INVESTIGATIONS ON THE METHODOLOGY OF PHASE EXTRACTION BY ANALYSIS OF SINGLE INTERFEROGRAMS USING HILBERT TRANSFORMATION

A Project Report
submitted by

Partha Pratim Basumallick
PH12C023

in partial fulfilment of the requirements
for the award of the degree of

Master of Science
in
Physics

under the guidance of

Prof. A. R. Ganesan

Department of Physics
Indian Institute of Technology, Madras
May, 2014

Thesis Certificate

This is to certify that the project work titled "Investigations on the Methodology of Phase Extraction by Analysis of Single Interferograms using Hilbert Transformation" submitted by Mr. Partha Pratim Basumallick (PH12C023) of the Department of Physics, Indian Institute of Technology (Madras) has been carried out under my supervision and guidance during the academic year 2013 – 2014. The contents of this project report, in full or parts have not been submitted to any other Institute or University for the award of any degree or diploma.

Place : Chennai – 600036
Date :

Prof. A. R. Ganesan
Applied Optics Laboratory
Department of Physics
IIT Madras

Acknowledgments

I humbly take this opportunity to express my gratitude towards my teacher and guide Prof. A. R. Ganesan without whose kind and invaluable mentorship I would have barely progressed in my endeavour. It is under his wing that I discovered the charm and beauty of modern research topics in Optics, the fascinating traits through which the subject is evolving by the dint of hard research work of people like him.

I am also extremely grateful to Mr. U. Somasundaram for his invaluable support in helping me out on matters related to experimental work which is the basic soul and essence of my project. I am also highly in debt of all the members of the Applied Optics laboratory for the amazing work environment they maintain in the lab.

Last but certainly not the least I would like to thank Nikhil Tajane my fellow student throughout the project without whose contributions this project would have hardly reached the conclusion that it did.

Partha Pratim Basumallick

ABSTRACT

Analysis of fringe patterns, with the object of phase extraction from them is one of the primary duties of the experimentalist in the field of optics. To achieve that, techniques like step variation etc. were the ones generally employed until very recently. However, of late the Hilbert transform has been applied successfully to extract the phase information from the experimentally recorded interferograms. The beauty of this approach is that only a single interferogram is required to obtain the necessary phase information from the recorded interferogram as the Hilbert transform operation analytically generates another signal with a specific change of $\frac{\pi}{2}$ in the phase angle. In the present project I have tried to study how well this theoretical proposition bears out with the data obtained from experiments. The basic experimental setup is that of the Michelson interferometer for the experimental data used in the experiment. Apart from this the other data used are from MATLAB generated fringes obtained from a code written for that specific purpose. The details of the experimental setup, relevant algorithms and the codes used in this project have been presented in the following sections of the report.

List of Figures

TABLE OF CONTENTS

1 INTRODUCTION

The study of interference patterns is perhaps what can be described as the heart of experimental optics. Over the years of perseverance in this field of study, people have come up with many ingenious ideas of how best to extract relevant information from the experimentally obtained fringe patterns. For example, evaluating the phase information of the fringe patterns etc. The methodologies that were in vogue until very recently, regarding white light (phase shifting) interferometry and similar topics, were five-step, eight-step algorithms that involved the introduction of a known amount of angular deviation to the phase part in the expression of the intensity distribution, thus obtaining five or eight separate equations from which we would be able to extract the necessary information about the phase of the recorded interferogram. The reason for using long and tedious to implement algorithms of five steps or more is that we want to eradicate the error factor by as much as possible. The source of this error is that no matter how much we try, the manually introduced phase shift can never be ascertained to a very high level of precision.

But in spite of our best efforts in reducing the errors in those analysing techniques, we have a problem, which can have no solution if we keep on using the multiple-step algorithms. In fact the more we lengthen our algorithms the more we increase our problem which is – to record as many interferograms as the number of steps used in the algorithm. Clearly, this doesn't let the experimenter get out of the lab too much! This is the reason that of late we have tried to implement somewhat novel tactics, that of a less time consuming one involving the **Hilbert Transform (HT)**. The idea behind implementing this purely mathematical concept in the field of interferometric analysis is that this mathematical operation automatically introduces a phase shift of $\frac{\pi}{2}$ to the function on which it is applied. And as this is a purely mathematical operation there is no scope of any error being introduced as in the previous case. Thus we need to record only one single interferogram and introduce the phase shift by **HT** which is sufficient to extract the phase information.

The present project report demonstrates how we use this technique in evaluating the phase of the recorded interferogram and why this method is more efficient than the other ones. The necessary algorithms and other details concerning the project report have been presented in the chapters to

come. I have explained the basic structure of the **HT** operation and how we apply it with the help of the software MATLAB to the fringe pattern. Over here I would like to mention that, I have used code generated fringes, in the beginning, to illustrate the functioning of the present methodology under discussion, and then, later on, moved on to the experimentally obtained fringes for comparison between – the results and functioning of the **HT** technique in both the cases.

To summarize in brief it can be said that in cases where it is required to obtain quantitative phase information from a single interferogram for quantitative analysis, a $\frac{\pi}{2}$ phase-shifted interferogram is required. The **Hilbert Transform** can generate a new signal with a phase altered by $\frac{\pi}{2}$ while the amplitude is left unchanged with respect to the original signal. Hence we can apply the **Hilbert Transform** method for phase analysis of both static and vibration speckle fringes. The usefulness and effectiveness of the method are demonstrated with the examples of code generated and experimentally obtained fringes.

2 INTERFEROMETRY AND THE EXPERIMENTAL SETUP

2.1 Overview(s)

Interferometry is perhaps the most important investigative technique amongst the sub-disciplines of fiber optics, optical metrology, spectroscopy, mechanical stress/strain measurements and several other fields such as seismology, oceanography, biomolecular interactions etc. Hence it is more than prudent that a short description of this experimental technique is included over here so that we have a better idea about the physics related to the experimental aspect of the project.

> **Definition –** *Interferometry is nothing but the assimilation of all the optical measurement techniques which involve the superimposition of two or more coherent electromagnetic waves resulting in an interference pattern which has to be analyzed to determine the parameters of interest. In this context it is necessary to put forth the definition of coherence which is nothing but a simple criterion which is necessary for a successful formation of an interference pattern.*

In regard to the formation of interference pattern it must be mentioned that it can be achieved in two ways which are : –

i) division of wavefront
ii) division of amplitude

> **Definition – i)** *A wavefront splitting interferometer divides a light wavefront emerging from a point or a narrow slit (i.e., spatially coherent light) and, after allowing the two parts of the wavefront to travel through different paths , allows them to recombine. The following Fig-2.1 illustrates* ***Young's interference experiment*** *which is a classic examples of interference by division of wavefront. This phenomena occurs when two parts of a wavefront whose wavefront phase is constant along a given wavefront produced by a monochromatic point source are selected and then redirected to a common volume in space.*

