IMAGING AND DATA PROCESSING: TASK SHEET REPORT

NATHAN CROOKS

Table of Contents

[Abstract 2](#_Toc180619696)

[1. Task 1 2](#_Toc180619697)

[1.1 Theory and methods 2](#_Toc180619698)

[1.2 Results and discussion 4](#_Toc180619699)

[2. Task 2 5](#_Toc180619700)

[2.1 Theory and methods 5](#_Toc180619701)

[2.2 Results and discussion 7](#_Toc180619702)

[3. Task 3 8](#_Toc180619703)

[3.1 Theory and methods 8](#_Toc180619704)

[3.2 Results and discussion 9](#_Toc180619705)

[4. Task 4 10](#_Toc180619706)

[4.1 Theory and methods 10](#_Toc180619707)

[4.2 Results and discussion 10](#_Toc180619708)

[5. Task 5 10](#_Toc180619709)

[5.1 Theory and methods 10](#_Toc180619710)

[5.2 Results and discussion 13](#_Toc180619711)

[6. References 14](#_Toc180619712)

# 

# Abstract

This project involved a range of different image and data processing techniques to complete five separate tasks. Task 1 included simple image processing techniques such as edge detection, segmentation and greyscale enhancement. Task 2 involved affine transforms to produce a video recreation of head movement data. Task 3 required the time-frequency decomposition of multiple signals using Fourier and Hilbert transforms. Task 4 focused on machine learning and reproducibility of scientific results. Task 5 involved suitable processing of 25 near-infrared images so they could be combined to simulate a long exposure image.

# Task 1

## Theory and methods

The aim of this task was to implement three different simple imaging process techniques. The first of which involved greyscale enhancement via point processing in which a photo was taken where information was lost due to poor lighting. Every pixel of the image was turned to greyscale, , using the following formula:

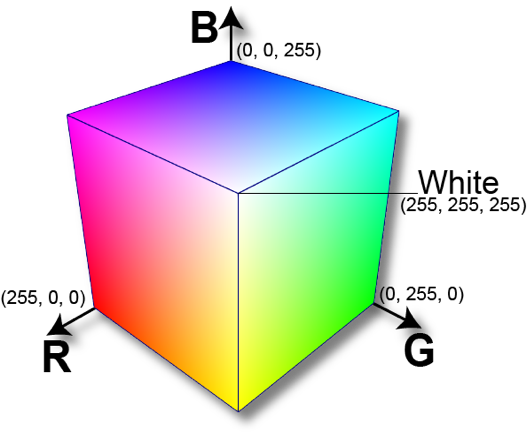
(1)

where and represent the intensities of the red, green and blue channels respectively of each pixel. Each value could range from . The nonlinear transmission function seen in equation 2 was then applied to each of the pixels:

(2)

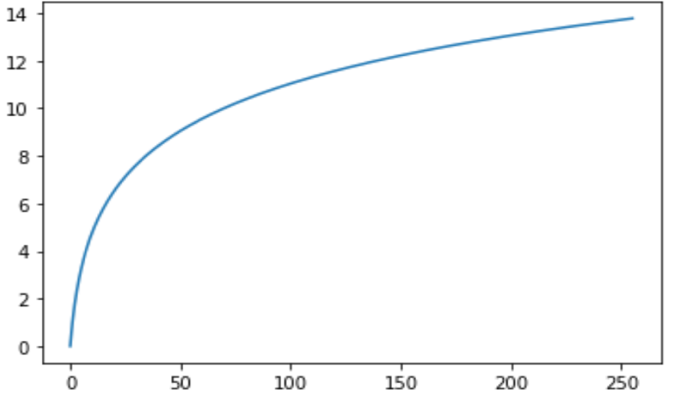
where was the new grey level value after the scaling and was a constant manually fine-tuned to give the best result. The value of was fine-tuned such that it stretched values at low greyscale value more extremely than it did for high greyscale values as seen in figure 1.

The value of led to sharp exponential increase for small values. This meant the difference in stretching between neighboring small values were much larger than for neighboring large values. This allowed the stretching to be more significant in darker regions than bright ones, enabling the detail hidden in these dark regions to be revealed.



**Figure 2:** A cube indicating the 3D representations for different colours in RGB space.

**Figure 1:** A plot of the non-linear transmission function seen in Equation 2 with .



The second process involved colour thresholding to segment an image based upon its different colour regions. The colour of a pixel can be represented in RGB space as seen in figure 2. To segment an image to highlight just the pure red pixels, code could be run that set all pixels with < 255, and to have values of zero in all three channels. This would leave only the pure red pixels and set all other pixels to black. However, in real life pure red is not often seen so a threshold in the red region of RGB space should instead be selected. For example, in this task if a pixel in the image had < 250, > 50 and > 30 then all its RGB values were set to equal to zero, i.e. a black pixel. All remaining pixels outside this threshold were left untouched so that the red colours in the image remained whilst everything else was set to black.

A diagram of a number

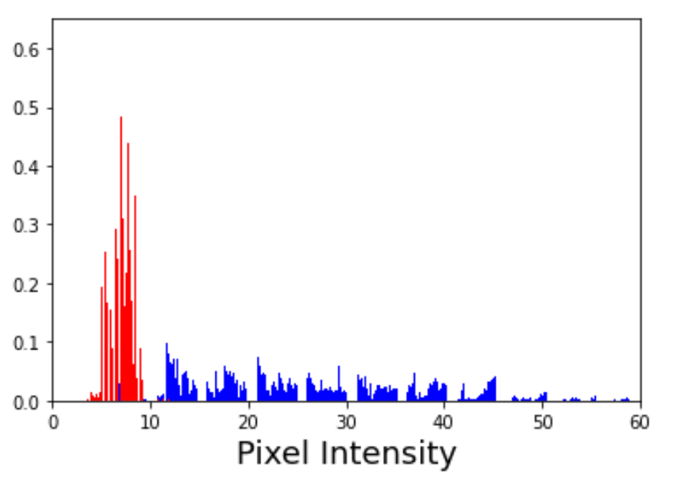
Description automatically generatedThe third process involved creation of a kernel to demonstrate vertical and horizontal edge detection on an image. The image was turned to greyscale using equation 1 again and Prewitt kernels [1] were applied to the greyscale value of each pixel, the process for the vertical edge detection can be seen in figure 3. To produce the output image the process seen in figure 3 was implemented, the red square was then moved along the image one pixel at a time to produce the output image. For an image this results in an output image as when the red square is centred on an edge value there are no pixels on the other side of the edge to make the matrix.

