadingText = QtWidgets.QTextEdit(self.scrollAreaWidgetContents)
        self.LoadingText.setGeometry(QtCore.QRect(0, 0, 525, 390)) # Location
and Dimensions
        self.LoadingText.setObjectName("LoadingText")
        self.LoadingText.setFontPointSize(10)
        self.scrollArea.setWidget(self.scrollAreaWidgetContents)
        self.scrollArea2 = QtWidgets.QScrollArea(MainWindow)
        self.scrollArea2.setGeometry(QtCore.QRect(720, 530, 525, 390)) # Locat
ion and Dimensions
        self.scrollArea2.setWidgetResizable(True)
        self.scrollArea2.setObjectName("scrollArea2")
        self.scrollAreaWidgetContents2 = QtWidgets.QWidget()
        self.scrollAreaWidgetContents2.setGeometry(QtCore.QRect(0, 0, 525, 390
)) # Location and Dimensions
        self.scrollAreaWidgetContents2.setObjectName("scrollAreaWidgetContents
2")
        self.OutputText = QtWidgets.QTextEdit(self.scrollAreaWidgetContents2)
        self.OutputText.setGeometry(QtCore.QRect(0, 0, 525, 390)) # Location a
nd Dimensions
        self.OutputText.setObjectName("OutputText")
        self.OutputText.setFontPointSize(10)
        self.scrollArea2.setWidget(self.scrollAreaWidgetContents2)



        ########### Graphs ##########
        self.Displaytabs = QtWidgets.QTabWidget(self.centralwidget)
        self.Displaytabs.setGeometry(QtCore.QRect(720, 10, 1100, 500)) # Locat
ion and Dimensions
        self.Displaytabs.setObjectName("Displaytabs")
        self.PlotMSETH = QtWidgets.QWidget()
        self.layMSETH = QtWidgets.QHBoxLayout(self.PlotMSETH)
        self.Displaytabs.addTab(self.PlotMSETH, "")
        self.PlotMSEZ = QtWidgets.QWidget()
        self.layMSEZ = QtWidgets.QHBoxLayout(self.PlotMSEZ)
        self.Displaytabs.addTab(self.PlotMSEZ, "")
        self.Plot2D = QtWidgets.QWidget()
        self.lay2D = QtWidgets.QHBoxLayout(self.Plot2D)
```

```
        self.Displaytabs.addTab(self.Plot2D, "")
        self.Plot3D = QtWidgets.QWidget()
        self.lay3D = QtWidgets.QHBoxLayout(self.Plot3D)
        self.Displaytabs.addTab(self.Plot3D, "")



        ########## Setup ###########
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 18))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        self.retranslateUi(MainWindow)
        self.Displaytabs.setCurrentIndex(1)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)



    ########## Protocol To Run ##########
    # LCD events
    def MinMSElcdUpdate(self, event):
        global MinMSE
        MinMSE = event # Setting Min MSE value
        self.MinMSElcd.display(event)

    def MaxMSElcdUpdate(self, event):
        global MaxMSE
        MaxMSE = event # Setting Max MSE value
        self.MaxMSElcd.display(event)

    def ThlcdUpdate(self, event):
        global threshold
        threshold = event/200 # Setting threshold value
        self.Thlcd.display(event/200)

    def TemplcdUpdate(self, event):
        global tempsteps
        tempsteps = event # Setting template steps
        self.Templcd.display(event)

    def FramelcdUpdate(self, event):
        global nth_frame
        nth_frame = event # Setting frame steps
        self.Framelcd.display(event)
```

```
    # Spinner events
    def nVal(self, event):
        global n
        n = round(event, 1) # Value of n template to use

    def rVal(self, event):
        global r
        r = round(event, 1) # Value of r template to use



    # Button events
    def Enter1(self):
        global background
        background = str(self.BackgroundInput.text()) # Background file to use
        print(self.BackgroundInput.text())
        self.OutputText.append('Background image chosen - {}' .format(backgrou
nd))

    def Enter2(self):
        global video
        video = str(self.VideoInput.text()) # Video file to use
        print(self.VideoInput.text())
        self.OutputText.append('Video file chosen - {}' .format(video))

    def Track(self):
        answer = QtWidgets.QMessageBox.question(
            self.centralwidget, 'Lauch Tracker', 'Are you sure you are ready t
o execute Tracker from Initilisation Settings?',
            QtWidgets.QMessageBox.Yes | QtWidgets.QMessageBox.No, QtWidgets.QM
essageBox.No)
        if answer == QtWidgets.QMessageBox.Yes:#
            self.LoadingText.setFontPointSize(20)
            self.LoadingText.append('------Tracker Launched------')
            self.LoadingText.setFontPointSize(10)
            self.Tracker()

        else:
            pass

    def Run(self):
        # Attempting to collect GUI input info to run
        # Start the Loading Bar
        # Display Output Text

        self.LoadingText.setFontPointSize(20)
        self.LoadingText.append('---Initialisation Launched---')
        self.LoadingText.setFontPointSize(10)
```

```
        global background
        self.OutputText.append('Background file chosen - {}' .format(backgroun
d))
        global video
        self.OutputText.append('Video file chosen - {}' .format(video))
        global count
        global end
        end = 11
        self.OutputText.append('Number of Frames to process - {}' .format(end
- 1))
        global nth_frame
        self.OutputText.append('Steps in Frames being counted - {}' .format(nt
h_frame))
        global contrast
        self.OutputText.append('Contrast Value - 20')
        global sharpness
        self.OutputText.append('Sharpness Value - 20')
        global n
        self.OutputText.append('Template Refractive index - {}' .format(n))
        global r
        self.OutputText.append('Template sphere radius - {}' .format(r))
        global Location
        global threshold
        threshold = 0.04
        self.OutputText.append('Threshold Value - {}' .format(threshold))
        global tempsteps
        tempsteps = 5
        self.OutputText.append('Steps in Templates being counted - {}' .format
(tempsteps))
        global tempcount
        global tempend
        global brightness
        global MinMSE
        global MaxMSE
        global PlotZ
        global PlotTH
        global PlotMSE



        ########## Background and Video Processing ##########
        bgd = cv2.imread('..Backgrounds/{}.png' .format(background)) # Locatio
n of background image
        bgdgray = cv2.cvtColor(bgd, cv2.COLOR_BGR2GRAY)
        arraybgdgray = np.array(bgdgray)
        vidcap = cv2.VideoCapture('..Videos/{}.avi' .format(video)) # Location
 of video
```

```
        ######### Create Frames #########
        while(count < end):
            tic = time.perf_counter() # Estimating run time starter

            # Capture frame-by-frame
            skip = nth_frame * count
            vidcap.set(1, skip)
            ret, frame = vidcap.read()

            self.LoadingText.append('Reading frame %d' % skip)

            # Operations on the frames come here
            imggray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # BGR or RGB (ca
st different readings)
            array1 = np.array(imggray)
            arrayDIV = np.divide(array1, arraybgdgray) * brightness
            divided = Image.fromarray(arrayDIV)
            final = divided.convert("L")

            # Image contraster
            enhancer1 = ImageEnhance.Contrast(final)
            enhanced1 = enhancer1.enhance(contrast)
            enhancer2 = ImageEnhance.Sharpness(enhanced1)
            enhanced2 = enhancer2.enhance(sharpness)

            # Save the resulting frame
            enhanced2.save("enhanced frames/frame%d.png" % count)
            self.LoadingText.append('Processed frame %d' % count)

            # Matching process
            img_rgb = cv2.imread("enhanced frames/frame%d.png" % count)
            img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)
            img_to_crop = Image.open("enhanced frames/frame%d.png" % count).co
nvert('L')

            while(tempcount < tempend):
                template = cv2.imread("templates/0.703 pix-
um/n={}, r={}/img{}.png" .format(n, r, tempcount), 0) # 0.711 pix-um/
                w, h = template.shape[::-1]
                templatePIL = Image.open("templates/0.703 pix-
um/n={}, r={}/img{}.png" .format(n, r, tempcount)).convert('L') # 0.711 pix-
um/
                w, h = templatePIL.size

                res = cv2.matchTemplate(img_gray,template,cv2.TM_CCOEFF_NORMED
)
                min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
                loc = np.where(res >= threshold)
```

```
                for pt in zip(*loc[::-1]):
                    cv2.circle(img_rgb, (pt[0] + int(h/2), pt[1] + int(h/2)),
10, (0,0,255), 1)

                    # Crop matched image to size of template for MSE
                    crop = img_to_crop.crop((pt[0], pt[1], pt[0] + w, pt[1] +
h))

                    crop = crop.save("cropped frames/crop f={}, s={}.png" .for
mat(count, tempcount))
                    crop = Image.open("cropped frames/crop f={}, s={}.png" .fo
rmat(count, tempcount))

                    def mse(actual, pred):
                        actual, pred = np.array(actual), np.array(pred)
                        return np.square(np.subtract(actual,pred)).mean()

                    mse_val = mse(templatePIL, crop)

                    plot3 = [[(tempcount) for col in range(1)] for row in rang
e(1)] # Z coordinates
                    plot4 = [[(max_val) for col in range(1)] for row in range(
1)] # Threshold
                    plot5 = [[(mse_val) for col in range(1)] for row in range(
1)] # MSE

                    PlotZ = PlotZ + plot3
                    PlotTH = PlotTH + plot4
                    PlotMSE = PlotMSE + plot5

                tempcount += tempsteps

            # Save the resulting matched frame
            cv2.imwrite("matched frames/frame%d.png" % count,img_rgb)

            toc = time.perf_counter() # Estimating run time stopper

            self.LoadingText.append('Matched frame %d \n' % count)
            self.LoadingText.append(f"Estimated Time Remaining: {int((toc - ti
c)*((end - 1) - count))} seconds")

            tempcount = 1
            count += 1
            self.progressBar.setValue((count/(end-
1))*100) # LoadingBar Protocol