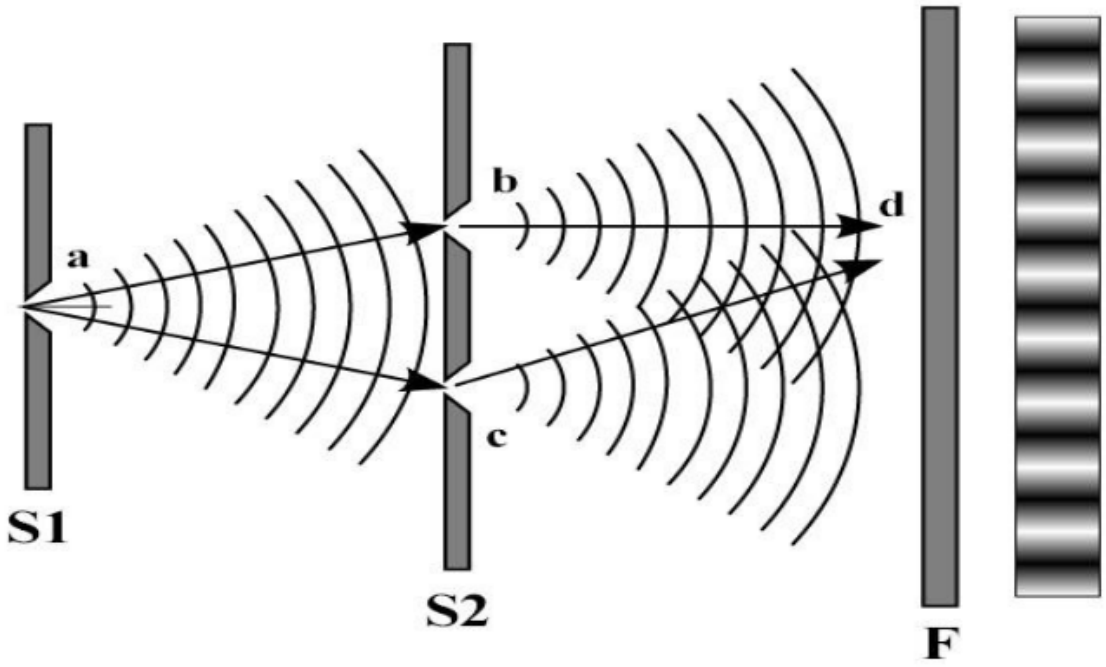


Figure 2.1: Interference pattern in Young's double slit experiment

Definition – ii) *As far as the definition of interference by division of amplitude goes it is stated as follows – when a beam of light is made incident on a partially reflecting surface or beam splitter, it suffers partial refraction and transmission at the surface, preserving the constant phase difference between the waves. These beams are superimposed to produce interference pattern. The best example of this kind of interferometry is provided by the Michelson interferometer. The following sub – section discusses the Michelson interferometer in detail as that is the experimental setup used for the project experiment.*

2.2 The Michelson Interferometer

The **Michelson–Morley experiment** is the one singular experiment in the echelons of experimental optics which claims the zenith slot without any trouble whatsoever. The primary reason lies in its historical significance which was to lay the foundation stone of special relativity, but the real brilliance of the experiment was the designing faculty of **A. A. Michelson** which provided a striking degree of precision to the experiment. A schematic diagram(Fig-2.2) illustrating the setup of the Michelson interferometer has been provided on the next page, sometimes a compensator is also placed in the setup although it is not an essential component of the experimental setup but is introduced mainly to reduce the intensity of the beam before it reaches the detector (which is more than often) as the camera sensor can be damaged by the extremely high intensity of the laser beam used in our experiment.

The Michelson interferometer consists of two mirrors one movable and the other fixed, a beam splitter (inclined at 45°), a light source from which the light beam goes to the beam splitter which bifurcates the incident beam into a partially reflected and a partially transmitted beam which go onto hit the two mirrors which are positioned mutually orthogonal to each other where the both the rays suffer reflection only to reunite together to form an interference pattern.

Having discussed the basic setup of the Michelson interferometer I will proceed with a concise account of the expression of intensity distribution for interference due to monochromatic light. As the project revolves around the goal of removing the components of the intensity distribution equation by the said technique(**HT**) to finally extract the phase information.

