Report title

Jordan Moshcovitis

November 6, 2020

Abstract

Computer vision identifies a field of computational science that has application in a large variety of commercial and academic subfields. This project investigates one application to the analysis of image data for specific nonuniform spatial features. Spatial cross correlation, spectral cross correlation and convolution are investigated as possible image processing techniques. Further real-world spatial distances are determined from images from depth maps. These techniques are combined to applying statistical. Find statistical information about moth eye SEM images. The defect quality and other properties are determined using computer vision algorithms and feature extraction. The final program gives an output of statistics for the moth eye SEM image.

Contents

List of Figures

List of Tables

1

Introduction

1.1. Overview

This project develops pattern recognition and aspects of image processing used in computer vision. Image data, and the way in which it may be interpreted, offers useful information to analyse complicated visual events[1]. Notably, implementation of algorithms which may replicate natural processes for distinguishing significant features for spatial characterisation of surroundings, environment, or objects[2]. Such an approach is called Computer Vision and serves to mimic the process by which human eyes evaluate spatial depth and unique spatial features from a pixel level. Furthermore, greater information may be determined using spectral decomposition techniques which surpass the processing capability of natural eyesight.

1.2. Experiment: Cross Correlation in 1D and 2D

A first set of experiments are implemented to investigate and evaluate standard pattern matching approaches using cross-correlation. Cross-correlation[3] is a widely used in statistical image processing as a method for image analysis. The correlation coefficient is a weighted measure of the similarity between any two data sets. The method is used to evaluate the degree of similarity of multidimensional signals, image patches, feature detection and pattern matching. Based on simple convolution, calculation of the correlation coefficient is a computationally direction

[1]Huang, T. Computer vision: Evolution and promise. CERN EURO-PEAN ORGANIZATION FOR NUCLEAR RESEARCH-REPORTS-CERN(1996), 21–26

[2]Szeliski, R.Computer vision: algorithms and applications. SpringerScience & Business Media, 2010

[3]It is an elementary approach to match two image patches, for feature detection as well as a component of more sophisticated techniques... There have been many works in literature which use crosscorrelation for matching `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.675.1379&rep=rep1&type=pdf` `https://ivpl.northwestern.edu/wp-content/uploads/2019/02/Digital-Signal-Processing-Handbook.pdf`

method that benefits from Fourier transformations that can reduce time complexity of operations. Further it may be implemented across unrelated data sets to find an unbiased measure of similarity. Cross-correlation is often combined with other image manipulation methods such as filtering to improve the resolution of the correlation coefficients.

The concept of cross-correlation has been developed in two distinct fields: signal processing and statistics. In the area of signal processing, the cross-correlation function can be used to transform one or more signals so that they can be viewed with an altered perspective. For instance, cross-correlation functions can be used to produce plots that make it easier to identify hidden signals within the data. Cross-correlation functions provide the basis for many more sophisticated signal-processing procedures as well. Digital imaging techniques also rely heavily on cross-correlation procedures, but these methods are not covered in the chapter.

1.3. Experiment: Patterns in Stereovision

A second set of experiments investigate the parallax nature of captured real world images when digitised. A depth map is a 3D reconstruction of an image from the binocular disparity map of a set of images, or in some advanced implementations, a monocular disparity map. Real world intensity and position is approximated from the perspective projection using distance calculations to pixel coordinates.

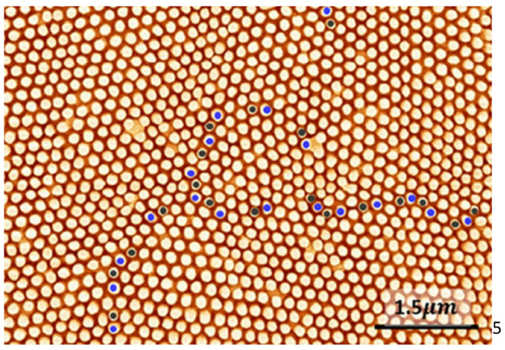

Figure 1.1. Moth eye[4]

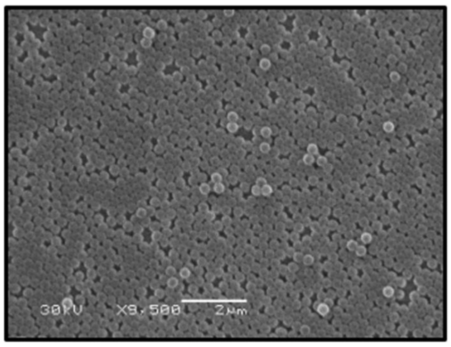

Figure 1.2. Textured Diamond

1.4. Application: Moth-eye SEM image anlaysis

Ultimately, this project ultimately aims to extend the investigated cross-correlation and computer vision methods to analyse experimental data from Scanning electron microscope images. Advancement in nano-lithographic techniques and physics of metamaterials surfaces have led to development of certain surfaces with unique intrinsic properties. Moth-eye surfaces are one

[4]https://pubs.acs.org/doi/10.1021/acsomega.0c02314

nano-scale surface that because of the arrangement of the particles the surface takes on antireflective properties. Thus, it is of interest to investigate a rapid way to estimate and characterise features.