        # Reset counter for multiple plotting events in singluar window
        count = 1
```

```
        self.LoadingText.setFontPointSize(15)
        self.LoadingText.append('\nAll Frames Created \nLoading Plots.......')
        self.LoadingText.setFontPointSize(12)

        # Arrays for Plots
        z = PlotZ
        th = PlotTH
        mse = PlotMSE

        # Output Data
        Matchval = np.mean(th)
        SDev = np.std(th)
        UDep = len(np.unique(z))
        RDep = len(z) - UDep
        MSEval = np.mean(mse)
        RecMinMSE = int(np.percentile(mse, 10))
        RecMaxMSE = int(np.percentile(mse, 90))
        RecMinTH = round(np.percentile(th, 10), 2)



        ######### MSE-TH plotter #########
        fig6 = plt.figure()

        self.canvas = FigureCanvas(fig6)
        self.axes = fig6.add_subplot()

        self.axes.axvline(RecMinMSE, color='b')
        self.axes.axvline(RecMaxMSE, color='b')
        self.axes.axhline(RecMinTH, color='b')

        self.axes.axvspan(RecMaxMSE, (np.max(mse)*1.05), facecolor='0.5', alph
a = 0.5)
        self.axes.axvspan(RecMinMSE, (np.min(mse)*0.9), facecolor='0.5', alpha
 = 0.5)
        self.axes.add_patch(Rectangle((RecMinMSE, (np.min(th)*0.9)), (RecMaxMS
E-RecMinMSE), (RecMinTH-(np.min(th)*0.9)), facecolor='0.5', alpha = 0.5))

        pnt = self.axes.scatter(mse, th, c=z, s=2)
        cbar = plt.colorbar(pnt)
        cbar.set_label("Z Axis")
        self.axes.set_title('n = {}, r = {}' .format(n, r))
        self.axes.set_xlabel('MSE Value')
        self.axes.set_ylabel('Match Value')
        self.axes.set_xlim((np.min(mse)*0.9), (np.max(mse)*1.05))
        self.axes.set_ylim((np.min(th)*0.9), (np.max(th)*1.05))
        self.axes.scatter(mse, th, c=z, s=2)
        self.axes.set_axisbelow(True)
        self.axes.minorticks_on()
```

```
        self.axes.grid(which='major', linestyle='-
', linewidth='0.5', color='red')
        self.axes.grid(which='minor', linestyle='-
', linewidth='0.1', color='black')

        plt.savefig('.MSE-TH Plots/MSE-Th(1) n={}, r={}.png' .format(n, r))
        self.layMSETH.addWidget(FigureCanvas(fig6))
        self.LoadingText.append('\nMSE-Th(1) Plot Created and saved')



        ######### Z-MSE plotter #########
        fig5, ax5 = plt.subplots()

        ax5.axhline(RecMinMSE, color='m')
        ax5.axhline(RecMaxMSE, color='m')
        ax5.plot(z, mse, '.', color='red', markersize=2)
        ax5.set_title('n = {}, r = {}, Average={}' .format(n, r, MSEval))
        ax5.axhspan(RecMaxMSE, (np.max(mse)*1.05), facecolor='0.5', alpha = 0.
5)
        ax5.axhspan(RecMinMSE, (np.min(mse)*0.9), facecolor='0.5', alpha = 0.5
)
        ax5.set_xlabel('Z Axis')
        ax5.set_ylabel('MSE Value')
        ax5.set_axisbelow(True)
        ax5.set_ylim((np.min(mse)*0.9), (np.max(mse)*1.05))
        ax5.minorticks_on()
        ax5.grid(which='major', linestyle='-', linewidth='0.5', color='blue')
        ax5.grid(which='minor', linestyle='-', linewidth='0.1', color='black')

        plt.savefig('.MSE-Z Plots/MSE-Z(1) n={}, r={}.png' .format(n, r))
        self.layMSEZ.addWidget(FigureCanvas(fig5))
        self.LoadingText.append('\nZ-MSE(1) Plot Created and saved\n')
        self.LoadingText.setFontPointSize(10)



        ######### Print Specifics #########
        self.OutputText.setFontPointSize(20)
        self.OutputText.append('\n \n------INPUT SPECIFICS-----')
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Frames Settings**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Start - %d, End - %d , Total - %d, Steps - %d
\n' % (nth_frame, nth_frame*(end-1), (end-1), nth_frame))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Enhancement Settings**')
        self.OutputText.setFontPointSize(10)
```

```
        self.OutputText.append('Contrast - %d, Sharpness - %d \n' % (contrast,
 sharpness))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Template Settings**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Folder - n={}, r={}, Threshold - {}, Steps - {
} \n \n' .format(n, r, threshold, tempsteps))
        self.OutputText.setFontPointSize(20)

        self.OutputText.append('-----OUTPUT SPECIFICS----')
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Matching Data**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Average Match Value - {}, S.Dev of Match Value
 - {}, Recommended Min Match Value - {} \n' .format(Matchval, SDev, RecMinTH))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Depth Data**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Total Number of Unique Depths - {}, Total Numb
er of Repeating Depths - {} \n' .format(UDep, RDep))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**MSE Data**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Average MSE Value - {}, Recommended Min MSE -
{}, Recommended Max MSE - {} \n' .format(MSEval, RecMinMSE, RecMaxMSE))

        PlotZ = []
        PlotTH = []
        PlotMSE = []

    def Tracker(self):
        # Attempting to collect GUI input info to run
        # Start the Loading Bar
        # Display Output Text

        global background
        global video
        global count
        global nth_frame
        global end
        end = int(10000/nth_frame) + 1 # Assuming 10,000 frames in video
        self.OutputText.append('Number of Frames to process - {}' .format(end
- 1))
        self.OutputText.append('Steps in Frames being counted - {}' .format(nt
h_frame))
        global contrast
        self.OutputText.append('Contrast Value - {}' .format(contrast))
        global sharpness
```

```
        self.OutputText.append('Sharpness Value - {}' .format(sharpness))
        global n
        self.OutputText.append('Template Refractive index - {}' .format(n))
        global r
        self.OutputText.append('Template sphere radius - {}' .format(r))
        global Location
        global threshold
        self.OutputText.append('Threshold Value - {}' .format(threshold))
        global tempsteps
        self.OutputText.append('Steps in Templates being counted - {}' .format
(tempsteps))
        global tempcount
        global tempend
        global brightness
        global MinMSE
        global MaxMSE
        global PlotX
        global PlotY
        global PlotZ
        global PlotTH
        global PlotMSE
        global PlotCOUNT



        ########## Background and Video Processing ##########
        bgd = cv2.imread('..Backgrounds/{}.png' .format(background)) # Locatio
n of background image
        bgdgray = cv2.cvtColor(bgd, cv2.COLOR_BGR2GRAY)
        arraybgdgray = np.array(bgdgray)
        vidcap = cv2.VideoCapture('..Videos/{}.avi' .format(video)) # Location
 of video



        ######### Create Frames #########
        while(count < end):
            tic = time.perf_counter() # Estimating run time starter

            # Capture frame-by-frame
            skip = nth_frame * count
            vidcap.set(1, skip)
            ret, frame = vidcap.read()

            self.LoadingText.append('Reading frame %d' % skip)

            # Operations on the frames come here
            imggray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            array1 = np.array(imggray)
```

```
            arrayDIV = np.divide(array1, arraybgdgray) * brightness
            divided = Image.fromarray(arrayDIV)
            final = divided.convert("L")

            # Image contraster
            enhancer1 = ImageEnhance.Contrast(final)
            enhanced1 = enhancer1.enhance(contrast)
            enhancer2 = ImageEnhance.Sharpness(enhanced1)
            enhanced2 = enhancer2.enhance(sharpness)

            # Save the resulting frame
            enhanced2.save("enhanced frames/frame%d.png" % count)

            self.LoadingText.append('Processed frame %d' % count)

            # Matching process
            img_rgb = cv2.imread("enhanced frames/frame%d.png" % count)
            img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)
            img_to_crop = Image.open("enhanced frames/frame%d.png" % count).co
nvert('L')

            while(tempcount < tempend):
                template = cv2.imread("templates/0.703 pix-
um/n={}, r={}/img{}.png" .format(n, r, tempcount), 0) # 0.711 pix-um/
                w, h = template.shape[::-1]
                templatePIL = Image.open("templates/0.703 pix-
um/n={}, r={}/img{}.png" .format(n, r, tempcount)).convert('L') # 0.711 pix-
um/
                w, h = templatePIL.size

                res = cv2.matchTemplate(img_gray,template,cv2.TM_CCOEFF_NORMED
)
                min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
                loc = np.where(res >= threshold)
                for pt in zip(*loc[::-1]):
                    cv2.circle(img_rgb, (pt[0] + int(h/2), pt[1] + int(h/2)),
10, (0,0,255), 1)

                    # Crop matched image to size of template for MSE
                    crop = img_to_crop.crop((pt[0], pt[1], pt[0] + w, pt[1] +
h))

                    crop = crop.save("cropped frames/crop f={}, s={}.png" .for
mat(count, tempcount))
                    crop = Image.open("cropped frames/crop f={}, s={}.png" .fo
rmat(count, tempcount))

                    def mse(actual, pred):
```

```
                        actual, pred = np.array(actual), np.array(pred)
                        return np.square(np.subtract(actual,pred)).mean()

                    mse_val = mse(templatePIL, crop)


                    if (mse_val > MinMSE) and (mse_val < MaxMSE):
                        plot1 = [[(pt[0] + int(h/2)) for col in range(1)] for
row in range(1)] # X coordinates
                        plot2 = [[(pt[1] + int(h/2)) for col in range(1)] for
row in range(1)] # Y coordinates
                        plot3 = [[(tempcount) for col in range(1)] for row in
range(1)] # Z coordinates
                        plot4 = [[(max_val) for col in range(1)] for row in ra
nge(1)] # Threshold
                        plot5 = [[(mse_val) for col in range(1)] for row in ra
nge(1)] # MSE
                        plot6 = [[(count) for col in range(1)] for row in rang
e(1)] # Frame Counts