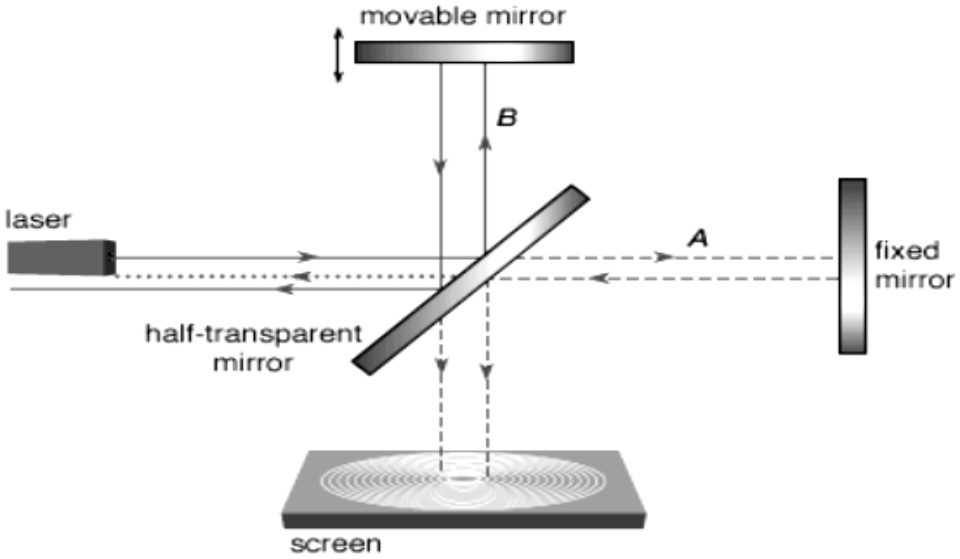


Figure 2.2: A Schematic Representation of The Michelson Interferometer

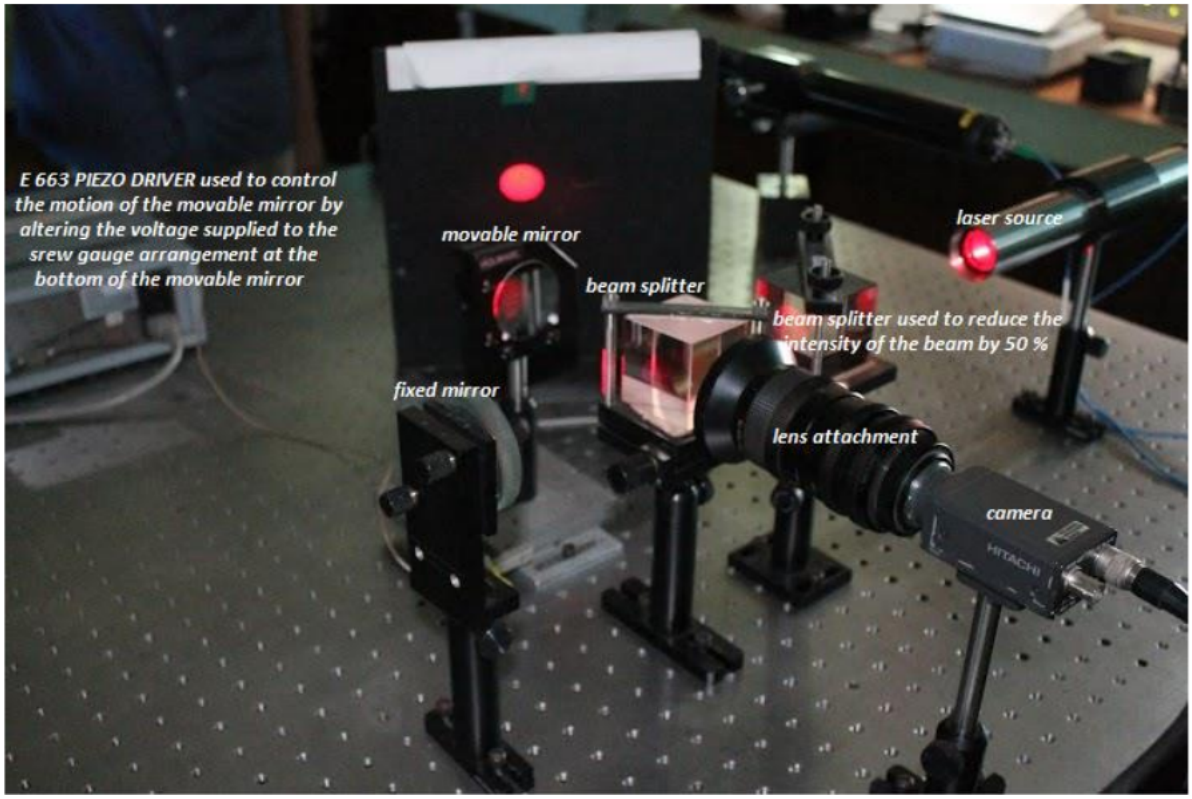


Figure 2.3: Snapshot of a Michelson Interferometric setup in our Applied Optics Laboratory

2.3 Intensity distribution for monochromatic light

The intensity distribution for interference pattern can be written as

$$I(x,y) = I_0(x,y)(1 + V(x,y)\cos(\phi(x,y))) \tag{2.1}$$

where ϕ is the phase difference between the beams from the two mirrors when they reunite at the beam splitter. It is determined by the time delay resulting from the optical path difference between

the beam splitter surface and the two mirrors. Hence if the said path difference is, say z, then the phase difference can be expressed as,

$$\phi = \frac{2\pi}{\lambda}(2z) \tag{2.2}$$

where $I_0(x,y)$ is the bias intensity, $V(x,y)$ is the visibility, $\phi(x,y)$ is the phase to be determined. The spatial co-ordinates can be ignored for simplicity.

V is the visibility of the fringe pattern i.e., the parameter which determines the contrast of the fringe pattern and I_0 is the background intensity. To clarify things more clearly let us just write the basic form of the intensity expression

$$I = I_1 + I_2 + 2\sqrt{I_1 I_2}\cos\phi \tag{2.3}$$

where I_1 & I_2 are the intensity of the beams from both the mirrors which helps us express the visibility factor as

$$V = \frac{2\sqrt{I_1 I_2}}{I_1 + I_2} \tag{2.4}$$

Now the contrast of the fringe pattern can be poor because of the noise associated with the background/DC term and the term $I_0 V$. How we go about removing these factors from the interference pattern by the **HT** method and why it is a superior approach as compared to the previous techniques has been discussed in detail in Chapter-3.

The following section is devoted to the experimental design and the specifications of the instruments used in the experiment.

2.4 Experimental Setup–Specifications

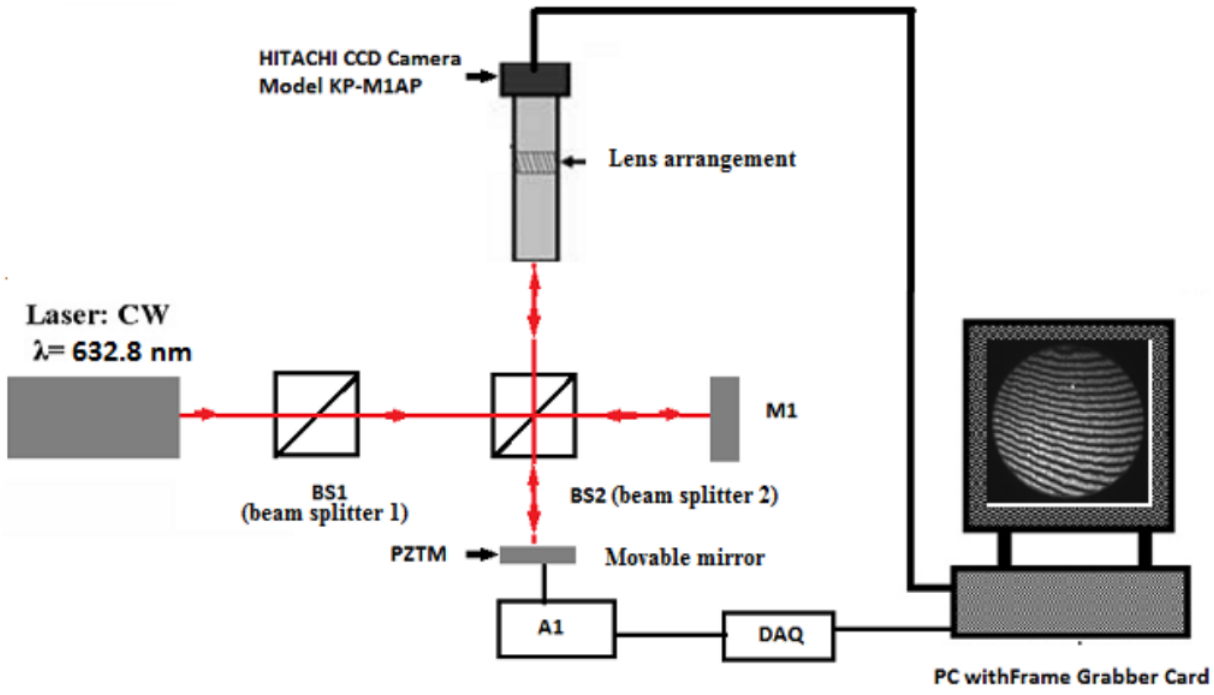


Figure 2.4: The schematic diagram of our experimental setup

The schematic of the Michelson Interferometer system as used in our comprehensive experimental setup is shown in Fig-2.4. The narrow beam from a 632.8 nm CW He-Ne laser is passed through a beam splitter (BS1) to reduce its intensity. It is divided into two beams using a beam splitter (BS2). One is reflected by a static mirror (M1) and the other is reflected by PZTM . The imaging system consists of Macro Video Zoom Lens (EHDZ68M) which can directly be connected to Hitachi CCD camera (KP-M1AP). The CCD is interfaced to a PC with Matrix Vision (pcIMAGE-SDIG) frame grabber card. The PZTM(Piezoelectric Transducer mirror) in the setup is used to visualize the time averaging fringe patterns. It is controlled by PC using DAQ (DT330) and an LVPZT- Amplifier (E-663). Thus we conclude our second chapter about the experimental setup and interferometry.

3 THE HILBERT TRANSFORM IN OPTICS'LAND'

The Hilbert Transform(**HT**) is a purely mathematical concept that is widely used nowadays in the field of signal processing and image processing owing to some of its special properties like returning the function with the same domain after operating on it. For instance, if we had a function $u(t)$ (say) and we applied **HT** on it then we would get the function $H(u)(t)$ that would be in the same domain as $u(t)$. Its importance in the field of image processing comes from the fact that it can yield an analytic representation from a signal. In the coming section I have discussed a mathematically rigorous definition of **HT** followed by a subsection, where I have mainly highlighted on those properties of the **HT** operation which are of essence to the project and justify its use in the field of image processing.