Images are processed and then analysed using Fourier transform methods and filtering to extract information relating to irregularities of the surfaces. The analysis will implement correlation to determine regions where there are grain non-uniformities and defects in the ordering.[5]

[5]More images include in the appendix at the end.

2
Theory

A simple way of determining the relationship between two separate datasets is by comparison of patterns with template data. Particularly, by a method of single or multi element template matching via correlation calculation. Correlation, also covariance, is a measure of the strength of a linear relationship between two or more quantitative variables. So, given two independent variables, that both vary in a way such that they are both related to a third in the same way such that their behaviour is positively correlated as a result of their individual relationship with the third variable. Thus, it is useful to derive this relationship for data that is to be compared. This relationship is quantified as the cross correlation between sets of data. Computationally, two classical approaches to find the cross-correlation, spatial and spectral, are implemented in this project.

2.1. Pattern matching[1]

2.1.1. Cross correlation

Statically, cross correlation provides a measure of association between datasets. The correlation value may be determined from a normalised unnormalized weighting of the data set. Quantitatively it provides the degree of strength of a linear relationship between two or more quantitative variables. The value of the cross correlation is determined in general by (2.1) and may be generalized for higher dimensions.

[1]https://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=1045&context=kin_pubs https://lib.dr.iastate.edu/kin_pubs/46

2.1.1.1. Convolutional Neural Networks (CNN)

Convolutional neural networks (CNN) are named after the mathematical operation convolution: The convolution of a function f with another function g is defined as

$$(f*g)(t) = \int_{-\infty}^{\infty} f(x)\cdot g(t-x)dx \tag{2.1}$$

$$(f*g)(t) = \sum_{x=-\infty}^{\infty} f(x)\cdot g(t-x) \tag{2.2}$$

in the continuous or discrete domain.

Convolution is often encountered in the context of image processing, where f is the intensity of a given pixel and g is a 2-dimensional weighting function that, is called kernel. g is usually non-zero only for a few values in the close neighbourhood to the central pixel and therefore the sum has to be computed only over those values instead of the whole image. The kernel g is often defined as a small square matrix whose size is called the kernel size k.

Depending on the values of the kernel, convolution can be used for several image manipulation operations: If the weights are all positive and for example reflect a Gaussian function with its maximum at the centre, a blurring effect is achieved by..

Where (2.1) and (2.2) are the two different data sets, n is a discrete element within the data, is the shifted position, and $c(s)$ is the correlation coefficient. The position where the coefficient is largest or smallest, s, identifies the shift position for which one data set most greatly matches the other. The effect of the offset position is visualised below (figure 2.1). The position of greatest correlation appears as a peak in the graph of correlation coefficient plotted against shift position (figure 2.2).

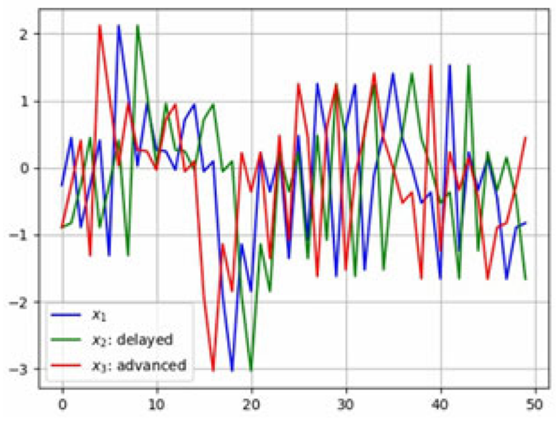

Figure 2.1. Code to generate in appendix A

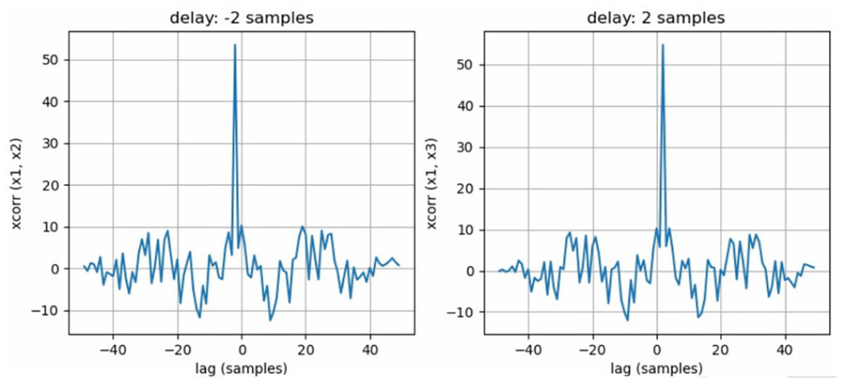

Figure 2.2

2.1.2. Normalised cross correlation[2]

Direct correlation, unless autocorrelation, is dependent on the amplitude of the data compared. This dependency is eliminated by normalising the correlation coefficient, referred to as the normalized cross-correlation. Normalized cross correlation is used to evaluate the degree of similarity between any two data sets. It has the advantage of less sensitivity to linear changes in the amplitude of the quantitative variables under comparison.[3]

The normalised cross correlation is divided by the product of the autocorrelation magnitude as this is the largest that correlation value the data may have. The autocorrelation of discrete-time signals is a useful mathematical concept that helps measure the maximum or total energy of a signal. Calculation results have a fixed range 0 to 1. If there is a large difference in amplitude between the matched regions, then the modified normalization potencies the correlation to zero (i.e. effectively uncorrelated). Finally, contrast normalization allows matching with regions that are principally uniform except for random changes or texture variations.