                        PlotX = PlotX + plot1
                        PlotY = PlotY + plot2
                        PlotZ = PlotZ + plot3
                        PlotTH = PlotTH + plot4
                        PlotMSE = PlotMSE + plot5
                        PlotCOUNT = PlotCOUNT + plot6

                tempcount += tempsteps

            # Save the resulting matched frame
            cv2.imwrite("matched frames/frame%d.png" % count,img_rgb)

            toc = time.perf_counter() # Estimating run time stopper

            self.LoadingText.append('Matched frame %d \n' % count)
            self.LoadingText.append(f"Estimated Time Remaining: {int((toc - ti
c)*((end - 1) - count))} seconds")

            tempcount = 1
            count += 1
            self.progressBar.setValue((count/(end-
1))*100) # LoadingBar Protocol

        # Reset counter for multiple plotting events in singluar window
        count = 1
        self.LoadingText.setFontPointSize(15)
        self.LoadingText.append('\nAll Frames Created \nLoading Plots.......')
        self.LoadingText.setFontPointSize(12)
```

```
        # Arrays for Plots
        x = PlotX
        y = PlotY
        z = PlotZ
        th = PlotTH
        mse = PlotMSE
        tracking = PlotCOUNT

        # Output Data
        UDep = len(np.unique(z))
        RDep = len(z) - UDep
        UX = len(np.unique(x))
        UY = len(np.unique(y))
        MSEval = np.mean(mse)



        ######### MSE-TH plotter #########
        fig6 = plt.figure()

        self.canvas = FigureCanvas(fig6)
        self.axes = fig6.add_subplot()

        pnt = self.axes.scatter(mse, th, c=z, s=2)
        cbar = plt.colorbar(pnt)
        cbar.set_label("Z Axis")

        self.axes.set_title('n = {}, r = {}' .format(n, r))
        self.axes.set_xlabel('MSE Value')
        self.axes.set_ylabel('Match Value')
        self.axes.scatter(mse, th, c=z, s=2)

        plt.savefig('.MSE-TH Plots/MSE-Th(2) n={}, r={}.png' .format(n, r))
        self.layMSETH.addWidget(FigureCanvas(fig6))
        self.LoadingText.append('\nMSE-Th(2) Plot Created and saved')



        ######### Z-MSE plotter #########
        fig5, ax5 = plt.subplots()

        ax5.plot(z, mse, '.', color='red', markersize=2)
        ax5.set_title('n = {}, r = {}, Average={}' .format(n, r, MSEval))
        ax5.set_xlabel('Z Axis')
        ax5.set_ylabel('MSE Value')

        plt.savefig('.MSE-Z Plots/MSE-Z(2) n={}, r={}.png' .format(n, r))
        self.layMSEZ.addWidget(FigureCanvas(fig5))
        self.LoadingText.append('\nZ-MSE(2) Plot Created and saved\n')
```

```
        ######### 2D plotter #########
        fig1 = plt.figure()

        self.canvas1 = FigureCanvas(fig1)
        self.axes1 = fig1.add_subplot()

        pnt = self.axes1.scatter(mse, th, c=z, s=2)
        cbar = plt.colorbar(pnt)
        cbar.set_label("Z Axis")

        self.axes1.set_title('n={}, r={}, f={}, s={}' .format(n, r, end-
1, tempsteps))
        self.axes1.set_xlabel('X Axis')
        self.axes1.set_ylabel('Y Axis')
        self.axes1.scatter(x, y, c=z, s=2)

        plt.savefig('.2D Plots/2D n={}, r={}, f={}, s={}.png' .format(n, r, en
d-1, tempsteps))
        self.lay2D.addWidget(FigureCanvas(fig1))
        self.LoadingText.append('\n2D Plot Created and saved')



        ######### 3D plotter #########
        fig7 = plt.figure()

        self.canvas = FigureCanvas(fig7)
        self.axes = fig7.add_subplot(111, projection='3d')

        pnt3d = self.axes.scatter(x, y, z, c=tracking, s=1)
        pnt3d.set_clim(np.min(tracking), np.max(tracking))
        cbar = plt.colorbar(pnt3d)
        cbar.set_label("Frame Number")

        self.axes.set_title('n={}, r={}, f={}, s={}.png' .format(n, r, end-
1, tempsteps))
        self.axes.set_xlabel('X Axis')
        self.axes.set_ylabel('Y Axis')
        self.axes.set_zlabel('Z Axis')
        self.axes.set_zlim([0, 220])
        self.axes.scatter(x, y, z, c=tracking, s=1)

        plt.savefig('.3D Plots/3D n={}, r={}, f={}, s={}.png' .format(n, r, en
d-1, tempsteps))
        self.lay3D.addWidget(self.canvas)
        self.LoadingText.append('\n3D Plot Created and saved \n')
```

```
        ########## Saving DATA files ###########
        np.savetxt('.Data files/X Data/n={}, r={}, f={}, s={}.txt' .format(n,
r, end-1, tempsteps), x)
        np.savetxt('.Data files/Y Data/n={}, r={}, f={}, s={}.txt' .format(n,
r, end-1, tempsteps), y)
        np.savetxt('.Data files/Z Data/n={}, r={}, f={}, s={}.txt' .format(n,
r, end-1, tempsteps), z)
        np.savetxt('.Data files/MSE Data/co={}, sh={}, th={}.txt' .format(cont
rast, sharpness, threshold), mse)

        self.LoadingText.append('All data points saved')
        self.LoadingText.setFontPointSize(10)



        ######### Print Specifics #########
        self.OutputText.setFontPointSize(20)
        self.OutputText.append('\n \n------INPUT SPECIFICS-----')
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Frames Settings**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Start - %d, End - %d , Total - %d, Steps - %d
\n' % (nth_frame, nth_frame*(end-1), (end-1), nth_frame))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Enhancement Settings**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Contrast - %d, Sharpness - %d \n' % (contrast,
 sharpness))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Template Settings**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Folder - n={}, r={}, Threshold - {}, Steps - {
} \n \n' .format(n, r, threshold, tempsteps))
        self.OutputText.setFontPointSize(20)

        self.OutputText.append('-----OUTPUT SPECIFICS----')
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Depth Data**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Total Number of Unique Depths - {}, Total Numb
er of Repeating Depths - {} \n' .format(UDep, RDep))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**MSE Data**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Average MSE Value - {} \n' .format(MSEval))
        self.OutputText.setFontPointSize(15)
```

```
        self.OutputText.append('**X-Y Data**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Total Number of Unique X Values - {}, Total Nu
mber of Unique Y Values - {} \n \n' .format(UX, UY))


    ########## Window Setup ##########
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "The Hologram Analy
sis System 2 [THAS2] (created by I.T.Booth)"))
        self.BackgroundLabel.setText(_translate("MainWindow", "Background"))
        self.VideoLabel.setText(_translate("MainWindow", "Video"))
        self.label_3.setText(_translate("MainWindow", "Min MSE"))
        self.label_4.setText(_translate("MainWindow", "Max MSE"))
        self.label_5.setText(_translate("MainWindow", "Threshold"))
        self.label_6.setText(_translate("MainWindow", "Template Steps"))
        self.label_7.setText(_translate("MainWindow", "Frame Steps"))
        self.nLabel.setText(_translate("MainWindow", "n ="))
        self.rLabel.setText(_translate("MainWindow", "r ="))

        self.RunButton.setText(_translate("MainWindow", "Initialise"))
        self.BackgroundButton.setText(_translate("MainWindow", "Enter"))
        self.VideoButton.setText(_translate("MainWindow", "Enter"))
        self.TrackButton.setText(_translate("MainWindow", "Track"))

        self.Displaytabs.setTabText(self.Displaytabs.indexOf(self.PlotMSEZ), _
translate("MainWindow", "Z-MSE"))
        self.Displaytabs.setTabText(self.Displaytabs.indexOf(self.PlotMSETH),
_translate("MainWindow", "MSE-Th (Z)"))
        self.Displaytabs.setTabText(self.Displaytabs.indexOf(self.Plot2D), _tr
anslate("MainWindow", "2D (Z)"))
        self.Displaytabs.setTabText(self.Displaytabs.indexOf(self.Plot3D), _tr
anslate("MainWindow", "3D"))


########## Execute above protocol ##########
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    app.setStyle('Fusion')
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
```

```
    sys.exit(app.exec_())
```

Appendix 6: The Python code for The Hologram Analysis System (updated), including in-depth explanations for every section of the code for clarity on how it works

```
#
       #
#
       #
#                                Created by I.T.Booth
       #
# ---------------------------------------------------------------------------
------ #
#
       #
#                          The Hologram Analysis System 3
       #
# ---------------------------------------------------------------------------
------ #
#                                     (THAS3)
       #
#
       #
# Attempts to match video frames against an extensive library of synthetic tem
plates #
# Specific properties about the templates and the frames themselves can be cho
sen    #
# Once chosen, the program performs several operations producing plots of inte
rest   #
# Displaying and saving the created DATA for later analysis by user if necessa
ry     #
#
       #
#
       #



########## Program Import Requirements ###########
import sys
import time
from PyQt5 import (QtCore, QtGui, QtWidgets)
from PyQt5.QtCore import QProcess
from PyQt5.QtWidgets import (QWidget, QMessageBox, QLabel, QHBoxLayout, QAppli
cation)
import numpy as np
from numpy import arange
import PIL
```

```
from PIL import Image, ImageEnhance, ImageFilter
import matplotlib
from matplotlib.patches import Rectangle
from matplotlib import pyplot
import matplotlib.pylab as plt
from matplotlib.backends.backend_qt5agg import FigureCanvas
from matplotlib.backends.backend_qt5agg import NavigationToolbar2QT as Navigat
ionToolbar
from math import sqrt,exp
import cv2
from mpl_toolkits.mplot3d import Axes3D
import array as arr
import keyboard
from pandas import read_csv
from scipy.optimize import curve_fit
from mayavi import mlab
from mayavi.modules.glyph import Glyph
matplotlib.use('QT5Agg')