Definition – *The Hilbert Transform of a signal $x(t)$ is defined by the equation-3.1 where the integral is the Cauchy principal value integral.*

$$H\left(x(t)\right) = \frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{x(\tau)}{t-\tau}\, \mathrm{d}\tau \tag{3.1}$$

The reconstruction formula is given by equation-3.2

$$x\left(t\right) = -\frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{H\left(x(\tau)\right)}{t-\tau}\, \mathrm{d}\tau \tag{3.2}$$

*and is defined as the Inverse Hilbert Transform. The pair $x(t)$, $H\left(x(t)\right)$ is called a Hilbert Transform Pair. The Hilbert Transform is important in the field of signal processing where it is used to derive the analytic representation of a signal $x(t)$. The properties of **HT** which make it such an indispensable tool in the field of image processing have been discussed below.*

3.1 Properties of The Hilbert Transform

A signal $x(t)$ and its **HT** $H(x(t))$ have

i) The same amplitude spectrum.
ii) The same autocorrelation function.
iii) They are orthogonal.
iv) The Hilbert transform of the function returns the same function with a phase change of $-\frac{\pi}{2}$.

As a result of the above properties the **HT** on operating twice on a function returns the function just negated in value i.e.,

$$H(H(x(t))) = -x(t) \tag{3.3}$$

provided the integrals defining both iterations converge in a suitable sense.

In order to illustrate the basic nature of **HT** operation I have provided MATHEMATICA plots of some functions and their **HT**'s in the same graph to illustrate the properties mentioned earlier.

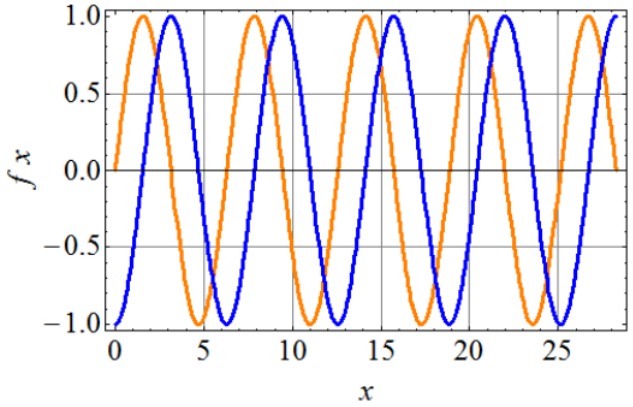


Figure 3.1: $\sin(x)$ & $[H(\sin(x)) = -\cos(x)]$ vs x

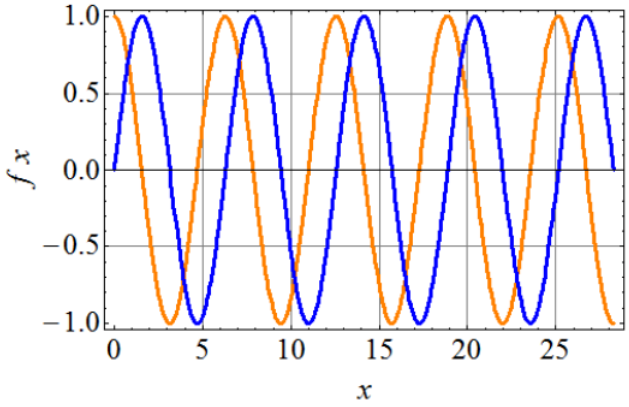


Figure 3.2: $\cos(x)$ & $[H(\cos(x)) = \sin(x)]$ vs x

In the above plots, the orange line denotes the original function whereas the blue line denotes the **HT** of the function. It can be clearly seen that the Hilbert transform yields the negated value of the original function on operating twice on the same function. The blue line in the second plot is the **HT** of $\cos(x)$ but $-\cos(x)$ was the **HT** of $\sin(x)$ to begin with, hence the graphical proof of equation-3.3. But just to drive home the point the function $\sin(x)$, its **HT** and double **HT** is provided below in the following page where the original function is in orange, the output after the first Hilbert operation is in blue, and finally the output on operating the function with Hilbert twice in purple. The MATHEMATICA file used to evaluate Hilbert Transform and plot the functions has been given in the Appendix. I have also provided similar plots of some other relevant functions occurring frequently in the field of optics to illustrate the effect of Hilbert on them and their subsequent dependence and relation with the variable.

Fig-3.4 shows the variation of Sinc(x) i.e., $\frac{\sin(x)}{x}$ and also its **HT** and double **HT** , this particular function is mainly important because Sinc(x) appears in the expression of intensity distribution for diffraction patterns and if we were to analyse them with **HT** method then we can be pretty confident that our approach is consistent with the existing theoretical framework.

On close scrutiny it is pretty evident that unlike its predecessors Fig-3.6 does not have the purple line the obvious reason is that the Hilbert Transform of the mother function which in this case is $\log(x)$ does not yield an output conducive to the requirements of **HT** being applied on it again. Thus we must be careful about the cases where we want to implement **HT** as well because for data whose

functional identity is not in agreement with the required conditions the mathematical operation will not yield any credible results.