2.1.3. Spatial

Spatial cross-correlation is a measure of the strength of the relationship between two temporal data sets. It is found by convolving each data set across each other that may expose patterns in the input data. This method is useful for finding the overlapping pattern over a periodic signal covered by noise. Specifically, this is implemented for two signal files that are analysed in this section. Since the two signals compared are given to be different, cross correlation is implemented. The cross correlation will reveal how much one signal has been shifted, for the piecewise continuous case of two signals with appropriately short sampling, to be congruous

[2]http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.675.1379&rep=rep1&type=pdf https://www-scientific-net.eu1.proxy.openathens.net/AMR.860-863.2800.pdf

[3]https://imagemagick.org/docs/AcceleratedTemplateMatchingUsingLocalStatisticsAndFourierTransforms.pdf

with another comparative signal. Defined formally:

$$c(s) = \sum_{n=-\infty}^{\infty} f_1^*[n] f_2[n+s], \tag{2.3}$$

The similarity of the two signals is given thus by the correlation coefficient c for a given lag s between the two signals.[4]

Further, this is made more general by the normalized cross-correlation in 2D. The normalized cross-correlation forces uniformity across the data sets where there is a wide spread of intensities/amplitudes for the data.

$$NCC(h,k) = \frac{\sum_{i,j}\Big(S(i,j)-M_S\Big)\Big(\big(L(i+h,j+k)-M_L\big)\Big)}{\left\{\sum_{i,j}\Big(S(i,j)-M_S\Big)^2 \sum_{i,j}\Big(\big(L(i+h,j+k)-M_L\big)\Big)^2\right\}^{0.5}}. \tag{2.4}$$

The Normalized Cross Correlation is confined to a range from the inclusion of the denominator so that the resulting correlation ranges from -1 to 1. A perfect match has a value of 1. $ML \equiv ML(h,k)$ is the mean of the subsection of the large image at offset (h,k), N is the number of pixels in the small image and $NCC(h,k)$ is the normalized cross correlation metric at offset (h,k). The numerator is essentially a simple cross correlation but using a zero mean small image and zero mean subsections of the larger image. The mean subtraction mitigates amplitude differences between the two images. The denominator is included so that the resulting correlation metric ranges from -1 to 1.

In spatial cross correlation, a weighting is needed to be derived at each alignment. This is computationally intensive as the time complexity for this process is N^2 where N is the input size of the larger array. The process of cross-correlation can be performed more efficiently for large inputs by transforming the inputs into the frequency domain.

2.1.4. Spectral

The Fourier transform produces another representation of a signal, specifically a representation as a weighted sum of complex exponential terms of the frequency spectrum of the data. Spectral cross correlation utilizes the same principles from which spatial cross correlation is are based, however operations between two data sets are made only in frequency space.

A Fourier transform is a signal processing technique which decomposes a function on the time domain into a function in the frequency domain. Times series, spatial data, is converted into frequency data using the Fourier transform. Fourier transforms decompose the frequencies of periodically sampled data and map them to corresponding amplitudes in the frequency domain. An approach not using normalization and directly computes the simple cross correlation,

[4]Goodman, Fourier optics.

$C(i,j)$, using forward and inverse Fourier transforms. A basic principle of Fourier transforms is that convolution in the spatial domain is equivalent to multiplication in the frequency domain. Likewise, correlation in the spatial domain is equivalent to multiplication in the frequency domain using the complex conjugate of one of the transformed arrays. Thus, either convolution or cross correlation can be used to perform spectral cross correlation. The cross correlation of $f(t)$ and $g(t)$ will have the same output as the convolution of $f(t)$ and $g(t)$ but reversed. This results in the formula below:

$$eqn? \tag{2.5}$$

As such the discrete convolution of two series signals may be determined by finding the inverse of product of the Fourier transforms of both signals[5]

$$(u * v)(\tau) = \mathcal{F}^{-1}\Big\{\mathcal{F}\{u\} \cdot \mathcal{F}\{v\}\Big\}. \tag{2.6}$$

This relies on the property that multiplication in the frequency domain is equal to convolution in the spatial domain. Because of this it is easier to apply multiple filters as products in the frequency domain, where they would have been successive convolutions in the time domain.

Since time series data is sample discretely, a summation analogue of the continuous Fourier transform is used (The Discrete Fourier Transform can be expressed as):

$$\mathbf{F}(n) = \sum_{k=0}^{N-1} f(k) e^{-j2\pi nkN} \quad (n = 0, \dots, 1; N-1) \tag{2.7}$$

In practice the discrete Fourier transform (DFT) is implemented typically using the Fast Fourier Transform (FFT).

In contrast to spatial correlation, the normalised cross correlation does not have a minimal frequency domain expression. It cannot be directly computed using the more efficient FFT (Fast Fourier Transform) in the spectral domain. Its computation time increases dramatically as the window size of the template gets corpulent.