########## Autorun callable settings ##########
background = "Back1" # Background image file name
video = "Vid1" # Video file name
videoframes = 10000 # Number of frames in video
count = 1 ## NON ALTERABLE ##
end = 11 # Number of frames to process (10) +1
nth_frame = 1000 # Taking frames every nth step (x/(end-
1)) where x=total frames
contrast = 70 # Enhancing frame contrast
sharpness = 70 # Enhancing frame sharpness
brightness = 250 # Sets pixel array multiplier
tempcount = 1 # Starting template
tempend = 201 # Ending template
tempsteps = 5 # Taking templates every nth step
n = 1.0 # Synthetic template refractive index
r = 1.0 # Synthetic template radius of sphere
Location = ("n={}, r={}" .format(n, r)) ## NON ALTERABLE ##
threshold = 0.07 # Threshold value for template matching
MinMSE = 0 # Sets minimum MSE value acceptable
MaxMSE = 0 # Sets maximum MSE value acceptable

PlotX = [] ## NON ALTERABLE ##
PlotY = [] ## NON ALTERABLE ##
PlotZ = [] ## NON ALTERABLE ##
PlotTH = [] ## NON ALTERABLE ##
PlotMSE = [] ## NON ALTERABLE ##
PlotCOUNT = [] ## NON ALTERABLE ##
```

```
########## Main Setup ###########
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("ANALYSER")
        MainWindow.resize(1900, 950) # Initial Window Dimensions
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("Icon.ico"), QtGui.QIcon.Selected, QtGui.
QIcon.On)
        MainWindow.setWindowIcon(icon)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")



        ########## Labels ##########
        self.BackgroundLabel = QtWidgets.QLabel(self.centralwidget)
        self.BackgroundLabel.setGeometry(QtCore.QRect(80, 27, 100, 20))
        self.BackgroundLabel.setObjectName("BackgroundLabel")
        self.VideoLabel = QtWidgets.QLabel(self.centralwidget)
        self.VideoLabel.setGeometry(QtCore.QRect(100, 87, 100, 20))
        self.VideoLabel.setObjectName("VideoLabel")
        self.nLabel = QtWidgets.QLabel(self.centralwidget)
        self.nLabel.setGeometry(QtCore.QRect(65, 160, 20, 20))
        self.nLabel.setObjectName("nLabel")
        self.rLabel = QtWidgets.QLabel(self.centralwidget)
        self.rLabel.setGeometry(QtCore.QRect(250, 160, 20, 20))
        self.rLabel.setObjectName("rLabel")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setGeometry(QtCore.QRect(30, 660, 100, 20))
        self.label_3.setObjectName("Min MSE")
        self.label_4 = QtWidgets.QLabel(self.centralwidget)
        self.label_4.setGeometry(QtCore.QRect(170, 660, 100, 20))
        self.label_4.setObjectName("Max MSE")
        self.label_5 = QtWidgets.QLabel(self.centralwidget)
        self.label_5.setGeometry(QtCore.QRect(325, 660, 100, 20))
        self.label_5.setObjectName("Threshold")
        self.label_6 = QtWidgets.QLabel(self.centralwidget)
        self.label_6.setGeometry(QtCore.QRect(460, 660, 120, 20))
        self.label_6.setObjectName("Template Steps")
        self.label_7 = QtWidgets.QLabel(self.centralwidget)
        self.label_7.setGeometry(QtCore.QRect(620, 660, 120, 20))
        self.label_7.setObjectName("Frame Steps")
```

```
        ########## Loading Bar ##########
        self.progressBar = QtWidgets.QProgressBar(self.centralwidget)
        self.progressBar.setGeometry(QtCore.QRect(25, 780, 670, 60))
        self.progressBar.setProperty("value", 0)
        self.progressBar.setObjectName("progressBar")



        ########## LCDs ##########
        self.MinMSElcd = QtWidgets.QLCDNumber(self.centralwidget)
        self.MinMSElcd.setGeometry(QtCore.QRect(25, 700, 70, 40))
        self.MinMSElcd.display(0)
        self.MinMSElcd.setObjectName("MinMSElcd")
        self.MaxMSElcd = QtWidgets.QLCDNumber(self.centralwidget)
        self.MaxMSElcd.setGeometry(QtCore.QRect(175, 700, 70, 40))
        self.MaxMSElcd.display(0)
        self.MaxMSElcd.setObjectName("MaxMSElcd")
        self.Thlcd = QtWidgets.QLCDNumber(self.centralwidget)
        self.Thlcd.setGeometry(QtCore.QRect(325, 700, 70, 40))
        self.Thlcd.display(0)
        self.Thlcd.setObjectName("Thlcd")
        self.Templcd = QtWidgets.QLCDNumber(self.centralwidget)
        self.Templcd.setGeometry(QtCore.QRect(475, 700, 70, 40))
        self.Templcd.display(0)
        self.Templcd.setObjectName("Templcd")
        self.Framelcd = QtWidgets.QLCDNumber(self.centralwidget)
        self.Framelcd.setGeometry(QtCore.QRect(625, 700, 70, 40))
        self.Framelcd.display(0)
        self.Framelcd.setObjectName("Framelcd")



        ########## Sliders ##########
        self.MinMSESlider = QtWidgets.QSlider(self.centralwidget)
        self.MinMSESlider.setGeometry(QtCore.QRect(30, 250, 60, 400))
        self.MinMSESlider.setMinimum(0)
        self.MinMSESlider.setMaximum(150)
        self.MinMSESlider.setPageStep(1)
        self.MinMSESlider.setOrientation(QtCore.Qt.Vertical)
        self.MinMSESlider.setObjectName("MinMSESlider")
        self.MinMSESlider.valueChanged.connect(self.MinMSElcdUpdate)
        self.MaxMSESlider = QtWidgets.QSlider(self.centralwidget)
        self.MaxMSESlider.setGeometry(QtCore.QRect(180, 250, 60, 400))
        self.MaxMSESlider.setMinimum(0)
        self.MaxMSESlider.setMaximum(150)
        self.MaxMSESlider.setPageStep(1)
        self.MaxMSESlider.setOrientation(QtCore.Qt.Vertical)
        self.MaxMSESlider.setObjectName("MaxMSESlider")
        self.MaxMSESlider.valueChanged.connect(self.MaxMSElcdUpdate)
```

```
        self.ThSlider = QtWidgets.QSlider(self.centralwidget)
        self.ThSlider.setGeometry(QtCore.QRect(330, 250, 60, 400))
        self.ThSlider.setMinimum(0)
        self.ThSlider.setMaximum(30)
        self.ThSlider.setPageStep(1)
        self.ThSlider.setOrientation(QtCore.Qt.Vertical)
        self.ThSlider.setObjectName("ThSlider")
        self.ThSlider.valueChanged.connect(self.ThlcdUpdate)
        self.TempSlider = QtWidgets.QSlider(self.centralwidget)
        self.TempSlider.setGeometry(QtCore.QRect(480, 250, 60, 400))
        self.TempSlider.setMinimum(0)
        self.TempSlider.setMaximum(5)
        self.TempSlider.setPageStep(1)
        self.TempSlider.setOrientation(QtCore.Qt.Vertical)
        self.TempSlider.setObjectName("TempSlider")
        self.TempSlider.valueChanged.connect(self.TemplcdUpdate)
        self.FrameSlider = QtWidgets.QSlider(self.centralwidget)
        self.FrameSlider.setGeometry(QtCore.QRect(630, 250, 60, 400))
        self.FrameSlider.setMinimum(0)
        self.FrameSlider.setMaximum(200)
        self.FrameSlider.setPageStep(1)
        self.FrameSlider.setOrientation(QtCore.Qt.Vertical)
        self.FrameSlider.setObjectName("FrameSlider")
        self.FrameSlider.valueChanged.connect(self.FramelcdUpdate)



        ########## Spinners ##########
        self.nSpinner = QtWidgets.QDoubleSpinBox(self.centralwidget)
        self.nSpinner.setGeometry(QtCore.QRect(100, 150, 100, 50))
        self.nSpinner.setDecimals(1)
        self.nSpinner.setMinimum(1.0)
        self.nSpinner.setMaximum(3.0)
        self.nSpinner.setSingleStep(0.2)
        self.nSpinner.setObjectName("nSpinner")
        self.nSpinner.valueChanged.connect(self.nVal)
        self.rSpinner = QtWidgets.QDoubleSpinBox(self.centralwidget)
        self.rSpinner.setGeometry(QtCore.QRect(280, 150, 100, 50))
        self.rSpinner.setDecimals(1)
        self.rSpinner.setMinimum(1.0)
        self.rSpinner.setMaximum(2.8)
        self.rSpinner.setSingleStep(0.2)
        self.rSpinner.setObjectName("rSpinner")
        self.rSpinner.valueChanged.connect(self.rVal)



        ########## Buttons ##########
        self.RunButton = QtWidgets.QPushButton(self.centralwidget)
```