**Figure 3:** A visual description of what applying a vertical edge detection kernel to a section of an image produces. The values represent the greyscale levels of each pixel in an image. The red square represents a matrix centred on (blue square).

## Results and discussion

Figure 4 clearly indicates that once greyscale enhancement had been applied the detail originally hidden in the room became significantly more visible. As in figure 4 the wardrobe could now be identified as well as slightly more of the detail in the

**Figure 5:** A greyscale histogram of the number densities of the greyscale level in the picture before (in blue) and after (in red) the non-linear scaling was applied.



Number density

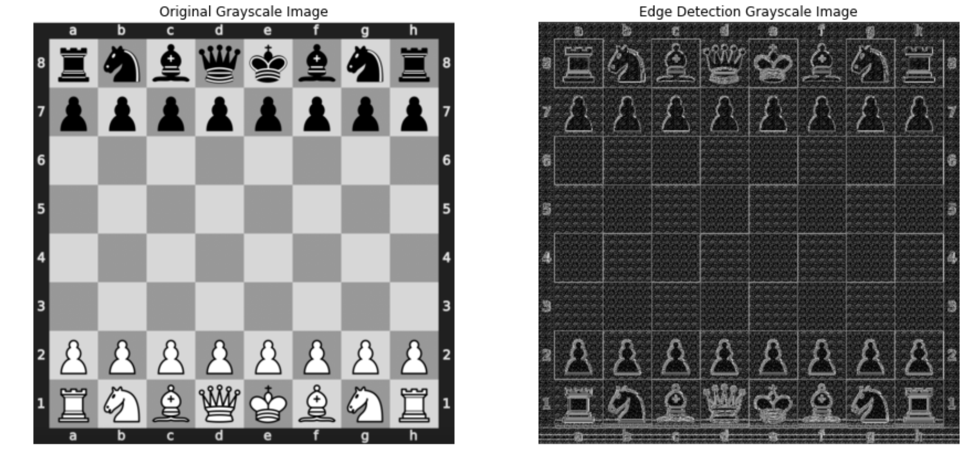
Greyscale intensity



**Figure 4:** The before and after of greyscale enhancement via point processing being implemented. The original image was taken in poor lighting and the enhancement was used to bring out the detail in the darkness.

posters on the doorframe. Figure 5 shows the effect applying this nonlinear scaling has on the greyscale histogram for the image. Before scaling, the blue histogram indicated there was a wide range of greyscale intensity values corresponding to dark and brighter regions. After scaling, the red histogram indicated that the new grey levels were all similar values as the originally darker regions had been scaled to values

From the second image in figure 6 it is clear the code maintained the regions of the image that were distinctly red. However, some of the undesired colours like brown and purple were also maintained, this was unavoidable as these colours have red undertones so will fall within the threshold defined for the red region. If a binary mask was applied, all regions within the threshold would be set to white and all others to black as seen in the third image of figure 6. Comparing the second and third images indicated that, for this particular image, applying a binary mask does not give much useful information as the regions with hints of red are indistinguishable from the regions of distinct red. From figure 7 it is clear the program was able to detect the outlines of the chess pieces as well as being able to indicate the general checkered pattern of a chess board. The program was only interested in the edges within the image and therefore the information about which pieces were white and which were black was lost.



**Figure 7:** The before and after a program to detect horizontal and vertical edges was implemented.



**Figure 6:** The before and after of colour thresholding being applied to highlight the red regions of an image. With the original image on the left, the segmented image maintaining colour in the middle and the segmented image using a binary mask on the right.



# 2. Task 2

## 2.1 Theory and methods

The aim of this task was to take 3D motion tracking data and use it to implement head motion via affine transformations. To present the results a 10 second clip displaying the part of the video that involved the most visible head movements. The data contained the coordinates of 50 markers placed on the subject’s head as well as the eight coordinates of the corners of the room, both at time before any head movement had occurred. The center of mass of the head could be approximated by averaging the and coordinates of the 50 marker coordinates, the equation for calculating the coordinate of the center of mass, can be seen in equation 3:

(3)

Having applied equation 3 for all three dimensions the COM, , was calculated to be . The reference frame was then translated so that the COM lie on the origin as the rotation matrices that will be mentioned later are only valid when being applied about the coordinate origin. The head was treated as a rigid body with six degrees of freedom [2] relating to the translation of the COM in the and dimensions as well as the rotations of the COM and about the and axis respectively. The remaining data was sampled at over a 60 second period giving 7200 data points for each degree of freedom.