2.2. Stereo vision

2.2.1. Depth mapping[6]

Stereovision aims to reproduce the depth determination capability of human vision. A depth map records the distance information with respect to a viewpoint in a 2D image of digitised real-world coordinates. The proposed algorithm uses two images of the same scene. Pixel to pixel matching is used between two images, left and right, to find the corresponding point in

[5] Proof in appendix.

[6] https://projet.liris.cnrs.fr/imagine/pub/proceedings/ICIP-2009/pdfs/0002357.pdf

the same scene from different views. As such, the same point is identified in 3D from the 2D image information. The left and right images are assumed to be taken with two digital cameras. They capture images of the same scene, at the same time. Given some horizontal displacement between the cameras, which may be approximated to that the spacing between two eyes. Given the geometric description of the location of the camera, separations and pixel location, a depth map may be created using a method binocular disparity.

The disparity value of a point is often interpreted as the inverse distances to the observed objects. Thus, disparity is inversely proportional to Depth. Therefore, finding the disparity is essential for the construction of the depth map. Darker regions in the Depth-Map are created for signifying that an object is far away, and this darker colour gradually decreases to brighter with decrease in depth and finally becomes white for closer objects.

2.3. Practical considerations

2.3.1. Data set size

Along with physical computational power, the efficiency of a cross correlation algorithm will vary with size of the data being compared. In the case of image data, which is the primary focus of the project, the degree of computational complexity will depend upon image size and resolution. The nature of image data is such that the size of the data set, a digital image $a[m,n]$, described in a 2D discrete space is derived from an analogue image $a(x,y)$, in a 2D continuous space through a sampling process. The process is often referred to as digitisation. The digitisation relates image data to pixel location. The data sets used in this study consider one- and two-dimensional analogue information that is already pre-digitised in text or image-based formats.

2.3.2. Time complexity[7]

Direct calculation of NCC is computationally intensive and the time cost is proportional to the region size. Assuming the average size of the matching regions is R, the computational complexity to match two images with image size of M and disparity range of D is $O(M \times R \times D)$. Since M, R and D are usually large. Since stereo matching usually needs to match each pixel across a disparity space for a whole image, the computational efficiency of the pixelwise similarity is necessary for a fast depth determination.

2.3.3. Spatial

The spatial cross-correlation methods considered in this project utilises convolution to relate the elements of the two compared data sets. The computational complexity for a $N \times M$ convolution

[7]http://hit.skku.edu/wp/wp-content/uploads/1-s2.0-S1051200410000047-main.pdf https://projet.liris.cnrs.fr/imagine/pub/proceedings/ICIP-2009/pdfs/0002357.pdf

kernel implemented in the spatial domain on an image of $N \times M$ is $O(N^2)$ where the complexity is measured per pixel on the basis of the number of multiplies-and-adds(reference).

2.3.4. Spectral

The Fast Fourier Transform, which is first introduced by Cooley and Tukey, 1965, the Cooley-Tukey algorithms. The time upper bound time complexity and operation counts is $O(N\log(N))$. Thus the FFT is an optimized algorithm that reduces the time complexity of the DFT from $N\hat{}4$ to $N\log N$.[8] The computational complexity per pixel of the Direct Fourier approach for an image of $N^2 \times N^2$ and for a convolution kernel of $N \times N$ is $O(\log N)$ complex MADDs independent of N.

[8] Reference.

3

Cross Correlation in 1D and 2D (6 pages)

3.1. 1D singal offset

3.1.1. Time series

In the case of time series, cross correlation produces a measure of the temporal similarity of the data sets. A correlogram plot of the cross-correlation values at each lag is useful for extracting noisy signals and synchronizing signals. Cross-correlations between the signals will peak at the lag in which the signals overlap. The lag is the number of time periods that separate the two time series. The delay is determined through cross-correlation of the signals. The peak cross-correlation value occurs at the lag where the signals have the greatest similarity. Figure 3.1 shows the correlogram between the two signals with what is likely random noise introduced, possibly from the environment or recording equipment. The noise conceals the signal pattern, however the correlogram shows a distinct peak cross-correlation at a zero lag. This indicates that the two signals are synchronized. For example, if these signals were from a radar, the peak occurs at a particular lag that is transformed into a transmission time through multiplying by the sampling period. The distance can be determined through multiplying the transmission time by the transmission velocity.

3.1.1.1. 1D signal application / data[1]

The data used in this experiment is discrete time series data. A discrete-time signal $x(n)$ is defined as a series of time instants corresponding to a sequence of quantities. Proakis et al. state that n in discrete signals represents the index of the discrete-time instants as the independent variable. This allows the signal to become a function of an integer variable. Therefore,

[1]http://dspace.calstate.edu/bitstream/handle/10211.3/190191/GonzalezAdrian_Thesis2017.pdf?sequence=1