```
        self.RunButton.setGeometry(QtCore.QRect(450, 150, 120, 50))
        self.RunButton.setObjectName("RunButton")
        self.RunButton.setStyleSheet('QPushButton {background-
color: #9e3920; color: blue;}')
        self.RunButton.clicked.connect(self.Run)
        self.BackgroundButton = QtWidgets.QPushButton(self.centralwidget)
        self.BackgroundButton.setGeometry(QtCore.QRect(450, 22, 120, 35))
        self.BackgroundButton.setObjectName("BackgroundButton")
        self.BackgroundButton.clicked.connect(self.Enter1)
        self.VideoButton = QtWidgets.QPushButton(self.centralwidget)
        self.VideoButton.setGeometry(QtCore.QRect(450, 85, 120, 35))
        self.VideoButton.setObjectName("VideoButton")
        self.VideoButton.clicked.connect(self.Enter2)
        self.TrackButton = QtWidgets.QPushButton(self.centralwidget)
        self.TrackButton.setGeometry(QtCore.QRect(300, 860, 100, 50))
        self.TrackButton.setObjectName("TrackButton")
        self.TrackButton.setStyleSheet('QPushButton {background-
color: #9e3920; color: blue;}')
        self.TrackButton.clicked.connect(self.Track)
        self.PlotButton = QtWidgets.QPushButton(self.centralwidget)
        self.PlotButton.setGeometry(QtCore.QRect(450, 860, 150, 50))
        self.PlotButton.setObjectName("PlotButton")
        self.PlotButton.setStyleSheet('QPushButton {background-
color: #A3C1DA; color: blue;}')
        self.PlotButton.clicked.connect(self.Plot)



        ########## Text Inputs ##########
        self.BackgroundInput = QtWidgets.QLineEdit(self.centralwidget)
        self.BackgroundInput.setGeometry(QtCore.QRect(200, 20, 200, 40))
        self.BackgroundInput.setObjectName("BackgroundInput")
        self.VideoInput = QtWidgets.QLineEdit(self.centralwidget)
        self.VideoInput.setGeometry(QtCore.QRect(200, 80, 200, 40))
        self.VideoInput.setObjectName("VideoInput")



        ########## Text Outputs ##########
        self.scrollArea = QtWidgets.QScrollArea(MainWindow)
        self.scrollArea.setGeometry(QtCore.QRect(1290, 530, 525, 390)) # Locat
ion and Dimensions
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setObjectName("scrollArea")
        self.scrollAreaWidgetContents = QtWidgets.QWidget()
        self.scrollAreaWidgetContents.setGeometry(QtCore.QRect(0, 0, 525, 390)
) # Location and Dimensions
        self.scrollAreaWidgetContents.setObjectName("scrollAreaWidgetContents"
)
```

```
        self.LoadingText = QtWidgets.QTextEdit(self.scrollAreaWidgetContents)
        self.LoadingText.setGeometry(QtCore.QRect(0, 0, 525, 390)) # Location
and Dimensions
        self.LoadingText.setObjectName("LoadingText")
        self.LoadingText.setFontPointSize(10)
        self.scrollArea.setWidget(self.scrollAreaWidgetContents)
        self.scrollArea2 = QtWidgets.QScrollArea(MainWindow)
        self.scrollArea2.setGeometry(QtCore.QRect(720, 530, 525, 390)) # Locat
ion and Dimensions
        self.scrollArea2.setWidgetResizable(True)
        self.scrollArea2.setObjectName("scrollArea2")
        self.scrollAreaWidgetContents2 = QtWidgets.QWidget()
        self.scrollAreaWidgetContents2.setGeometry(QtCore.QRect(0, 0, 525, 390
)) # Location and Dimensions
        self.scrollAreaWidgetContents2.setObjectName("scrollAreaWidgetContents
2")
        self.OutputText = QtWidgets.QTextEdit(self.scrollAreaWidgetContents2)
        self.OutputText.setGeometry(QtCore.QRect(0, 0, 525, 390)) # Location a
nd Dimensions
        self.OutputText.setObjectName("OutputText")
        self.OutputText.setFontPointSize(10)
        self.scrollArea2.setWidget(self.scrollAreaWidgetContents2)



        ########### Graphs ##########
        self.Displaytabs = QtWidgets.QTabWidget(self.centralwidget)
        self.Displaytabs.setGeometry(QtCore.QRect(720, 10, 1100, 500)) # Locat
ion and Dimensions
        self.Displaytabs.setObjectName("Displaytabs")
        self.PlotMSETH = QtWidgets.QWidget()
        self.layMSETH = QtWidgets.QHBoxLayout(self.PlotMSETH)
        self.Displaytabs.addTab(self.PlotMSETH, "")
        self.PlotMSEZ = QtWidgets.QWidget()
        self.layMSEZ = QtWidgets.QHBoxLayout(self.PlotMSEZ)
        self.Displaytabs.addTab(self.PlotMSEZ, "")
        self.PlotTHZ = QtWidgets.QWidget()
        self.layTHZ = QtWidgets.QHBoxLayout(self.PlotTHZ)
        self.Displaytabs.addTab(self.PlotTHZ, "")
        self.Plot2D = QtWidgets.QWidget()
        self.lay2D = QtWidgets.QHBoxLayout(self.Plot2D)
        self.Displaytabs.addTab(self.Plot2D, "")
        self.Plot3D = QtWidgets.QWidget()
        self.lay3D = QtWidgets.QHBoxLayout(self.Plot3D)
        self.Displaytabs.addTab(self.Plot3D, "")



        ########## Setup ###########
```

```
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 18))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        self.retranslateUi(MainWindow)
        self.Displaytabs.setCurrentIndex(1)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)



    ########## Protocol To Run ##########
    # LCD events
    def MinMSElcdUpdate(self, event):
        global MinMSE
        MinMSE = event # Setting Min MSE value
        self.MinMSElcd.display(event)

    def MaxMSElcdUpdate(self, event):
        global MaxMSE
        MaxMSE = event # Setting Max MSE value
        self.MaxMSElcd.display(event)

    def ThlcdUpdate(self, event):
        global threshold
        threshold = event/200 # Setting threshold value
        self.Thlcd.display(event/200)

    def TemplcdUpdate(self, event):
        global tempsteps
        tempsteps = event # Setting template steps
        self.Templcd.display(event)

    def FramelcdUpdate(self, event):
        global nth_frame
        nth_frame = event # Setting frame steps
        self.Framelcd.display(event)



    # Spinner events
    def nVal(self, event):
        global n
        n = round(event, 1) # Value of n template to use

    def rVal(self, event):
        global r
```

```
        r = round(event, 1) # Value of r template to use



    # Button events
    def Enter1(self):
        global background
        background = str(self.BackgroundInput.text()) # Background file to use
        print(self.BackgroundInput.text())
        self.OutputText.append('Background image chosen - {}' .format(backgrou
nd))

    def Enter2(self):
        global video
        video = str(self.VideoInput.text()) # Video file to use
        print(self.VideoInput.text())
        self.OutputText.append('Video file chosen - {}' .format(video))

    def Plot(self):
        x = PlotX
        y = PlotY
        z = PlotZ
        tracking = PlotCOUNT

        pts = mlab.points3d(x, y, z, tracking, colormap="copper", mode='point'
)
        pts.actor.actor.property.render_points_as_spheres = True
        pts.actor.actor.property.point_size = 5.0

        mlab.show()

    def Track(self):
        answer = QtWidgets.QMessageBox.question(
            self.centralwidget, 'Lauch Tracker', 'Are you sure you are ready t
o execute Tracker from Initilisation Settings?',
            QtWidgets.QMessageBox.Yes | QtWidgets.QMessageBox.No, QtWidgets.QM
essageBox.No)
        if answer == QtWidgets.QMessageBox.Yes:#
            self.LoadingText.setFontPointSize(20)
            self.LoadingText.append('------Tracker Launched------')
            self.LoadingText.setFontPointSize(10)
            self.Tracker()

        else:
            pass

    def Run(self):
        # Attempting to collect GUI input info to run
        # Start the Loading Bar
```

```
        # Display Output Text

        self.LoadingText.setFontPointSize(20)
        self.LoadingText.append('---Initialisation Launched---')
        self.LoadingText.setFontPointSize(10)

        global background
        self.OutputText.append('Background file chosen - {}' .format(backgroun
d))
        global video
        self.OutputText.append('Video file chosen - {}' .format(video))
        global count
        global end
        end = 11
        self.OutputText.append('Number of Frames to process - {}' .format(end
- 1))
        global nth_frame
        self.OutputText.append('Steps in Frames being counted - {}' .format(nt
h_frame))
        global contrast
        self.OutputText.append('Contrast Value - {}' .format(contrast))
        global sharpness
        self.OutputText.append('Sharpness Value - {}' .format(sharpness))
        global n
        self.OutputText.append('Template Refractive index - {}' .format(n))
        global r
        self.OutputText.append('Template sphere radius - {}' .format(r))
        global Location
        global threshold
        threshold = 0.07
        self.OutputText.append('Threshold Value - {}' .format(threshold))
        global tempsteps
        tempsteps = 5
        self.OutputText.append('Steps in Templates being counted - {}' .format
(tempsteps))
        global tempcount
        global tempend
        global brightness
        global MinMSE
        global MaxMSE
        global PlotZ
        global PlotTH
        global PlotMSE



        ########## Background and Video Processing ##########
        bgd = cv2.imread('..Backgrounds/{}.png' .format(background)) # Locatio
n of background image
```

```
        bgdgray = cv2.cvtColor(bgd, cv2.COLOR_BGR2GRAY)
        arraybgdgray = np.array(bgdgray)
        vidcap = cv2.VideoCapture('..Videos/{}.avi' .format(video)) # Location
 of video



        ######### Create Frames #########
        while(count < end):
            tic = time.perf_counter() # Estimating run time starter

            # Capture frame-by-frame
            skip = nth_frame * count
            vidcap.set(1, skip)
            ret, frame = vidcap.read()

            self.LoadingText.append('Reading frame %d' % skip)

            # Operations on the frames come here
            imggray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # BGR or RGB (ca
st different readings)
            array1 = np.array(imggray)
            arrayDIV = np.divide(array1, arraybgdgray) * brightness
            divided = Image.fromarray(arrayDIV)
            final = divided.convert("L")

            # Image contraster
            enhancer1 = ImageEnhance.Contrast(final)
            enhanced1 = enhancer1.enhance(contrast)
            enhancer2 = ImageEnhance.Sharpness(enhanced1)
            enhanced2 = enhancer2.enhance(sharpness)

            # Save the resulting frame
            enhanced2.save("enhanced frames/frame%d.png" % count)
            self.LoadingText.append('Processed frame %d' % count)