To use this data to give an image of the position of the head at each point in time affine transformations need to be applied. To do so there are four associated affine transformation matrices that need to be used. One matrix corresponds to the translating a 3D vector in all three dimensions, , the remaining three , and correspond to rotations of a 3D vector through and . The equation showing how the translation matrix can be used to give the translated coordinates can be seen in equation 4:

(4)

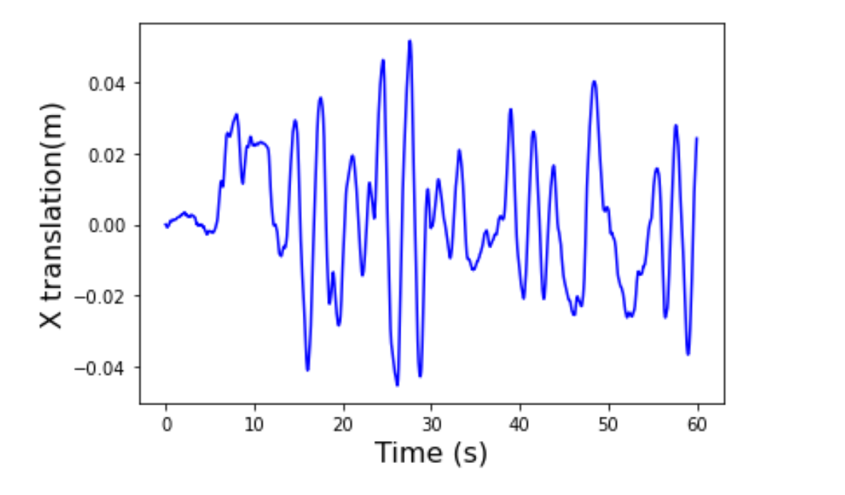
where , and are the coordinates before the translation and and are the translations of the point in the and dimensions respectively. The equation for rotating a 3D vector through (about the x axis) is given in equation 5:

(5)

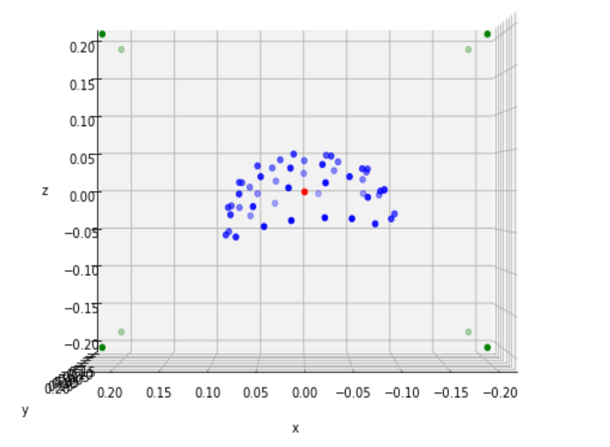
where , and are the new coordinates of the point after the rotation [3]. The remaining two matrices for rotations through and are slightly different but will not be included in the report as the application of them is the same as seen in equation 5. The only difference is the and terms appear at different matrix elements. The data given was for the translation and rotations of the COM, which supplied the values that needed to be substituted into the matrices, all four matrices were then applied to each of the 50 head points vectors to determine the new coordinate vector for each of the points. Every time this was done, and the resultant plot recorded, corresponded to one frame of the movie of the head movement.

## 2.2 Results and discussion

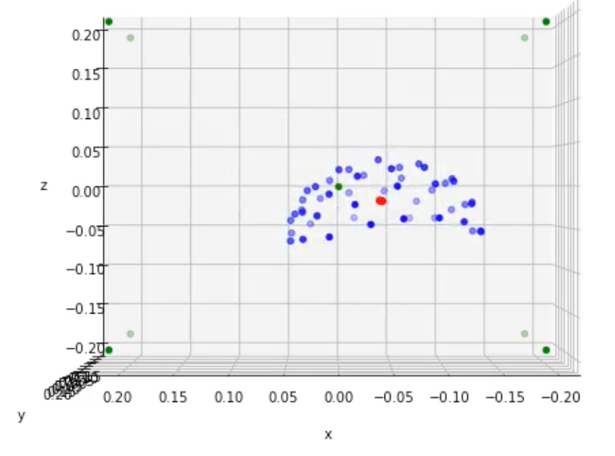
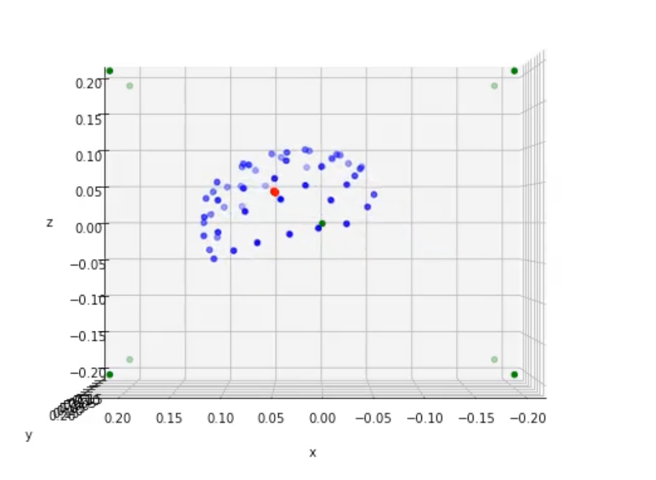
**Figure 8:** A plot of the how the translation of the COM in the dimension varied over the 60 seconds of data.



To determine which 10 seconds of the 60 seconds of data contained the most visible head movements, the timeseries of the data for each of the 6 data sets (degrees of freedom) was plotted. One of these time series plots can be seen below in figure 8. The most movement for this plot was clearly between the 20 – 30 second mark, as seen by the sharp high amplitude oscillations. Having checked the remaining five timeseries plots the 20 -30 second period appeared to have the most prominent movement.