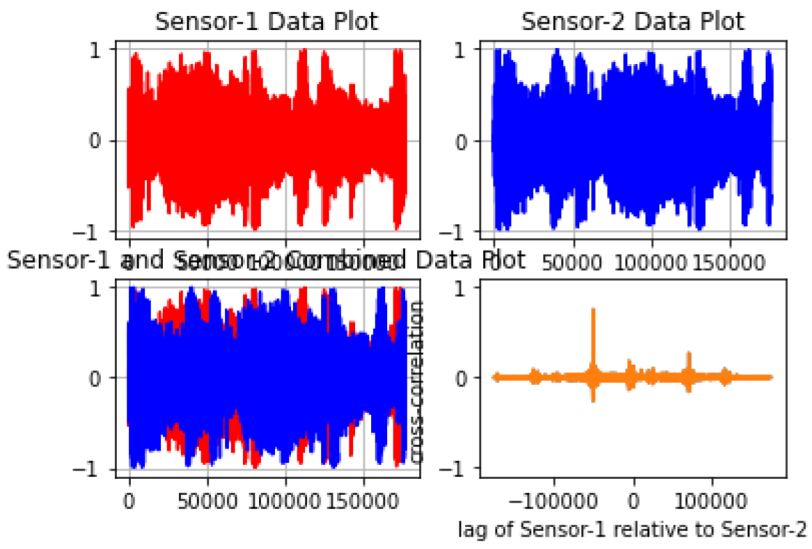

Figure 3.1

$x(n)$ refers to the nth-sample, and can be represented mathematically by a sequence of real or complex number.

The signals used in the 1D case is are analogue time series, with constant sample period, T, the time between successive samples. The reciprocal of the sampling period is known as the sampling rate Fs, which represents the samples per second.

The signal data is stored in a .txt file, (UTF-8 encoding) with data point separated with a carriage return operator. The function *read_file* opens the file from the a specified location in the directory, then l=using list comprehension, removes the header using the *islice(data, 1,..)* library function and *.strip()* to separate the numerical values from the carriage return.

3.1.1.2. Analysis – temporal

Applied to temporal 1D cross correlation, the theory outline in Chapter 2 is implemented to find the distance between two speakers given two output signals, respectively.

The chosen pattern and template signals are stored separately as lists and passed into the *find_offset* function. *find_offset* determines the shift by calculating the index of the maximum value determined from a list of cross correlation coefficients. To determine the cross-correlation coefficients, the template array is zero-padded with the length of the pattern array -1 at either end. Then the product of the overlap between the pattern array and non-padding elements of template array for an iterated offset position of the index. The corresponding cross-correlation function will show a maximum at the index where the overlaps are most congruent. This is returned alongside with the value of the index.

$$r = \frac{1}{N}\sum_{i=1}^{i=N}\left(f(i)-\overline{f}\right)\left(g(i)-\overline{g}\right) \tag{3.1}$$

The normalised cross correlation of two padded signal arrays is found from the maximum of the convolution between the two time series signals and the corresponding index. For the two-signal data, the offset of the maximum cross correlation value indicates the number of sample periods between the beginning of each signal file and the maximum overlap.

- Off-Set $= -50082$
- Off-Set Time $= -1.136$
- Distance between two sensors $= 378.17$ meters

However, the implementation takes two hours to compute.

3.1.1.3. Analysis – Spectral

To improve performance, spectral cross-correlation was then employed to reduce the computation time. The code implements the spectral decomposition of the two signal files as outlined above, to determine the cross-correlation coefficient and associated lag. The high outlying high amplitude-low frequency noise and general sensor related noise is reduced when applying a low pass filter. The dominant frequencies are more easily apparent[2]. As all the signal data is real valued, the related matrices are Hermitan. So the frequency spectrum is consequently symmetric, and may be mirrored or removed above the Nyquist frequency of Fsample/2 and filtered using a low pass filter.

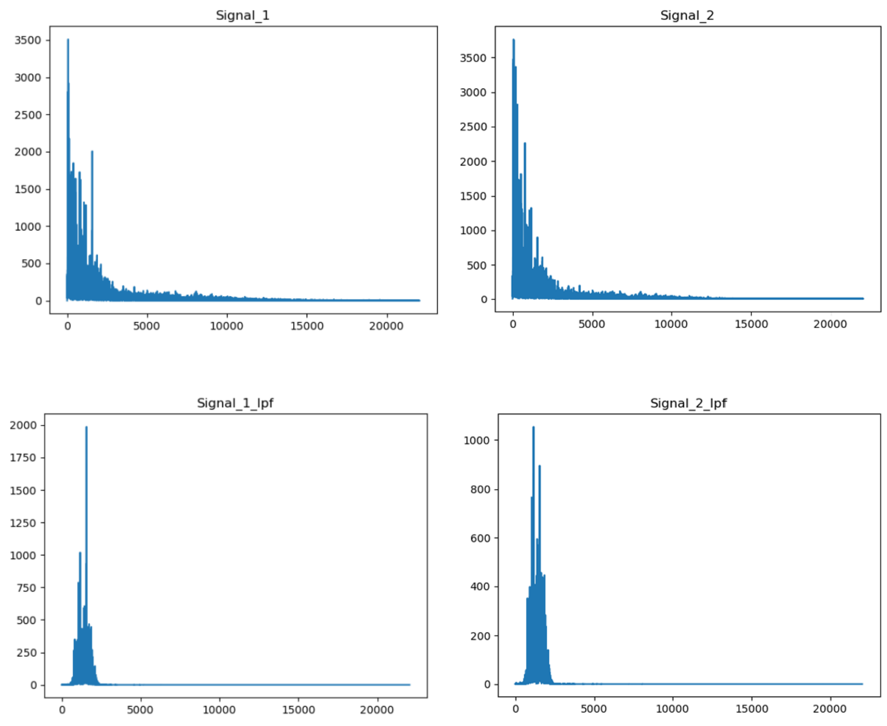

Figure 3.2

[2]R. N. Bracewell and R. N. Bracewell, The Fourier transform and its applications