            # Matching process
            img_rgb = cv2.imread("enhanced frames/frame%d.png" % count)
            img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)
            img_to_crop = Image.open("enhanced frames/frame%d.png" % count).co
nvert('L')

            while(tempcount < tempend):
                template = cv2.imread("templates/0.703 pix-
um/n={}, r={}/img{}.png" .format(n, r, tempcount), 0) # 0.711 pix-um/
                w, h = template.shape[::-1]
                templatePIL = Image.open("templates/0.703 pix-
um/n={}, r={}/img{}.png" .format(n, r, tempcount)).convert('L') # 0.711 pix-
um/
```

```
                w, h = templatePIL.size

                res = cv2.matchTemplate(img_gray,template,cv2.TM_CCOEFF_NORMED
)
                min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
                loc = np.where(res >= threshold)
                for pt in zip(*loc[::-1]):
                    cv2.circle(img_rgb, (pt[0] + int(h/2), pt[1] + int(h/2)),
10, (0,0,255), 1)

                    # Crop matched image to size of template for MSE
                    crop = img_to_crop.crop((pt[0], pt[1], pt[0] + w, pt[1] +
h))

                    crop = crop.save("cropped frames/crop f={}, s={}.png" .for
mat(count, tempcount))
                    crop = Image.open("cropped frames/crop f={}, s={}.png" .fo
rmat(count, tempcount))

                    def mse(actual, pred):
                        actual, pred = np.array(actual), np.array(pred)
                        return np.square(np.subtract(actual,pred)).mean()

                    mse_val = mse(templatePIL, crop)

                    plot3 = [[(tempcount) for col in range(1)] for row in rang
e(1)] # Z coordinates
                    plot4 = [[(max_val) for col in range(1)] for row in range(
1)] # Threshold
                    plot5 = [[(mse_val) for col in range(1)] for row in range(
1)] # MSE

                    PlotZ = PlotZ + plot3
                    PlotTH = PlotTH + plot4
                    PlotMSE = PlotMSE + plot5

                tempcount += tempsteps

            # Save the resulting matched frame
            cv2.imwrite("matched frames/frame%d.png" % count,img_rgb)

            toc = time.perf_counter() # Estimating run time stopper

            self.LoadingText.append('Matched frame %d \n' % count)
            self.LoadingText.append(f"Estimated Time Remaining: {int((toc - ti
c)*((end - 1) - count))} seconds")

            tempcount = 1
```

```
            count += 1
            self.progressBar.setValue(int((count/(end-
1))*100)) # LoadingBar Protocol

        # Reset counter for multiple plotting events in singluar window
        count = 1
        self.LoadingText.setFontPointSize(15)
        self.LoadingText.append('\nAll Frames Created \nLoading Plots.......')
        self.LoadingText.setFontPointSize(12)

        # Arrays for Plots
        z = PlotZ
        th = PlotTH
        mse = PlotMSE

        # Output Data
        Matchval = np.mean(th)
        SDev = np.std(th)
        UDep = len(np.unique(z))
        RDep = len(z) - UDep
        MSEval = np.mean(mse)
        RecMinMSE = int(np.percentile(mse, 10))
        RecMaxMSE = int(np.percentile(mse, 90))
        RecMinTH = round(np.percentile(th, 10), 2)



        ######### MSE-TH plotter #########
        fig6 = plt.figure()

        self.canvas = FigureCanvas(fig6)
        self.axes = fig6.add_subplot()

        self.axes.axvline(RecMinMSE, color='b')
        self.axes.axvline(RecMaxMSE, color='b')
        self.axes.axhline(RecMinTH, color='b')

        self.axes.axvspan(RecMaxMSE, (np.max(mse)*1.05), facecolor='0.5', alph
a = 0.5)
        self.axes.axvspan(RecMinMSE, (np.min(mse)*0.9), facecolor='0.5', alpha
 = 0.5)
        self.axes.add_patch(Rectangle((RecMinMSE, (np.min(th)*0.9)), (RecMaxMS
E-RecMinMSE), (RecMinTH-(np.min(th)*0.9)), facecolor='0.5', alpha = 0.5))

        pnt = self.axes.scatter(mse, th, c=z, s=2)
        cbar = plt.colorbar(pnt)
        cbar.set_label("Z Axis")
        self.axes.set_title('n = {}, r = {}' .format(n, r))
        self.axes.set_xlabel('MSE Value')
```

```
        self.axes.set_ylabel('Match Value')
        self.axes.set_xlim((np.min(mse)*0.9), (np.max(mse)*1.05))
        self.axes.set_ylim((np.min(th)*0.9), (np.max(th)*1.05))
        self.axes.scatter(mse, th, c=z, s=2)
        self.axes.set_axisbelow(True)
        self.axes.minorticks_on()
        self.axes.grid(which='major', linestyle='-
', linewidth='0.5', color='red')
        self.axes.grid(which='minor', linestyle='-
', linewidth='0.1', color='black')

        plt.savefig('.MSE-TH Plots/MSE-Th(1) n={}, r={}.png' .format(n, r))
        self.layMSETH.addWidget(FigureCanvas(fig6))
        self.LoadingText.append('\nMSE-Th(1) Plot Created and saved')



        ######### Z-MSE plotter #########
        fig5, ax5 = plt.subplots()

        ax5.axhline(RecMinMSE, color='m')
        ax5.axhline(RecMaxMSE, color='m')
        ax5.plot(z, mse, '.', color='red', markersize=2)
        ax5.set_title('n = {}, r = {}, Average={}' .format(n, r, MSEval))
        ax5.axhspan(RecMaxMSE, (np.max(mse)*1.05), facecolor='0.5', alpha = 0.
5)
        ax5.axhspan(RecMinMSE, (np.min(mse)*0.9), facecolor='0.5', alpha = 0.5
)
        ax5.set_xlabel('Z Axis')
        ax5.set_ylabel('MSE Value')
        ax5.set_axisbelow(True)
        ax5.set_ylim((np.min(mse)*0.9), (np.max(mse)*1.05))
        ax5.minorticks_on()
        ax5.grid(which='major', linestyle='-', linewidth='0.5', color='blue')
        ax5.grid(which='minor', linestyle='-', linewidth='0.1', color='black')

        plt.savefig('.MSE-Z Plots/MSE-Z(1) n={}, r={}.png' .format(n, r))
        self.layMSEZ.addWidget(FigureCanvas(fig5))
        self.LoadingText.append('\nZ-MSE(1) Plot Created and saved')



        ######### Z-Th plotter #########
        fig4, ax4 = plt.subplots()

        ax5.axhline(RecMinTH, color='m')
        ax4.plot(z, th, '.', color='red', markersize=2)
        ax4.set_title('n = {}, r = {}, Average={}' .format(n, r, Matchval))
        ax4.axhspan(RecMinTH, (np.min(th)*0.9), facecolor='0.5', alpha = 0.5)
```

```
        ax4.set_xlabel('Z Axis')
        ax4.set_ylabel('Match Value')
        ax4.set_axisbelow(True)
        ax4.set_ylim((np.min(th)*0.9), (np.max(th)*1.05))
        ax4.minorticks_on()
        ax4.grid(which='major', linestyle='-', linewidth='0.5', color='blue')
        ax4.grid(which='minor', linestyle='-', linewidth='0.1', color='black')

        plt.savefig('.Th-Z Plots/Th-Z(1) n={}, r={}.png' .format(n, r))
        self.layTHZ.addWidget(FigureCanvas(fig4))
        self.LoadingText.append('\nZ-Th(1) Plot Created and saved\n')
        self.LoadingText.setFontPointSize(10)



        ######### Print Specifics #########
        self.OutputText.setFontPointSize(20)
        self.OutputText.append('\n \n------INPUT SPECIFICS-----')
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Frames Settings**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Start - %d, End - %d , Total - %d, Steps - %d
\n' % (nth_frame, nth_frame*(end-1), (end-1), nth_frame))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Enhancement Settings**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Contrast - %d, Sharpness - %d \n' % (contrast,
 sharpness))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Template Settings**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Folder - n={}, r={}, Threshold - {}, Steps - {
} \n \n' .format(n, r, threshold, tempsteps))
        self.OutputText.setFontPointSize(20)

        self.OutputText.append('-----OUTPUT SPECIFICS----')
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Matching Data**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Average Match Value - {}, S.Dev of Match Value
 - {}, Recommended Min Match Value - {} \n' .format(Matchval, SDev, RecMinTH))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Depth Data**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Total Number of Unique Depths - {}, Total Numb
er of Repeating Depths - {} \n' .format(UDep, RDep))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**MSE Data**')
        self.OutputText.setFontPointSize(10)
```

```
        self.OutputText.append('Average MSE Value - {}, Recommended Min MSE -
{}, Recommended Max MSE - {} \n' .format(MSEval, RecMinMSE, RecMaxMSE))

        PlotZ = []
        PlotTH = []
        PlotMSE = []

    def Tracker(self):
        # Attempting to collect GUI input info to run
        # Start the Loading Bar
        # Display Output Text

        global background
        global video
        global videoframes
        global count
        global nth_frame
        global end
        end = int(videoframes/nth_frame) + 1 # Assuming 10,000 frames in video
        self.OutputText.append('Number of Frames to process - {}' .format(end
- 1))
        self.OutputText.append('Steps in Frames being counted - {}' .format(nt
h_frame))
        global contrast
        self.OutputText.append('Contrast Value - {}' .format(contrast))
        global sharpness
        self.OutputText.append('Sharpness Value - {}' .format(sharpness))
        global n
        self.OutputText.append('Template Refractive index - {}' .format(n))
        global r
        self.OutputText.append('Template sphere radius - {}' .format(r))
        global Location
        global threshold
        self.OutputText.append('Threshold Value - {}' .format(threshold))
        global tempsteps
        self.OutputText.append('Steps in Templates being counted - {}' .format
(tempsteps))
        global tempcount
        global tempend
        global brightness
        global MinMSE
        global MaxMSE
        global PlotX
        global PlotY
        global PlotZ
        global PlotTH
        global PlotMSE
        global PlotCOUNT
```