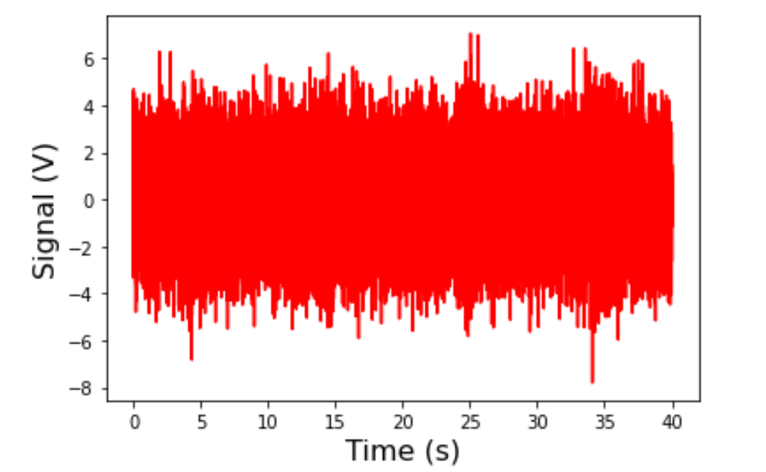
**Figure 9:** A 3D scatter plot of the head before any movement had occurred on the left, the middle and right image are snapshots of the head during its movement. The blue points correspond to the 50 head points, the red dot is the COM (initially centred on the origin) and the green dots represent the corners of the room and the origin of the reference frame in the middle.



The initial frame of the movie, and some subsequent ones can be seen in figure 9. The new COM of the head was calculated and plotted after every frame to help highlight the movement that, in the video is clear, but cannot be as easily seen in stills. The units of the translations and rotations were not given so had to be sensibly inferred. One corner to another of the room being ~2 units meant these units could be easily inferred as being in meters, this was reinforced by the head translations being on the order of ~10-2 which would correspond to a couple centimeter movements which once again sounded sensible. The rotations could not be as easily inferred so the code was simply run assuming the data was in degrees. This resulted in head movement with no rotation, from which it was inferred they were in radians. The code was rerun, and the resulting head movement had rotation about all three axes, hence verifying the units were radians.

# 3. Task 3

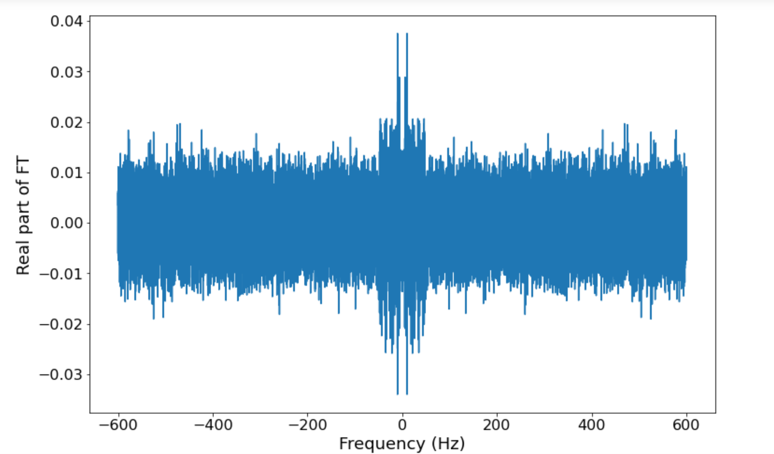
**Figure 10:** A plot showing how the analogue signal varied as a function of time.



## 3.1 Theory and methods

This task involved a 40 second analogue signal that initially appeared to have no visible structure, as seen in figure 10. The unit of the signal was not given so was taken to be in volts. Information on the dominant frequencies for different parts of the signal was given. The dominant signal between was in the frequency band, for it was in the band, for it was in the band and for it was in the band. This information would help to determine whether the time frequency spectrum plotted later was correct. The first step of processing the signal was to apply a Fourier transform (FT) as seen in equation 6:

**Figure 11:** A plot of the real part of the Fourier transformed signal as a function of frequency. Only the function for the positive frequencies have been plotted.



(6)

where is the signal being transformed

A blue and red sound wave

Description automatically generatedand is the angular frequency [4]. Viewing figure 11, the resultant plot still contained a lot of noise but there were now clearer regions of interest in the region as expected. Code was run to determine the highest FT peak present. This value was then used to determine amplitude of the band pass filters that would be applied to the FT, as if the amplitude was too small it would filter out relevant information. A gaussian bandpass filter [5] was selected, the form of which can be seen below:

(7)

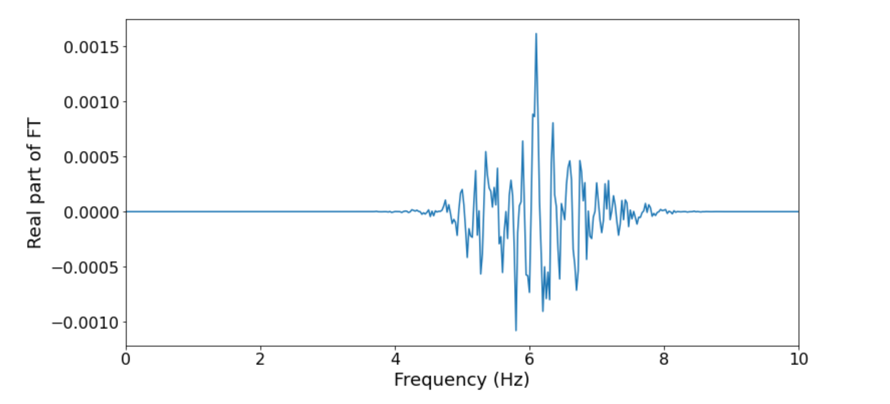
**Figure 12:** A plot showing the application of one of the bandpass filters used to filter out the undesired frequencies.

where is the max FT peak, is the

frequency the filter was centred on and is the standard deviation that determined how wide the filter would be. The value for was manually fine-tuned, judging by eye whether the filter included all the relevant information and omitted the rest. The gaussian was selected as it could be easily centred on the desired frequency and due to its exponential nature, quickly tailed off to zero at values out of this range, as seen in figure 12.