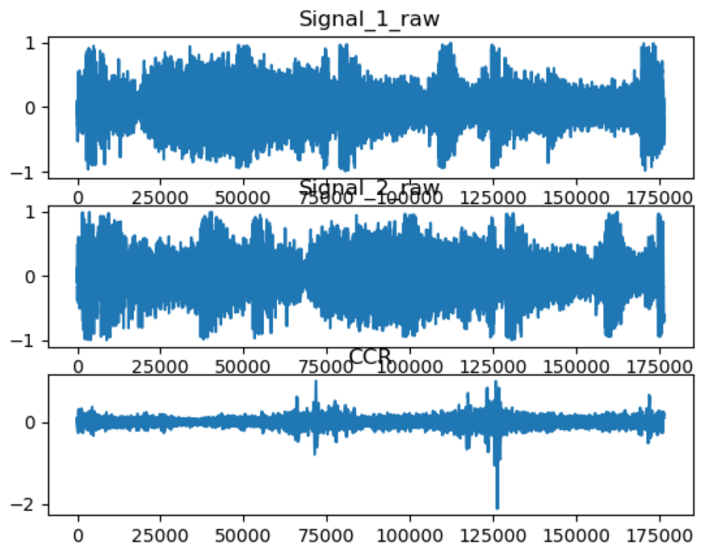

Figure 3.3

The resulting values:

- Freq. $= 44100$
- Off-Set $= -50081$
- Off-Set Time $= -1.136$

Distance between two sensors $= 378.16$ meters

3.1.1.4. Evaluation – Temporal

For fast implementation, the pattern and template lists are converted into numpy arrays inside the *n_corr* function. Notably implementation is very slow due to the use of for loops in the *n_corr* and *calculate_energy* functions. These enforce a minimum time complexity of N^2, and greater. To reduce the calculation time, cached valued are used for the determination of the sum of the squared elements of the *pattern* array. Initially, the calculation was performed for every time the *calculate_energy* function was called *i* many times[3]. Caching the value reduced the loop time for *n_corr* by half. However, the calculation time was still .5 s per loop in *n_corr*. It was found when disabling *calculate_scores*, the computation time was significantly reduced. This is likely due to the zeros that are multiplied each time the pattern and template are passed. An if statement is used to require only non-zero values are multiplied. This reduced the loop time again by half. The calculation was compared to the a library implementation of the cross correlation using *numpy.correlate*. The run time in comparison was 3.01 s. The difference is firstly due to the optimisation of the numpy libraries, but also because the second method does not pad the second array. Thus, the convolution is faster. It may be possible to speed up the calculation time again by using Just in Time (JIT) decorators, like numba or PyPy.

[3] *i* many times to iterate over the size of the correlation array, = template_padded – pattern

These changes resulted in a 64% decrease in run time of the *n_corr* loop, but at the cost of proper normalisation techniques. The offset was still calculated to be 389 m but given a significant speed decrease, and the loss of proper normalization, this method was abandoned.

3.1.1.5. Evaluation – Spectral

To improve performance, spectral cross-correlation was then employed to decrease the run time. Spectral cross-correlation initially took 2.03 (min:sec) to perform. This technique should be much faster than temporal cross-correlation, so it was clear that something was impacting performance. After researching, it was found that numpy.fft uses the Cooley-Tukey algorithm to calculate the Fourier transform. As it is a divide and-conquer algorithm, it runs most optimally on inputs of a power of two in size, and least with sizes or prime numbers. The signal lengths are prime numbers, and so by padding these inputs with zeros to the closest power of two, the run time was significantly faster. After padding, the run time to find the distance between the two sensors was: 0.15s. The computation time compared to the Temporal cross correlation was noticeably faster. This is due to: Temporal cross correlation must repeat the same operations for each slice. Temporal cross correlation consequently also considers the offset of the neighbours of the slice. So the spectral cross correlation is thus faster, and need only be padded if signals are of different length.

Spectral Cross correlation	Temporal Cross correlation
• Off-Set = -50081 • Off-Set Time = -1.136 • Distance between two sensors = 378.16 meters	• Off-Set = -50082 • Off-Set Time = -1.136 • Distance between two sensors = 378.17 meters

3.1.2. 2D pattern finding

3.1.2.1. Intro

A simple way of determining relationships between images is by comparison of patterns with template data. In this experiment a method of pixel-based template matching via correlation calculation is used.

3.1.2.2. Test/Mean shift

Shifting the value of the pixel intensity spectrum has the effect of improving the cross-correlation algorithm accuracy and efficiency. In comparison of mean-shifted and non-mean shifted, the mean shifted speed and accuracy was found to perform better. This was apparent in the case of both the spatial cross correlation and spectral convolution approaches. In the case of the spatial correlation, the speed up was 40 seconds, whilst marginally more efficient 0.022 s for the spectral approach. (figure). The effect of the zero-mean is to redistribute the intensity over a positive and negative range. In the case of a greyscale image, ranging from black to white, the dark regions of the images will now have a much weaker correlation to brighter regions of equal amplitude, but now opposite sign.