```
        ########## Background and Video Processing ##########
        bgd = cv2.imread('..Backgrounds/{}.png' .format(background)) # Locatio
n of background image
        bgdgray = cv2.cvtColor(bgd, cv2.COLOR_BGR2GRAY)
        arraybgdgray = np.array(bgdgray)
        vidcap = cv2.VideoCapture('..Videos/{}.avi' .format(video)) # Location
 of video



        ######### Create Frames #########
        while(count < end):
            tic = time.perf_counter() # Estimating run time starter

            # Capture frame-by-frame
            skip = nth_frame * count
            vidcap.set(1, skip)
            ret, frame = vidcap.read()

            self.LoadingText.append('Reading frame %d' % skip)

            # Operations on the frames come here
            imggray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            array1 = np.array(imggray)
            arrayDIV = np.divide(array1, arraybgdgray) * brightness
            divided = Image.fromarray(arrayDIV)
            final = divided.convert("L")

            # Image contraster
            enhancer1 = ImageEnhance.Contrast(final)
            enhanced1 = enhancer1.enhance(contrast)
            enhancer2 = ImageEnhance.Sharpness(enhanced1)
            enhanced2 = enhancer2.enhance(sharpness)

            # Save the resulting frame
            enhanced2.save("enhanced frames/frame%d.png" % count)

            self.LoadingText.append('Processed frame %d' % count)

            # Matching process
            img_rgb = cv2.imread("enhanced frames/frame%d.png" % count)
            img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)
            img_to_crop = Image.open("enhanced frames/frame%d.png" % count).co
nvert('L')

            while(tempcount < tempend):
```

```
                template = cv2.imread("templates/0.703 pix-
um/n={}, r={}/img{}.png" .format(n, r, tempcount), 0) # 0.711 pix-um/
                w, h = template.shape[::-1]
                templatePIL = Image.open("templates/0.703 pix-
um/n={}, r={}/img{}.png" .format(n, r, tempcount)).convert('L') # 0.711 pix-
um/
                w, h = templatePIL.size

                res = cv2.matchTemplate(img_gray,template,cv2.TM_CCOEFF_NORMED
)
                min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)
                loc = np.where(res >= threshold)
                for pt in zip(*loc[::-1]):
                    cv2.circle(img_rgb, (pt[0] + int(h/2), pt[1] + int(h/2)),
10, (0,0,255), 1)

                    # Crop matched image to size of template for MSE
                    crop = img_to_crop.crop((pt[0], pt[1], pt[0] + w, pt[1] +
h))

                    crop = crop.save("cropped frames/crop f={}, s={}.png" .for
mat(count, tempcount))
                    crop = Image.open("cropped frames/crop f={}, s={}.png" .fo
rmat(count, tempcount))

                    def mse(actual, pred):
                        actual, pred = np.array(actual), np.array(pred)
                        return np.square(np.subtract(actual,pred)).mean()

                    mse_val = mse(templatePIL, crop)


                    if (mse_val > MinMSE) and (mse_val < MaxMSE):
                        plot1 = [[(pt[0] + int(h/2)) for col in range(1)] for
row in range(1)] # X coordinates
                        plot2 = [[(pt[1] + int(h/2)) for col in range(1)] for
row in range(1)] # Y coordinates
                        plot3 = [[(tempcount) for col in range(1)] for row in
range(1)] # Z coordinates
                        plot4 = [[(max_val) for col in range(1)] for row in ra
nge(1)] # Threshold
                        plot5 = [[(mse_val) for col in range(1)] for row in ra
nge(1)] # MSE
                        plot6 = [[(count) for col in range(1)] for row in rang
e(1)] # Frame Counts

                        PlotX = PlotX + plot1
                        PlotY = PlotY + plot2
                        PlotZ = PlotZ + plot3
```

```
                        PlotTH = PlotTH + plot4
                        PlotMSE = PlotMSE + plot5
                        PlotCOUNT = PlotCOUNT + plot6

                tempcount += tempsteps

            # Save the resulting matched frame
            cv2.imwrite("matched frames/frame%d.png" % count,img_rgb)

            toc = time.perf_counter() # Estimating run time stopper
            self.LoadingText.append('Matched frame %d \n' % count)
            self.LoadingText.append(f"Estimated Time Remaining: {int((toc - ti
c)*((end - 1) - count))} seconds")

            tempcount = 1
            count += 1
            self.progressBar.setValue(int((count/(end-
1))*100)) # LoadingBar Protocol

        # Reset counter for multiple plotting events in singluar window
        count = 1
        self.LoadingText.setFontPointSize(15)
        self.LoadingText.append('\nAll Frames Created \nLoading Plots.......')
        self.LoadingText.setFontPointSize(12)

        # Arrays for Plots
        x = PlotX
        y = PlotY
        z = PlotZ
        th = PlotTH
        mse = PlotMSE
        tracking = PlotCOUNT

        # Output Data
        UDep = len(np.unique(z))
        RDep = len(z) - UDep
        UX = len(np.unique(x))
        UY = len(np.unique(y))
        Matchval = np.mean(th)
        MSEval = np.mean(mse)



        ######### MSE-TH plotter #########
        fig6 = plt.figure()

        self.canvas = FigureCanvas(fig6)
        self.axes = fig6.add_subplot()
```

```
        pnt = self.axes.scatter(mse, th, c=z, s=2)
        cbar = plt.colorbar(pnt)
        cbar.set_label("Z Axis")

        self.axes.set_title('n = {}, r = {}' .format(n, r))
        self.axes.set_xlabel('MSE Value')
        self.axes.set_ylabel('Match Value')
        self.axes.scatter(mse, th, c=z, s=2)

        plt.savefig('.MSE-TH Plots/MSE-Th(2) n={}, r={}.png' .format(n, r))
        self.layMSETH.addWidget(FigureCanvas(fig6))
        self.LoadingText.append('\nMSE-Th(2) Plot Created and saved')



        ######### Z-MSE plotter #########
        fig5, ax5 = plt.subplots()

        ax5.plot(z, mse, '.', color='red', markersize=2)
        ax5.set_title('n = {}, r = {}, Average={}' .format(n, r, MSEval))
        ax5.set_xlabel('Z Axis')
        ax5.set_ylabel('MSE Value')

        plt.savefig('.MSE-Z Plots/MSE-Z(2) n={}, r={}.png' .format(n, r))
        self.layMSEZ.addWidget(FigureCanvas(fig5))
        self.LoadingText.append('\nZ-MSE(2) Plot Created and saved')



        ######### Z-Th plotter #########
        fig4, ax4 = plt.subplots()

        ax4.plot(z, th, '.', color='red', markersize=2)
        ax4.set_title('n = {}, r = {}, Average={}' .format(n, r, Matchval))
        ax4.set_xlabel('Z Axis')
        ax4.set_ylabel('Match Value')

        plt.savefig('.Th-Z Plots/Th-Z(2) n={}, r={}.png' .format(n, r))
        self.layTHZ.addWidget(FigureCanvas(fig4))
        self.LoadingText.append('\nZ-Th(2) Plot Created and saved')



        ######### 2D plotter #########
        fig1 = plt.figure()

        self.canvas1 = FigureCanvas(fig1)
        self.axes1 = fig1.add_subplot()
```

```
        pnt = self.axes1.scatter(mse, th, c=z, s=2)
        cbar = plt.colorbar(pnt)
        cbar.set_label("Z Axis")

        self.axes1.set_title('n={}, r={}, f={}, s={}' .format(n, r, end-
1, tempsteps))
        self.axes1.set_xlabel('X Axis')
        self.axes1.set_ylabel('Y Axis')
        self.axes1.scatter(x, y, c=z, s=2)

        plt.savefig('.2D Plots/2D n={}, r={}, f={}, s={}.png' .format(n, r, en
d-1, tempsteps))
        self.lay2D.addWidget(FigureCanvas(fig1))
        self.LoadingText.append('\n2D Plot Created and saved')



        ######### 3D plotter #########
        fig7 = plt.figure()

        self.canvas = FigureCanvas(fig7)
        self.axes = fig7.add_subplot(111, projection='3d')

        pnt3d = self.axes.scatter(x, y, z, c=tracking, s=1)
        pnt3d.set_clim(np.min(tracking), np.max(tracking))
        cbar = plt.colorbar(pnt3d)
        cbar.set_label("Frame Number")

        self.axes.set_title('n={}, r={}, f={}, s={}.png' .format(n, r, end-
1, tempsteps))
        self.axes.set_xlabel('X Axis')
        self.axes.set_ylabel('Y Axis')
        self.axes.set_zlabel('Z Axis')
        self.axes.set_zlim([0, 220])
        self.axes.scatter(x, y, z, c=tracking, s=1)

        plt.savefig('.3D Plots/3D n={}, r={}, f={}, s={}.png' .format(n, r, en
d-1, tempsteps))
        self.lay3D.addWidget(self.canvas)
        self.LoadingText.append('\n3D Plot Created and saved \n')



        ########## Saving DATA files ###########
        np.savetxt('.Data files/X Data/n={}, r={}, f={}, s={}.txt' .format(n,
r, end-1, tempsteps), x)
        np.savetxt('.Data files/Y Data/n={}, r={}, f={}, s={}.txt' .format(n,
r, end-1, tempsteps), y)
```

```
        np.savetxt('.Data files/Z Data/n={}, r={}, f={}, s={}.txt' .format(n,
r, end-1, tempsteps), z)
        np.savetxt('.Data files/Count Data/n={}, r={}, f={}, s={}.txt' .format
(n, r, end-1, tempsteps), tracking)
        np.savetxt('.Data files/MSE Data/co={}, sh={}, th={}.txt' .format(cont
rast, sharpness, threshold), mse)

        self.LoadingText.append('All data points saved')
        self.LoadingText.setFontPointSize(10)