The bandpass filter with (red gaussian in figure 12) was applied and the resultant plot can be seen in figure 13. Outside of the range the signal had been filtered to zero, leaving only the desired information about the signal within this region. This process was repeated from by increments of , to sample the signal at all of the integer frequencies present. The signals were then inverse Fourier transformed (IFT) to convert the signals from the frequency back to the time domain. A Hilbert transform was then applied to the signal and the absolute value of the result was taken [6]. This resulted in a time course displaying the instantaneous envelope of the oscillations, the envelope associated with the bandpass filter can be seen plotted over the signal in figure 14.

**Figure 13:** The Fourier transform of the signal after the bandpass filter had been applied.



## A close-up of a graph Description automatically generated3.2 Results and discussion

A blue and red sound wave

Description automatically generated

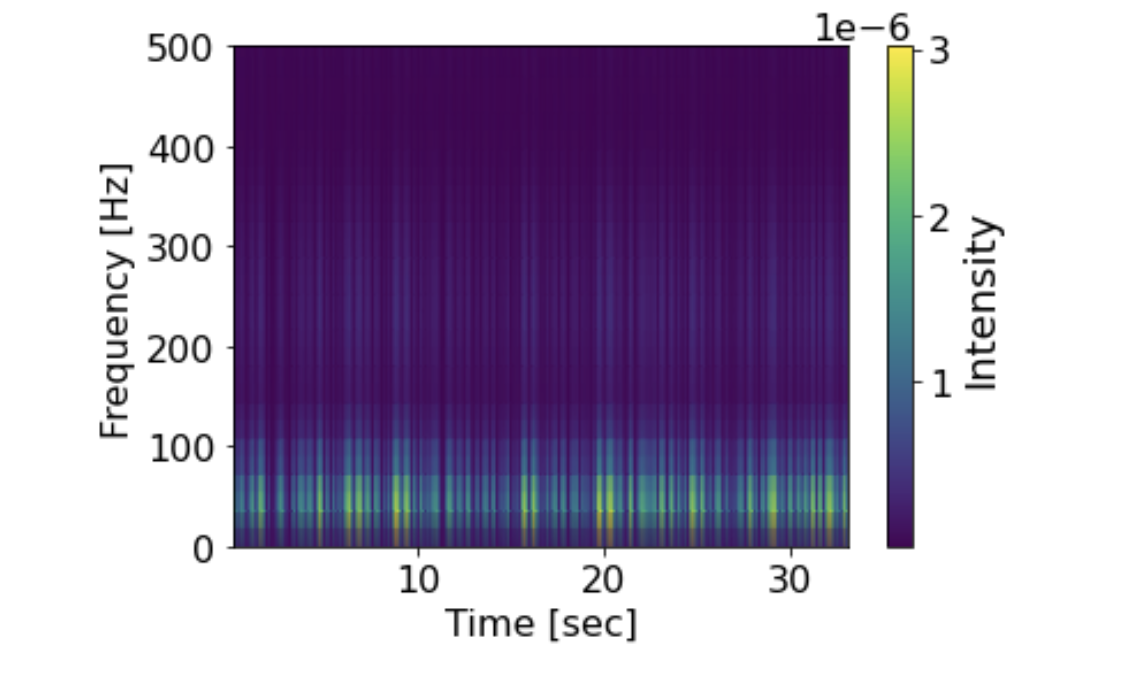
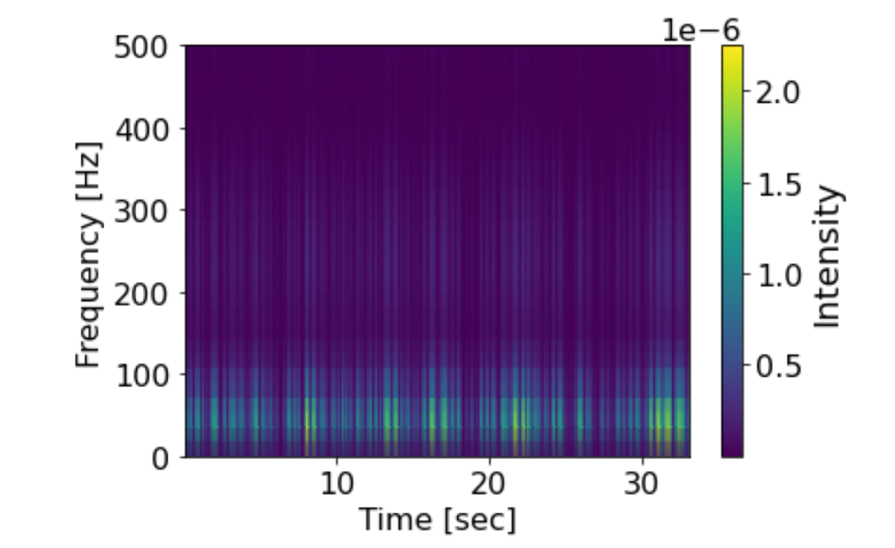
**Figure 15:** A time frequency spectrum for the signal with a colour bar to highlight the strength of the different frequency signals at each point in time.

**Figure 14:** A plot containing the filtered analytic signal after an IFT has been applied in blue with the instantaneous envelope of oscillations in red.

Figure 14 shows the signal filtered using a bandpass filter, which clearly has its largest intensity signal in the range, this was as expected as in this period the dominant signal was in the range. The plot of time frequency spectrum seen in figure 15 contains some anticipated features as well as some surprising ones. The traces stop at as expected as the largest of the dominant frequency ranges was . The yellow peaks in the range between was expected for reasons previously stated, however the yellow peaks in the remaining three intervals were not anticipated. The dominant frequencies in these intervals were greater than , yet the yellow intensity peaks occurred below this, the expected results were yellow peaks at frequencies ~ the centre of the frequency ranges given. For example, a peak at around between , a peak at between and one at around for the interval.