In this experiment, a “rocketman’ was searched for in a larger template image to investigate 2D cross correlation techniques.

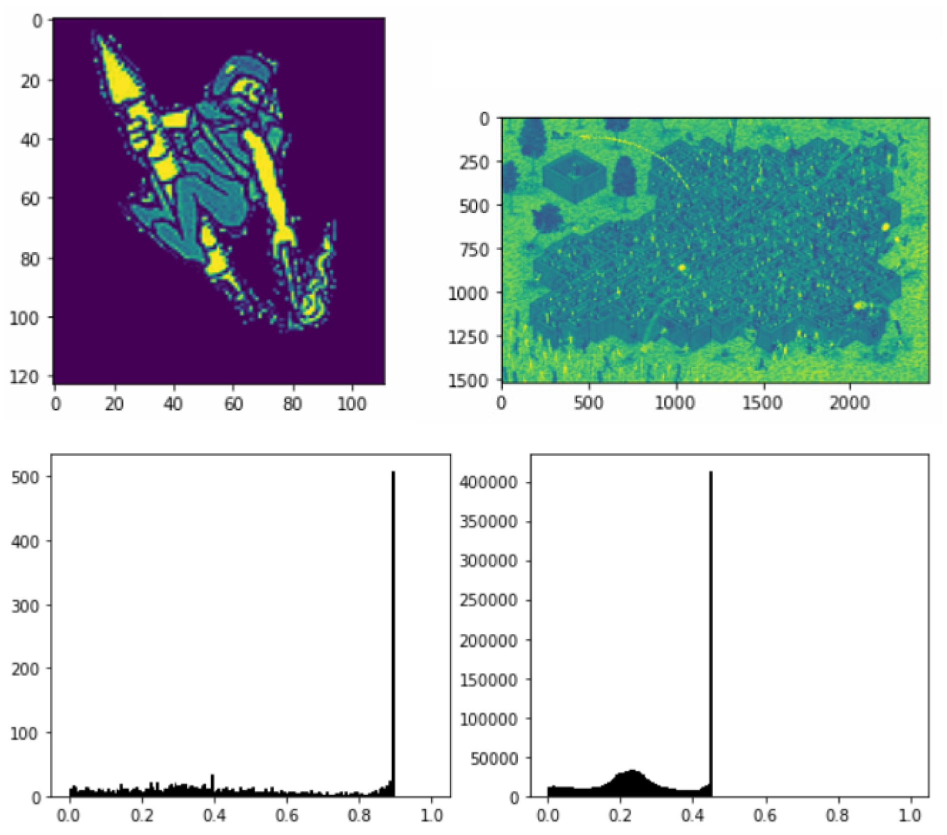

Figure 3.4

3.1.2.3. Application/Data

The images that are used are formed by a matrix of pixels. The pattern and template image information is stored in tensor-like array of intensities over three colour channels, (RGB) and one transparency channel (A). Each channel list represents a pixel with three components of colour, stored at 8 bits[4], ranging from 0 to 255. To determine the scalar cross correlation value, the images are converted into greyscale by finding the relative intensity of grey from a color table(reference). The functions m*atplotlib.image* and *conert_grey()* rescale the UTF-8 data from each channel to float32 values between 0.0 and 1.0. The weights used in the conversion are calibrated for contemporary CRT phosphors: $I = 0.2125[R] + 0.7154[G] + 0.0721[B]$.

3.1.2.4. Spatial – Analysis

The approach to find the location of a pattern image inside a larger target image can be implemented by extending the method of the 1D spatial correlation. the pattern is moved one pixel at

[4]Human eye purportedly can only see 8 bits of colour. Reference

a time over a larger image. The output pixel value is the weighted sum of the input pixels within the window where the weights are the values of the cross-correlation assigned to every pixel of the window itself. The window with its weights is called the convolution kernel. For each shift position, the cross-correlation coefficient is calculated pixel by pixel between the small image and the correspondingly sized region of the larger image.

In a similar way to the 1D spatial convolution, the template image is padded at either end of its coordinated axes with the length of the pattern -1. The cross correlation is then found by *n_corr2d* by sliding the pattern across the padded template and finding the sum product of every instance of the overlap 2D arrays over the non-padded region of the template image.

The *find_offset* function will then return the coordinated which corresponds to the largest correlation value given by the position of the top left corner of the pattern image array.

The pattern image is found within the template image with top left corner coordinate of $(x,y) = (529,983)$.

The first technique employed was spatial cross-correlation. The smaller pattern image was passed over the template image and a correlation value determine for each window overlap position.

3.1.2.5. Spectrum – Analysis

For simple cross correlation, the Fourier transform procedure is as follows. The images are read and converted into greyscale as in the spatial cross correlation script. Within *ccr_2d* the smaller pattern image is padded with zeros at the bottom and right sides to reshape it to the size of the larger template image. The FFT is then applied to both images, along rows then columns in this way twice to transform information about the neighbour elements as well as the frequency. Thus, a full spatial description in the 2D array of the image is found for any given channel. The phase information is removed by taking the complex conjugate of one of the transformed matrices before evaluating their product. The transformed arrays are multiplied elementwise. Last, the inverse Fourier transform is taken to return the convolution of the two images in the spatial domain. This process is much faster than doing the unnormalized correlation in the spatial domain.