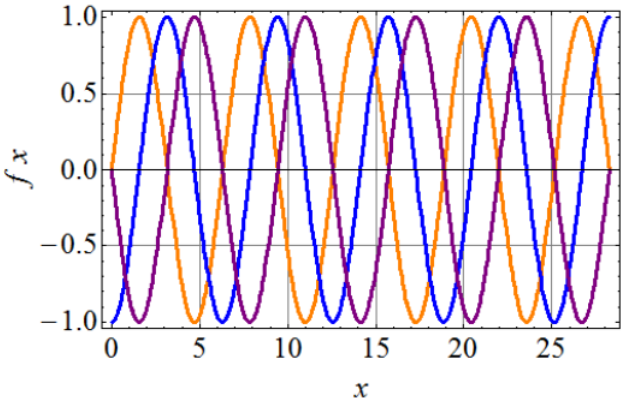


Figure 3.3: $\sin(x)$, $H(\sin(x))$ & $H(H(\sin(x)))$

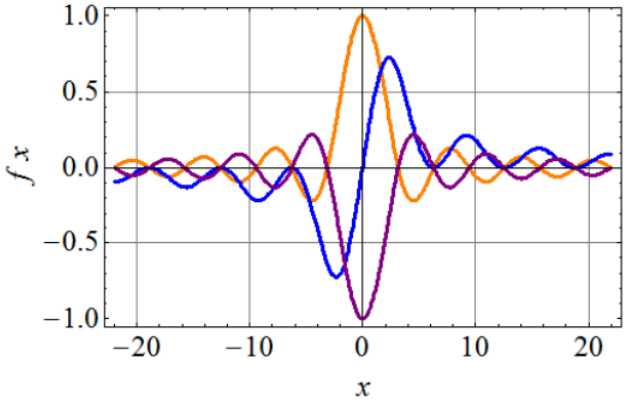


Figure 3.4: $Sinc(x)$, $H(Sinc(x))$ & $H(H(Sinc(x)))$

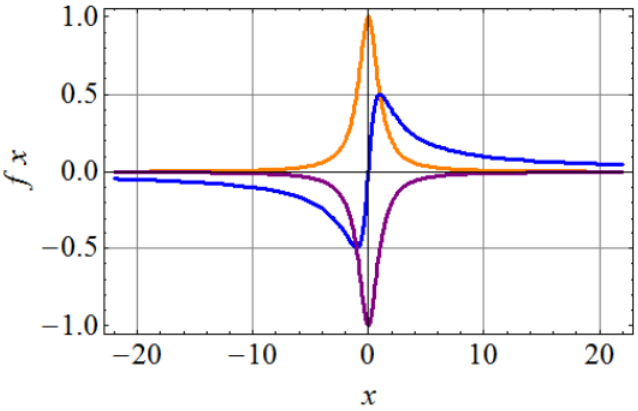


Figure 3.5: $f = \frac{1}{1+x^2}$, $H(f)$, $H(H(f))$

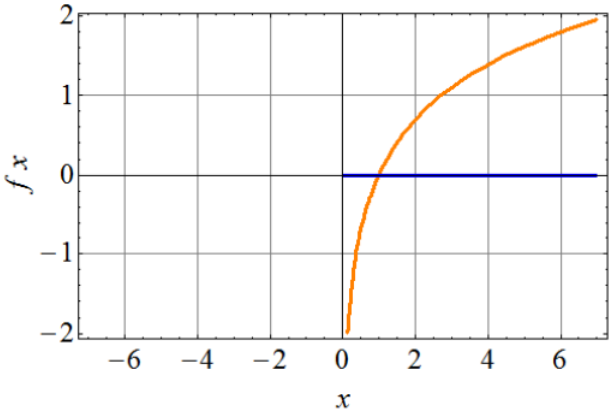


Figure 3.6: $\log(x)$ and it's HT

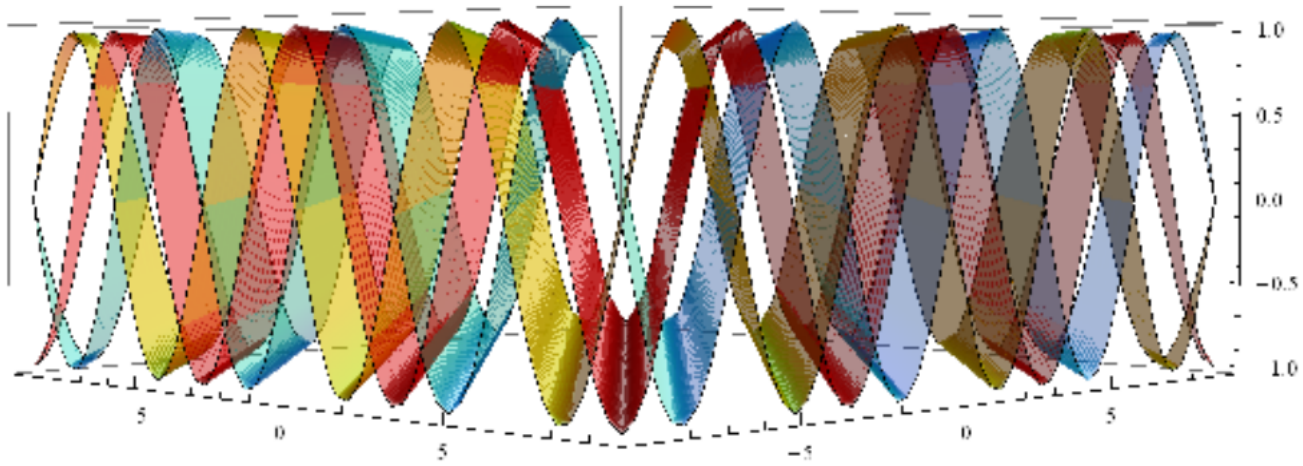


Figure 3.7: 3D plot of $\sin(x)$ and it's 1^{st} HT followed by it's 2^{nd} HT

The 3D plot performed in MATHEMATICA of the $\sin(x)$ function and it's **HT** and double **HT** is given below to illustrate exactly how they overlap with each other. The yellow strip is the original function, the red strip is the output after 1^{st} **HT** operation and finally the cyan-blue strip which gives the output after the 2^{nd} **HT** operation on the function.

Now, in order to demonstrate exactly how the above properties of **HT** make it a necessary and indeed an indispensable tool in the field of image processing I have attached the processed images of an experimentally obtained interferogram. It should be mentioned in this context that the experimental fringe was obtained by speckle interferometry and hence has lots of noise. However, that does not affect our objective in the least.

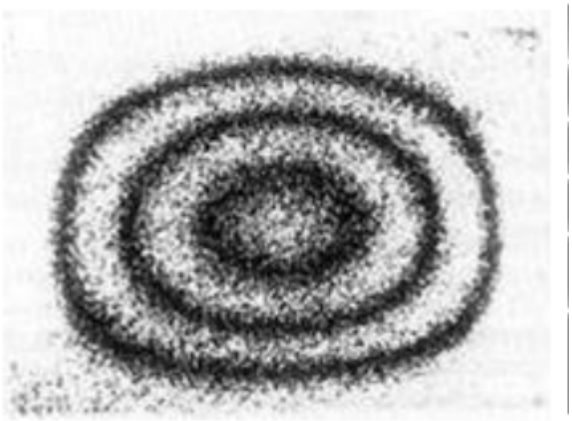


Figure 3.8: Original speckled image

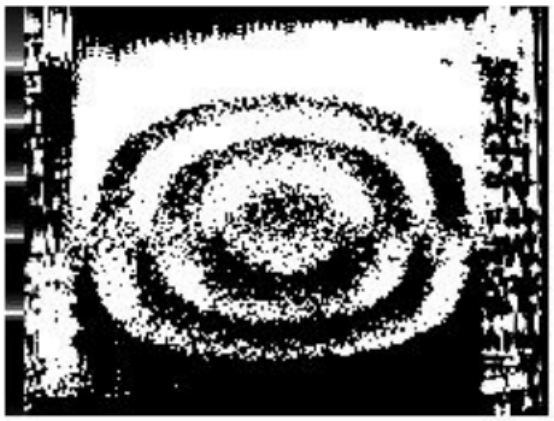


Figure 3.9: Image after applying HT once

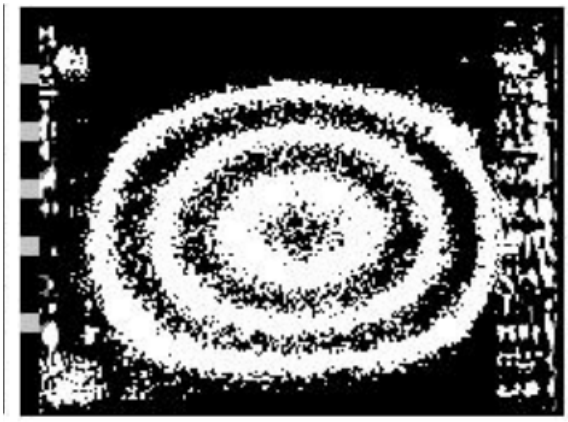


Figure 3.10: Image after applying HT twice

Figure 3.11: After negating the Fig-3.10 image

To describe the sequence of images given above – we concentrate on the second image(3.9) initially, which was obtained after applying **HT** on the first image, the second image clearly shows a sign ambiguity across its line of fringe orientation which arises since the Hilbert operation in MATLAB doesn't recognize the signature of the phase of the image, hence resulting in the sign ambiguity. Moving onto the third image(3.10) which is extracted by applying **HT** twice on the image we can see that it is the replica of the original image barring a phase shift of π, which shows that the original image's negated version has been produced by the **HT** operator by its application on the mother image twice. In order to present the above claim with an iron-clad justification, a negated image of the third image has been presented which is the replica of the initial image with the added feature of highly enhanced contrast. Now of course the question arising, in this case, is that if our mathematical definition predicted the reproduction of the original function after negating the output yielded on applying **HT** twice to the function, why then in the case of image processing with MATLAB we are coming up with this contrast enhancement? The answer lies in the fact that the Hilbert operation in MATLAB yields a complex signal whose real part is the original function and the imaginary part is the **HT** of the original function. Hence obviously we extract the imaginary part of the signal yielded by MATLAB as a result of which every time we invoke the **HT** operation a certain amount of data is compromised which results in the enhancement of the image being processed.