The exact same steps were taken to process two different brain signals and the resultant time frequency spectrums can be seen in figure 16. For both the signals the predominant activity appeared to be in the range, however it was harder to verify whether the time frequency spectra were accurate for these signals as no information on the dominant frequency ranges at each interval was given. The data was assumed to have been recorded over a 40 second period again but with a larger sampling frequency, . The larger was required as there were much larger frequencies present in these signals and so the Nyquist frequency needed to be larger to prevent aliasing.

**Figure 16:** The time frequency spectrum for two different brain signals.



# The Hilbert transform method was just one of several different ways to produce the time frequency spectra. Another method would be to use short-time Fourier transforms, in which the signal would be split into shorter segments all of equal time length. The Fourier transform would then be performed on each of the smaller segments separately to reveal their spectra, these would then be combined to show how the spectra change over time. Another method is the wavelet transform which is used because the Fourier transform must inherently compromise on one of time or frequency resolution. By using multiple wavelets, the wavelet transform achieves good time and frequency resolution that is required for an informative time-frequency spectrum.

# 4. Task 4

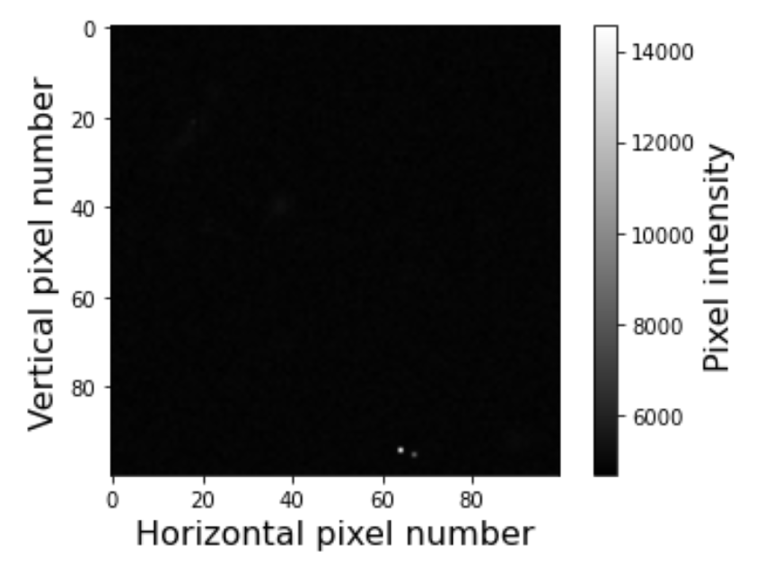
## 4.1 Theory and methods

## 4.2 Results and discussion

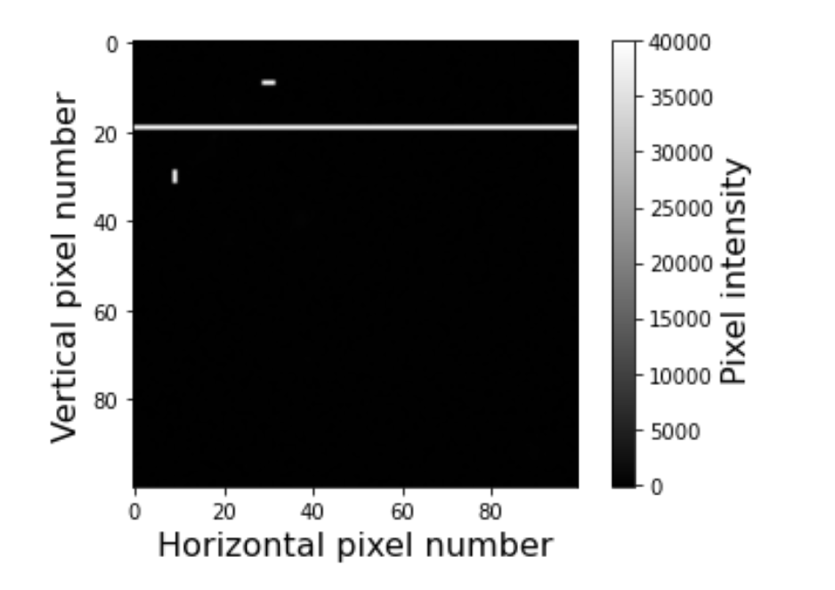
# 5. Task 5

## 5.1 Theory and methods

This task was based on 25 near-infrared images that had been taken sequentially, but each with the same exposure time. These images were to be processed before combining them together to simulate one long exposure photo. The first step was to remove known defective pixels; this was done by averaging the intensity values of neighbouring pixels and assigning the result to the defective pixel. As the data was corrupted for the pixel it was a fair to assume it would have a value similar to its surrounding, unlikely to have one bright pixel surrounded by very dark pixels, there would likely be a gradient of intensity instead. Most defective pixels occurred on a horizontal strip as seen in figure 17. Note that this plot was not the actual result of the defective pixels, but instead the values of the defective pixels have been set to very large (bright white) values to help visually indicate where in the image they occurred. The fact that the defective pixels mainly lie next to each other affected how the neighbourhood had to be determined. As including other defective pixels in this neighbourhood average would not be desirable, therefore vertical five-pixel strips centred on the defective pixel were used for the interpolation.