Fourier transform of $f(t)$, Fourier transform of $g(t)$, element-wise multiplication, Inverse Fourier transform of the result. For this reason, using spectral cross correlation should dramatically cut down running time of the python programs that will be created.

Where F is the Fourier transform and $*$ denotes the complex conjugate. By taking the complex conjugate of one of the inputs we are essentially performing the time reversal of that input, thus the output functions would now be the same as for convolution.[5]

[5] https://imagemagick.org/docs/AcceleratedTemplateMatchingUsingLocalStatisticsAndFourierTransforms.pdf

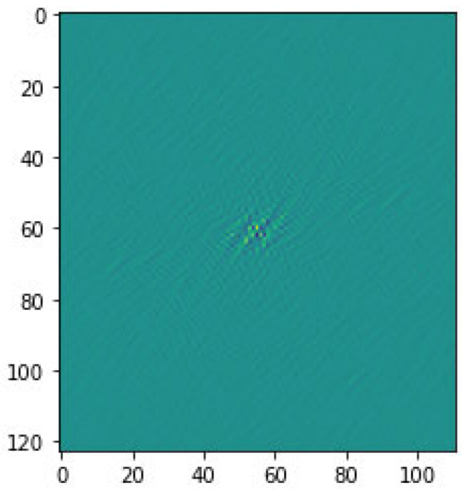

Figure 3.5

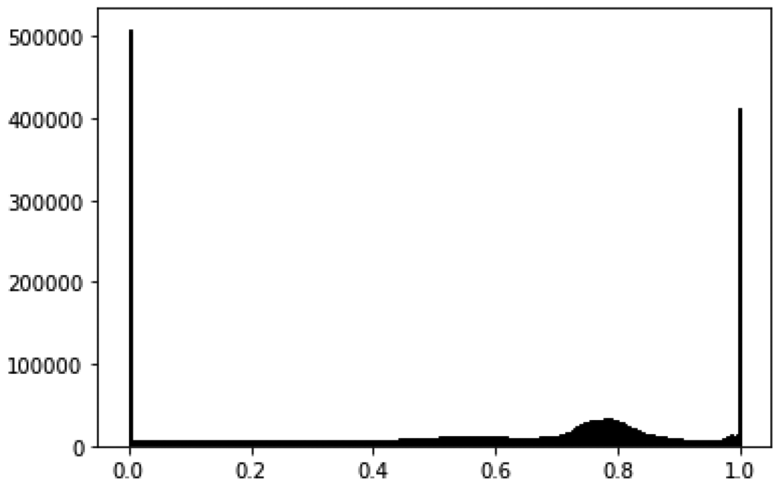

Figure 3.6

Because the image in the Fourier domain is decomposed into its sinusoidal components, it is easy to examine or process certain frequencies of the image, thus influencing the geometric structure in the spatial domain. In most implementations the Fourier image is shifted in such a way that the DC-value (i.e. the image mean) $F(0,0)$ is displayed in the center of the image. The further away from the center an image point is, the higher is its corresponding frequency.

3.1.2.6. Spatial – Evaluation

- Typically, normalised cross correlation of for image searching is computational slow. The calculation time for this implementation was 40.2 s.
- Normalized cross correlation, as described by equation (1), is computationally intensive and slow. Part of the complexity has to do with evaluating the numerator correlation in the spatial domain when the template image is large. The other aspect that adds to the

complexity is the computation of the mean and standard deviation of each subsection of the larger image.

- Improvements may be made by using more robust algorithms which take statistical samples of the template image region to reduce the size of the search region. [6]

3.1.2.7. Spectral – Evaluation

The method is much faster than spatial correlation since the number of operations is $N\hat{}22\log N$ not $N\hat{}4$. This is because of the implementation of the FFT. If the DFT were used, it would require a double sum over the indices for each offset position. The nested for loops would be as below:

Algorithm 1:

```
Def (explicit_correlation(image, kernel): This is O(n∧4))
hi, wi= image.shape
hk, wk = kernel.shape
image_padded = np.zeros(shape=(hi + hk - 1, wi + wk - 1))
image_padded[hk//2:-hk//2, wk//2:-wk//2] = image
out = np.zeros(shape=image.shape)
for row in range(hi): do
    for for col in range(wi): do
        for i in range(hk): do
            for j in range(wk): do
                out[row, col] += image_padded[row + i, col + j]*kernel[i, j]
return out
```

This is for RGB colour images, so will take FFT three times. Above algorithm is for a single channel.

3.1.2.8. Frequency spectrum features

3.2. Conclusions

A ’rocketman’ was located within a complex image to demonstrate the accuracy of these techniques and the speed of spectral cross-correlation. This can be seen in figure 3.4. Both spatial and spectral cross-correlation correctly located the rocketman within the search area, but there was a drastic difference in runtime. The spectral cross correlation took less than one second. As both techniques produced identical results, spectral crose-correlation was chosen for all remaining alignment tasks in this document.

[6] https://imagemagick.org/docs/AcceleratedTemplateMatchingUsingLocalStatisticsAndFourierTransforms.pdf