Although our previous discussions were quite extensive there are some other aspects of Hilbert Transform which must be discussed concerning its proposed applicability for the present project. For instance, how do we know that the imaginary part of the analytic function generated by MATLAB is unique and we can use it for our purpose? The following sections address these points.

3.2 Hilbert Transform as a Boundary – Value Problem

To establish the uniqueness of the companion function, we note that any analytic function $f_{ext}(z) = f_R(z) + if_I(z)$ defined on the complex plane $z = x + iy$ must satisfy Cauchy-Riemann equations,

$$\frac{\partial f_R}{\partial x} = \frac{\partial f_I}{\partial y} \tag{3.4}$$

$$\frac{\partial f_R}{\partial y} = -\frac{\partial f_I}{\partial x} \tag{3.5}$$

Consequently, both f_R and f_I satisfy Laplace's equation, as follows

$$\frac{\partial^2 f_R}{\partial x^2} + \frac{\partial^2 f_R}{\partial y^2} = 0 \tag{3.6}$$

$$\frac{\partial^2 f_I}{\partial x^2} + \frac{\partial^2 f_I}{\partial y^2} = 0 \tag{3.7}$$

over the region where $f_{ext}(z)$ is analytic. Conventionally, by requiring $f_{ext}(z)$ to be analytic in the upper half-plane, the quest of finding the **HT** for any given function $f(x)$ can be formulated as a boundary value problem (Scott, 1970). By specifying the following boundary conditions :–

i) $f_R(x, 0) = f(x)$ and
ii) $f_R(x, y) = 0$ as $x \to \pm\infty$ or $y \to \infty$

$f_R(x, y)$ can be uniquely determined by solving Laplace's equation-3.6 in the upper half plane. Then, through Cauchy-Riemann equations, $f_I(x, y)$ can be calculated in the entire upper half plane and in particular on its boundary the x-axis (Scott, 1970). Thus $g(x) = F_I(x, y)$ is the Hilbert transform of the given function $f(x)$.

3.3 Applications of HT for Signal Processing

Signal processing is nowadays conveniently conducted in the digital domain. The first step of signal digitization involves sampling an analog signal $x(t)$ at a constant rate $f_s = \frac{1}{T}$ where T is the sampling period. In this section, we denote the sampled waveform as $x[n] = x(nT)$, using the square brackets [ ] to indicate that the signal is sampled in discrete time. The discrete-time Fourier Transform (**DTFT**) is defined as follows :

$$X(j\omega) = \sum_{n=-\infty}^{\infty} x[n] \exp^{-j\omega n} \tag{3.8}$$

Note that $X(j\omega)$ is periodic at every 2π interval in the frequency domain. In the following section, I have discussed how Hilbert Transform can be defined in discrete time.

3.4 The Discrete – Time Hilbert Transform and Hilbert Transform

Recall that the Hilbert transform introduces a 90° phase shift to all sinusoidal components in a signal. In the discrete time periodic-frequency domain, the transfer function of Hilbert Transform is specified as follows,

$$H(j\omega) = \begin{cases} -j, & 0 < \omega < \pi \\ +j, & -\pi < \omega < 0 \end{cases} \tag{3.9}$$

The convolution kernel for $H(j\omega)$ can be calculated through inverse Fourier transform (Oppenheim & Schafer, 2010):

$$h[n] = \frac{1}{2\pi} \int_{-\pi}^{+\pi} H(j\omega) \exp^{-j\omega n} \, \mathrm{d}\omega \tag{3.10}$$

$$= \begin{cases} \frac{2}{\pi} \frac{\sin^2(\pi n)}{n} & n \neq 0 \\ 0 & n = 0 \end{cases} \tag{3.11}$$

Note that $h[n]$ has infinite support from $n = -\infty$ to $+\infty$. In practice, the entire function cannot be stored digitally. To circumnavigate this difficulty, we now discuss two major methods for calculating the discrete-time Hilbert Transform. The relevance behind the discussion about discrete-time Hilbert Transform has been elaborately discussed in the next section ***especially*** concerning the implementation with programming in MATLAB.

3.5 The implementation of HT in MATLAB

The universally popular scientific-computing software MATLAB (MathWorks, Natick, Massachusetts, USA) has a hilbert() function that "computes the so-called discrete-time analytic signal $x = x_r + ix_i$ such that x_i is the Hilbert Transform of x_r". MATLAB's implementation of the hilbert() function takes advantage of the Fast Fourier Transform (FFT). Essentially, the hilbert () function completes the calculation in three steps:–

i) perform the FFT of x_r,
ii) set the elements in FFT which correspond to frequency range $-\pi < \omega < 0$ to zero,
iii) perform the inverse FFT.

That the three steps above can work is a consequence of the fact that, if $x_i[n]$ is the discrete **HT** of $x_r[n]$, the Fourier Transform of $x_r[n] + ix_i[n]$ vanishes for all negative frequencies $-\pi < \omega < 0$. This can be verified by inspecting the definition given in equation-3.9.

Remarks: It must be noted in this context that because of MATLAB's popularity and easiness to use, there is a genuine scope of mis-interpreting the returned $x = hilbert(x_r)$ as the actual mathematical **HT** of x_r. One must be aware of these deviations from the conventional definition to avoid unnecessary confusion.

The following figures represent the nature of a Discrete Hilbert Transform (**DHT**) depending on the no. of points; I have plotted the combined plot of triangular and square wave and their subsequent **DHT**. The original data sets are in blue whereas their **DHT**'s are in purple.

It can be easily noticed from the following plots how **DHT** operation works and hence exactly how MATLAB carries out its hilbert operation. As evident from the figure with $n = 500$ it can be easily appreciated that as we keep including more and more points the discreteness disappears and the functions are considerably smoothened out and not to mention become continuous. The reason behind emphasizing this point lies in the fact that in the course of this report we will see that the **HT** method of phase extraction in MATLAB yields perfect results for the case of simulated fringes. ***However, for actual speckled images recorded there is a reverse somersault, as far as the image quality of the resulting processed images are concerned. The reason being the presence of the noise elements present in the original source images. Although I do work on noise removal in this project to some extent the detailed noise treatment of speckled fringes is beyond the scope of the present***

project. These noise elements are treated as discontinuities in the ideal profile of the obtained data set and hence the operation of HT also yields a similarly noisy profile instead of the expected smooth profile.