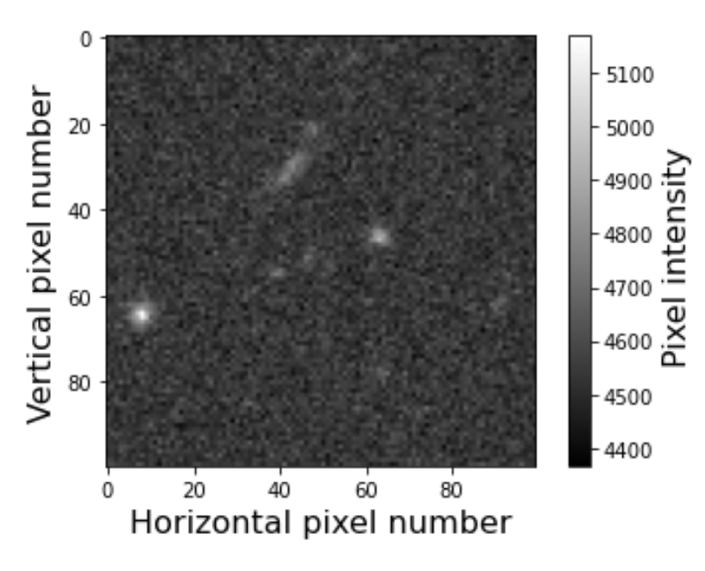


**Figure 18:** A plot of one of the images affected by cosmic rays.



**Figure 17:** A plot of the defective pixels in white and remainder in black to visually highlight their location in the images.

The next step of processing the images involved dealing with pixels hit by cosmic rays (CR). A pixel hit by a CR would record an intensity far above any of the pixels reacting to light, as seen by the white dots at the bottom of the image in figure 18. To identify these pixels, code was run to determine the max intensity value in every image. The lowest out of these 25 values was then defined as the threshold intensity for what determined whether a pixel was due to a cosmic ray or just a bright pixel. Any pixel above this threshold was then interpolated in a similar way to the defective pixels, except the neighbourhood was now a 3x3 matrix centred on the CR pixel. The result of removing these CR pixels can be seen in figure 19. Once these anomalous pixels had been removed the detail, initially hidden in the dark regions of the plot, became visible, highlighting the structures of interest within the images. For a couple of the images the original threshold set for determining CR pixels was too high, and therefore this detail had still not been revealed. Therefore, the threshold was manually lowered until these bright structures were visible in all the images. It was important to distinguish between what constituted a bright region in the sky and what was just a lower intensity CR pixel. The main difference was that CR pixels would be one or two bright white pixels surrounded by many dark pixels, whereas an actual light source would be many pixels with a bright centre that gradually faded towards the edges.

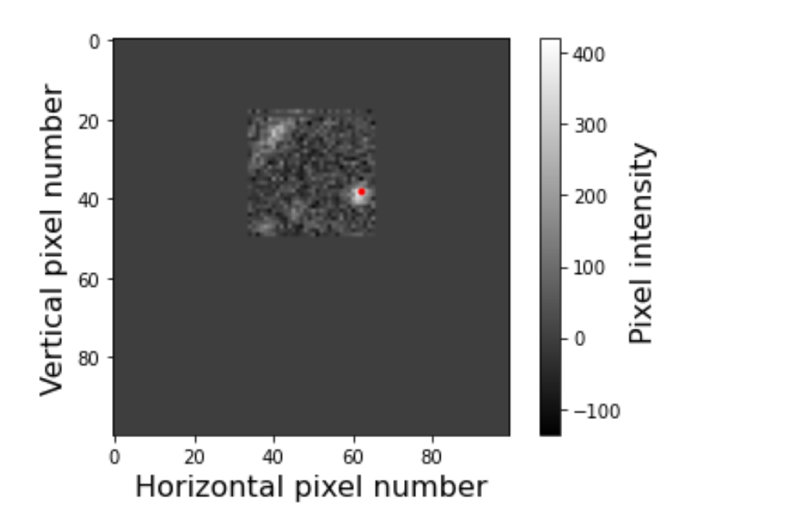


**Figure 19:** A plot of one of the images after the cosmic ray pixels have been interpolated.

Despite the exposure time being the same for each image, the brightness of the background sky varied from image to image. This needed to be corrected before the images could be combined as a darker region in one image might have a higher pixel intensity than a bright region in another image and then when they were combined the information about the bright region would be lost [7]. To correct for this, code was run that first determined the pixel intensity value that 90% of the pixels in an image were smaller than. As the bright regions of interest made up a small portion of the image and the large remainder was just background sky. The average was then taken of all the pixels below this threshold to give an average intensity of the background sky. This average was calculated for each image and the value then subtracted from all the pixels in the image to give the corrected values. The resultant images all had similar pixel ranges with a max pixel intensity of around 600. The threshold was determined by trial and error of different values until the optimal value of 90% was decided.

The spatial offset between images was the next issue that required solving. To quantify this an object needed to be tracked in every image and the amount its centre of mass moved recorded. To achieve this took some steps similar to the sky subtraction, except now the threshold was 99.99%. For a pixel image there would only be one pixel above this threshold and therefore the code returned the brightest pixel in each image, which was assumed to correspond to the centre of the brightest object in each image. However, plotting the location of this brightest pixel in each image highlighted that the same object was not being tracked in each image. This occurred as in some images there were multiple sources of bright light and therefore the brightest pixel would not correspond to the same object each time. To overcome this issue required visual interpretation of the images, in which a specific bright object (SBO) was chosen for the tracking. The movement of this object from image to image was then tracked by eye, and a general region that the object moved within estimated. The shape of this region was chosen to be a square of side length 32 pixels with a top left corner index , the size and location of this square was refined by trial and error. This refining process was once again done by eye and involved checking that the SBO was the only bright object within the square for every image. As if any other bright objects were within the square the algorithm could start tracking the wrong object again. The square region was created by creating an array of zeros with the same shape of the images and then setting the values within the square to their associated values from the image. Running the code to find the highest intensity pixel now tracked the centre of the same bright object each time, as seen in figure 20.