        ######### Print Specifics #########
        self.OutputText.setFontPointSize(20)
        self.OutputText.append('\n \n------INPUT SPECIFICS-----')
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Frames Settings**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Start - %d, End - %d , Total - %d, Steps - %d
\n' % (nth_frame, nth_frame*(end-1), (end-1), nth_frame))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Enhancement Settings**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Contrast - %d, Sharpness - %d \n' % (contrast,
 sharpness))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Template Settings**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Folder - n={}, r={}, Threshold - {}, Steps - {
} \n \n' .format(n, r, threshold, tempsteps))
        self.OutputText.setFontPointSize(20)

        self.OutputText.append('-----OUTPUT SPECIFICS----')
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**Depth Data**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Total Number of Unique Depths - {}, Total Numb
er of Repeating Depths - {} \n' .format(UDep, RDep))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**MSE Data**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Average MSE Value - {} \n' .format(MSEval))
        self.OutputText.setFontPointSize(15)
        self.OutputText.append('**X-Y Data**')
        self.OutputText.setFontPointSize(10)
        self.OutputText.append('Total Number of Unique X Values - {}, Total Nu
mber of Unique Y Values - {} \n \n' .format(UX, UY))
```

```
    ########## Window Setup ##########
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "The Hologram Analy
sis System 3        [THAS3]       (created by I.T.Booth)"))
        self.BackgroundLabel.setText(_translate("MainWindow", "Background"))
        self.VideoLabel.setText(_translate("MainWindow", "Video"))
        self.label_3.setText(_translate("MainWindow", "Min MSE"))
        self.label_4.setText(_translate("MainWindow", "Max MSE"))
        self.label_5.setText(_translate("MainWindow", "Threshold"))
        self.label_6.setText(_translate("MainWindow", "Template Steps"))
        self.label_7.setText(_translate("MainWindow", "Frame Steps"))
        self.nLabel.setText(_translate("MainWindow", "n ="))
        self.rLabel.setText(_translate("MainWindow", "r ="))

        self.RunButton.setText(_translate("MainWindow", "Initialise"))
        self.BackgroundButton.setText(_translate("MainWindow", "Enter"))
        self.VideoButton.setText(_translate("MainWindow", "Enter"))
        self.TrackButton.setText(_translate("MainWindow", "Track"))
        self.PlotButton.setText(_translate("MainWindow", "Interactive 3D"))

        self.Displaytabs.setTabText(self.Displaytabs.indexOf(self.PlotMSEZ), _
translate("MainWindow", "Z-MSE"))
        self.Displaytabs.setTabText(self.Displaytabs.indexOf(self.PlotTHZ), _t
ranslate("MainWindow", "Z-Th"))
        self.Displaytabs.setTabText(self.Displaytabs.indexOf(self.PlotMSETH),
_translate("MainWindow", "MSE-Th (Z)"))
        self.Displaytabs.setTabText(self.Displaytabs.indexOf(self.Plot2D), _tr
anslate("MainWindow", "2D (Z)"))
        self.Displaytabs.setTabText(self.Displaytabs.indexOf(self.Plot3D), _tr
anslate("MainWindow", "3D"))


########## Execute above protocol ##########
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    app.setStyle('Fusion')
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Appendix 7: The Python code for The Hologram Analysis System 3, producing the final user interface to be used in conjunction with the hologram creation optical computer

```
import sys
import numpy as np
from traits.api import Instance
from mayavi import mlab
from mayavi.core.ui.api import MlabSceneModel
from PyQt5 import (QtCore, QtGui, QtWidgets)
from PyQt5.QtCore import QProcess
from PyQt5.QtWidgets import (QWidget, QDesktopWidget, QMessageBox, QApplicatio
n)


count = 0
########## Main Setup ###########
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("Plotter")
        MainWindow.resize(580, 240) # Initial Window Dimensions
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("Icon.ico"), QtGui.QIcon.Selected, QtGui.
QIcon.On)
        MainWindow.setWindowIcon(icon)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")



        ########## Labels ##########
        self.FrameLabel = QtWidgets.QLabel(self.centralwidget)
        self.FrameLabel.setGeometry(QtCore.QRect(30, 27, 200, 20))
        self.FrameLabel.setObjectName("FrameLabel")
        self.nLabel = QtWidgets.QLabel(self.centralwidget)
        self.nLabel.setGeometry(QtCore.QRect(35, 100, 20, 20))
        self.nLabel.setObjectName("nLabel")
        self.rLabel = QtWidgets.QLabel(self.centralwidget)
        self.rLabel.setGeometry(QtCore.QRect(135, 100, 20, 20))
        self.rLabel.setObjectName("rLabel")
        self.sLabel = QtWidgets.QLabel(self.centralwidget)
        self.sLabel.setGeometry(QtCore.QRect(235, 100, 20, 20))
        self.sLabel.setObjectName("sLabel")



        ########## Text Inputs ##########
        self.FrameInput = QtWidgets.QLineEdit(self.centralwidget)
        self.FrameInput.setGeometry(QtCore.QRect(180, 20, 145, 40))
        self.FrameInput.setObjectName("FrameInput")

        self.nSpinner = QtWidgets.QLineEdit(self.centralwidget)
        self.nSpinner.setGeometry(QtCore.QRect(66, 87, 60, 50))
```

```
        self.nSpinner.setObjectName("nSpinner")

        self.rSpinner = QtWidgets.QLineEdit(self.centralwidget)
        self.rSpinner.setGeometry(QtCore.QRect(166, 87, 60, 50))
        self.rSpinner.setObjectName("rSpinner")

        self.sSpinner = QtWidgets.QLineEdit(self.centralwidget)
        self.sSpinner.setGeometry(QtCore.QRect(266, 87, 60, 50))
        self.sSpinner.setObjectName("sSpinner")



        ########## Buttons ##########
        self.PlotButton = QtWidgets.QPushButton(self.centralwidget)
        self.PlotButton.setGeometry(QtCore.QRect(350, 87, 200, 50))
        self.PlotButton.setObjectName("PlotButton")
        self.PlotButton.setStyleSheet('QPushButton {background-
color: #A3C1DA; color: blue;}')
        self.PlotButton.clicked.connect(self.Plot)



        ########## Text Outputs ##########
        self.scrollArea = QtWidgets.QScrollArea(MainWindow)
        self.scrollArea.setGeometry(QtCore.QRect(30, 160, 520, 60)) # Location
 and Dimensions
        self.scrollArea.setWidgetResizable(True)
        self.scrollArea.setObjectName("scrollArea")

        self.OutputText = QtWidgets.QTextEdit(self.scrollArea)
        self.OutputText.setGeometry(QtCore.QRect(0, 0, 520, 60)) # Location an
d Dimensions
        self.OutputText.setObjectName("OutputText")
        self.OutputText.setFontPointSize(10)



        ########## Setup ###########
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 18))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
```

```
    ########## Protocol ##########
    def Plot(self):
        global n1
        global r1
        global f1
        global s1
        global count

        count += 1
        f1 = int(self.FrameInput.text()) # Value of f frame to use
        n1 = float(self.nSpinner.text()) # Value of n template to use
        r1 = float(self.rSpinner.text()) # Value of r template to use
        s1 = int(self.sSpinner.text()) # Value of s template to use

        answer = QtWidgets.QMessageBox.question(
            self.centralwidget, 'Replot', 'Are you sure you are ready to execu
te Replotter from Initilisation Settings?',
            QtWidgets.QMessageBox.Yes | QtWidgets.QMessageBox.No, QtWidgets.QM
essageBox.No)

        if answer == QtWidgets.QMessageBox.Yes:
            global fname1
            fname1 = (f'n={n1}, r={r1}, f={f1}, s={s1}.txt')
            self.OutputText.append(f'Loading files under - {fname1}')

            try:
                Scene = (f'scene{count}')
                self.OutputText.append(f'Scene number - {Scene}')
                Scene = Instance(MlabSceneModel, ())

                CountArray1 = np.loadtxt(f".Data files - Copy/Count Data/{fnam
e1}")
                XArray1 = np.loadtxt(f".Data files - Copy/X Data/{fname1}")
                YArray1 = np.loadtxt(f".Data files - Copy/Y Data/{fname1}")
                ZArray1 = np.loadtxt(f".Data files - Copy/Z Data/{fname1}")

                def draw_scene(scene):
                    mlab.figure(figure=scene.mayavi_scene)
                    pnts1 = mlab.points3d(XArray1, YArray1, ZArray1, CountArra
y1, colormap="copper", mode='point', figure=scene.mayavi_scene)
                    pnts1.actor.actor.property.render_points_as_spheres = True
                    pnts1.actor.actor.property.point_size = 5.0
                    mlab.title(f'{fname1}')
                    mlab.show()

                draw_scene(Scene)

            except:
```

```
                self.OutputText.append('Selected files cannot be found')
                count -= 1

        else:
            pass



    ########## Window Setup ##########
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Replotter       (c
reated by I.T.Booth)"))

        self.FrameLabel.setText(_translate("MainWindow", "Frames Processed"))
        self.nLabel.setText(_translate("MainWindow", "n ="))
        self.rLabel.setText(_translate("MainWindow", "r ="))
        self.sLabel.setText(_translate("MainWindow", "s ="))

        self.PlotButton.setText(_translate("MainWindow", "Plot Interactive 3D"
))



########## Execute above protocol ##########
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    app.setStyle('Fusion')
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    ag = QDesktopWidget().availableGeometry()
    sg = QDesktopWidget().screenGeometry()
    widget = MainWindow.geometry()
    x = 0.1 * (ag.width() - widget.width())
    y = 0.5 * ag.height() - widget.height()
    MainWindow.move(x, y)
    MainWindow.show()
    sys.exit(app.exec_())
```

Appendix 8: The Replotter interface code to allow THAS3 data files to be quickly and easily looked for and replotted if and when needed, as well as allowing multiple plots to be compared if necessary, for analysis