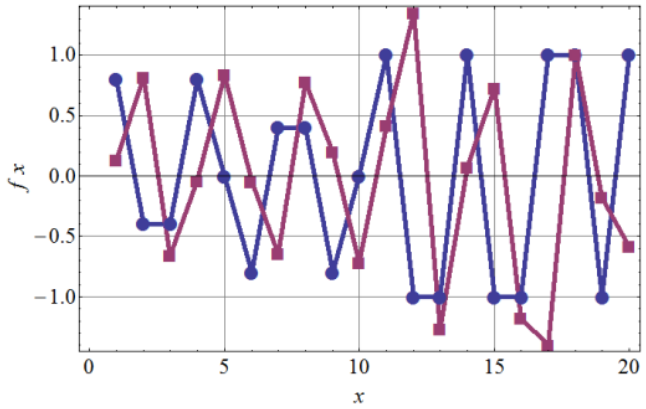


Figure 3.12: Data vs DHT plot for $n = 10$

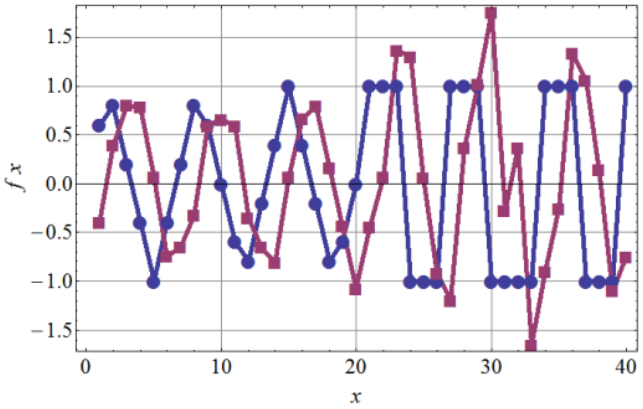


Figure 3.13: Data vs DHT plot for $n = 20$

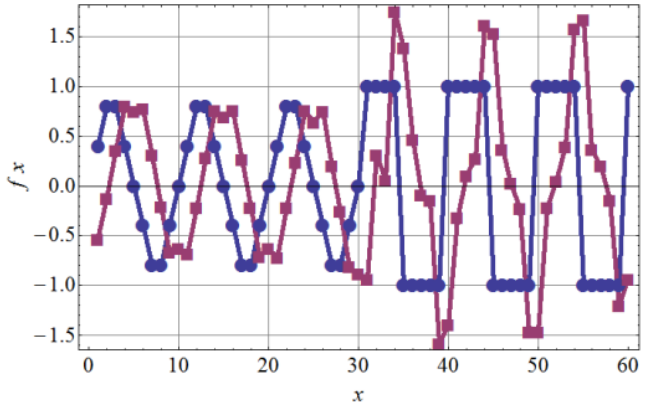


Figure 3.14: Data vs DHT plot for $n = 30$

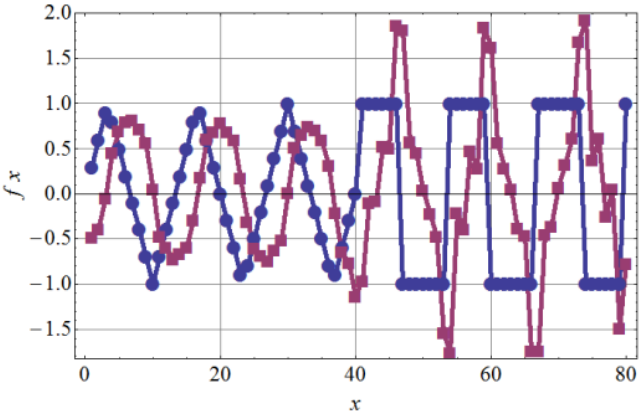


Figure 3.15: Data vs DHT plot for $n = 40$

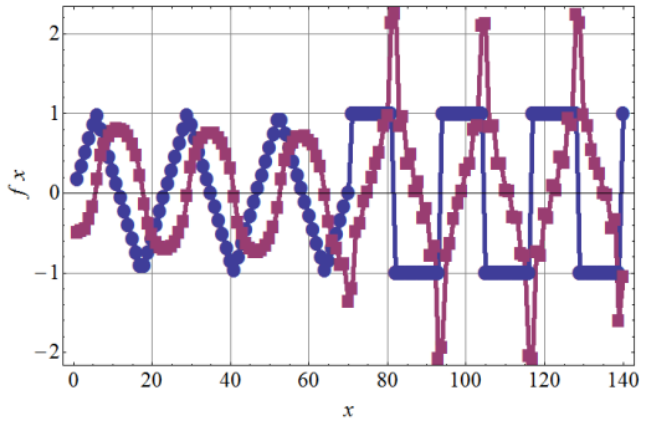


Figure 3.16: Data vs DHT plot for $n = 60$

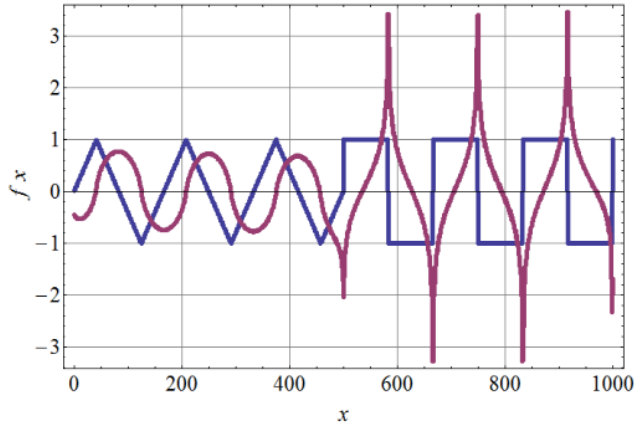


Figure 3.17: Data vs DHT plot for $n = 500$

Every relevant property, application and *modus operandi* of MATLAB in implementing the **HT** has been discussed in this chapter and now we will try to harvest the effectiveness of this technique in the next chapter where the algorithm used for phase extraction from interferograms has been presented along with the fringe patterns analyzed on basis of that.

4 THE HILBERT TRANSFORM(*GOOD*), THE FRINGE PATTERN(*BAD*) AND THE ANALYSIS(*UGLY ?*)

The Hollywoodesque title of the chapter kind of gives away the whole story doesn't it? Throughout the course of this chapter, it will be established how the title is aptly suited as the underlying theme. The reason behind coining this particular title is that for code generated fringe patterns our algorithm works perfectly ***'good'*** but as soon as we go over to analyzing actual fringe patterns we find that the quality of the 3D – phase map deteriorates, primarily owing to the presence of noise elements in the experimentally recorded image(s) which filters down to the phase map, in accordance to the *modus operandi* of MATLAB's hilbert operation as explained earlier in the previous chapter. Thus the ***'bad'*** quality of experimental images have a major say about what quality of phase map we are going to get. Finally how the analysis of the experimental fringes helps to eradicate the seemingly ***'ugly'*** disagreement between the theoretical proposition and practical implementation is described in detail.

Throughout the course of this chapter I discuss the flowchart describing the code used to calculate the phase map of the fringe-pattern along with detailed explanation of the individual steps of the flowchart. The detailed description of the algorithm is facilitated by the adjoining images obtained during the intermediate stages of pre-processing and analysis.