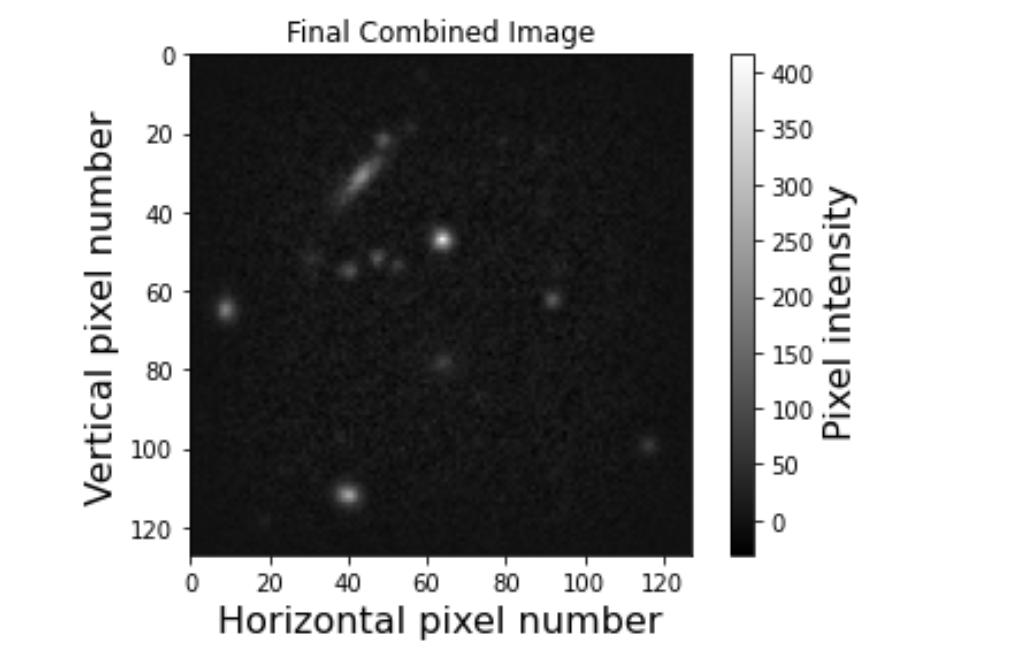
**Figure 20:** Plot of one of the images in which only a square region was being considered. The red dot indicates the highest pixel intensity in this region.



The indexes (coordinates) of the pixel corresponding to the centre of the object in each image were recorded. The reference centre was taken as the coordinates of the object centre in the first image, the spatial offset for the image was then calculated as the difference between the reference centre and the object centre of the image. This offset determined how much each pixel within the image needed to be translated by to align itself with the other images. To allow the images to be translated the arrays containing them needed to be extended, and so the maximum offset in each direction was calculated to indicate the amount they needed to be extended by. The largest offset in each direction was -27, i.e. an offset towards the top left corner of figure 20. To extend the array code was run that centred the image at the centre of the array and then increased the array by 27 in both directions for each dimension. This resulted in pixel images which was obviously larger than needed but it was the simplest way to code it, and the view of the images could be altered to ignore the empty space at a later point. Each of the images were then shifted by their respective offsets to align them in all in the larger image, if done correctly the centre of the object being tracked should occur at the exact same coordinate for all the images. This might not have occurred exactly as the movement of the tracked object in the sky, and therefore the offsets, did not correspond to integer movements of the centre. This would only result in the object centre being off by unit in each dimension at most and therefore was not an issue. The images were then combined into one larger image by calculating the mean pixel intensity at every coordinate, the result of which can be seen in figure 21.

## 5.2 Results and discussion

The resultant pixel image could be seen in figure 21, with a clear reduction in the signal to noise ratio after having processed the images. The central regions of the image were sharper as they were produced by the overlap of most of the images, whereas the edges only corresponded to the overlap of a couple images. This method was very successful for enhancing and maintaining visual information about the sky; however it would not be a suitable technique for obtaining precise quantitative data about the brightness of the objects within the image. This was due to the step in which the brightness of the background sky was subtracted from each image. To estimate the actual pixel intensity from an object in the final image, the average sky value could be calculated across all 25 images and then added back to the image to return some of this lost information.



**Figure 21:** The final combined larger image with a greatly reduced signal to noise ratio.

The main weakness with this method for processing the image was the reliance on judgement by eye to determine the region the tracked object was in. For only 25 images this was a manageable task, but if the task involved processing thousands of images, then an automated process for identifying the region would need to be developed. This could consist of a neural network to help classify whether a region contained the desired object or not.

# 6. References

[1] Amer, G.M.H. and Abushaala, A.M., 2015, March. Edge detection methods. In *2015 2nd World Symposium on Web Applications and Networking (WSWAN)* (pp. 1-7). IEEE.

[2] Feng, B. and King, M.A., 2013. Estimation of 6-degree-of-freedom (6-DOF) rigid-body patient motion from projection data by the principal-axes method in iterative reconstruction. *IEEE Transactions on Nuclear Science*, *60*(3), pp.1658-1663.

[3] Andrei, O., 2006. *3D affine coordinate transformations*.

[4] Sneddon, I.N., 1995. *Fourier transforms*. Courier Corporation.

[5] Matei, R., 2013, July. Parametric 2D Gaussian circular band-pass filters. In *International Symposium on Signals, Circuits and Systems ISSCS2013* (pp. 1-4). IEEE.

[6] Kschischang, F.R., 2006. The hilbert transform. *University of Toronto*, *83*, p.277.

[7] Soto, K.T., Lilly, S.J., Bacon, R., Richard, J. and Conseil, S., 2016. ZAP–enhanced PCA sky subtraction for integral field spectroscopy. *Monthly Notices of the Royal Astronomical Society*, *458*(3), pp.3210-3220.