4.1 The Implementation of HT in the Experimental Domain

Before presenting the flowchart some details concerning the pre-processing of the image as well as the logic behind the applying of **HT** for phase extraction are described in this section. Now as

mentioned earlier in the Chapter-2 the expression for intensity distribution is given by

$$I(x,y) = I_0(x,y)\,(1 + V(x,y)\cos(\phi(x,y))) \tag{4.1}$$

In order to successfully extract the phase information our first object is to remove the bias intensity I_0 irrespective of the fact whether it is uniform or non-uniform. This is a basically straightforward task where we have to subtract the mean value of the signal from the original signal. The image obtained after this operation will be denoted as E-frame.

$$E = I_1 - < I_1 >= I_0 V \cos\phi \tag{4.2}$$

However, for the cases where the bias intensity varies slowly in the spatial domain, we apply *local average filtering* to remove the bias. The effectiveness and simplicity of this digital filter are beyond any question whose *modus operandi* is to simply average a given number of points in the spatial dimensions.

Now when we first apply hilbert in MATLAB on the images it effectively works as a filtering agent owing to the reasons mentioned in previous chapters. Now when Hilbert carries out this default filtering the amplitudes of the spectral components are left unaltered, except for the obvious side effect of change in phase by 90°, now whether this phase change is positive or negative is entirely dependent on the sign of x where x is the argument of ϕ(the phase). Therefore the **HT** of the bias removed signal E is

$$G = HT\{E\} = -I_0 V \sin\phi \tag{4.3}$$

Combining equations 4.2 & 4.3 the factor $I_0 V$ can be written as

$$\sqrt{E^2 + G^2} = I_0 V \tag{4.4}$$

Now if we combine equations 4.4 & 4.2 we can obtain a fringe pattern with enhanced contrast features described as

$$F = \cos\phi \tag{4.5}$$

Now, we proceed on to the task of extracting the phase which contains all the information about surface height variations encoded in it. The phase of the interferogram is ascertained by performing **HT** on the frame F and then taking the arctan of its ratio to F. The expression is simply given as

$$\phi^{'} = \arctan\left(\frac{HT\{F\}}{F}\right) \tag{4.6}$$

The phase $\phi^{'}$ thus obtained is affected by sign jumps (π shifts) in parts of the phase map due to the lack of discrimination between positive and negative spatial frequencies. There have been many attempts to resolve this problem previously, for instance using a ***vortex operator*** to filter the data in the frequency domain. But the approach used here is that of filtering the calculated phase with π jumps. The π shift corrected phase $\phi^{''}$ can be ascertained using the relation

$$\phi^{''} = \phi^{'} sgn\,(diff\,(\cos(\theta))) \tag{4.7}$$

where, sgn is the signum function which is defined as

$$sgn(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} \tag{4.8}$$

and θ is the fringe orientation which can be evaluated by using the expression

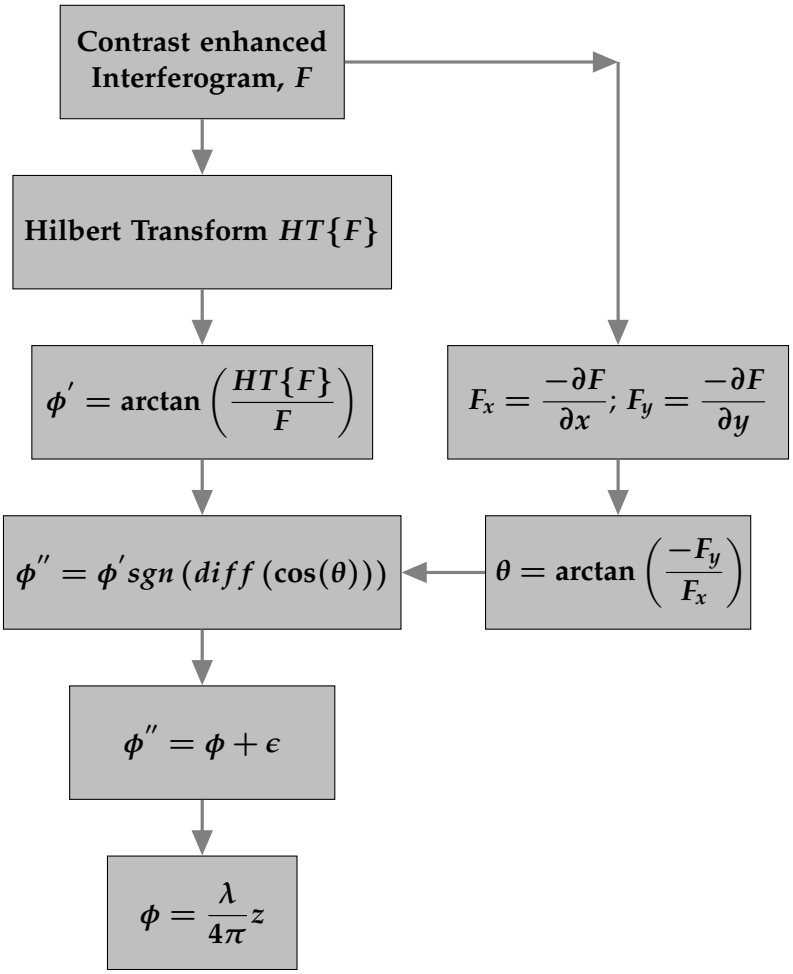
$$\theta = \arctan\left(\frac{-F_y}{F_x}\right) \tag{4.9}$$

where $F_x = \frac{-\partial F}{\partial x}$ and $F_y = \frac{-\partial F}{\partial y}$ are the local intensity gradient components along $x-$ and $y-$ axes.

The magnitude of ϕ'' calculated via this method is not in agreement with the original argument ϕ of the cosine term in the original expression. The reason behind this is that the **HT** operation in MATLAB uses finite limits instead of the $-\infty \rightarrow +\infty$ limit mentioned in the theoretical definition of **HT**. Thus there is an error introduced which is denoted by ϵ over here in the equation-4.10.

$$\phi'' = \phi' + \epsilon \tag{4.10}$$

4.2 The Flowchart



The steps of the flowchart have been already explained in the previous section hence we will move onto the next section where the above flowchart has been implemented to extract the phase map of a code generated circular fringe pattern.

4.3 Analysis of Code Generated Fringe Patterns

Figures 4.5, 4.6 are of prime interest to us now as the other figures prior to that are quite self-explanatory and also have been illuminated upon earlier. Fig-4.6 which shows the wrapped phase map easily demonstrates the phase jumps of π discussed previously. From the figure, it can be easily realized that the upper half of the image is what was expected but there is a shift of π in the lower semicircle region. Now, what is the handiest way to get rid of this 'glitch'? Undoubtedly the most straightforward solution to this question is to perform some operation on this phase map so that a further phase shift of π is suffered by the lower semicircle. In order to achieve this feat, we need a si-

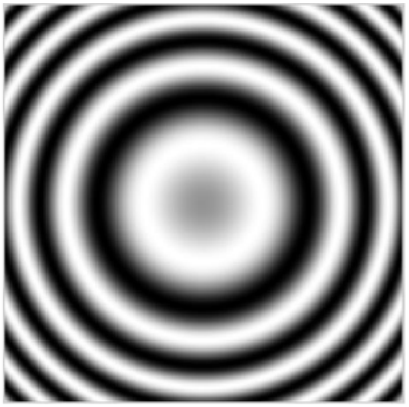


Figure 4.1: Code generated circular fringe pattern

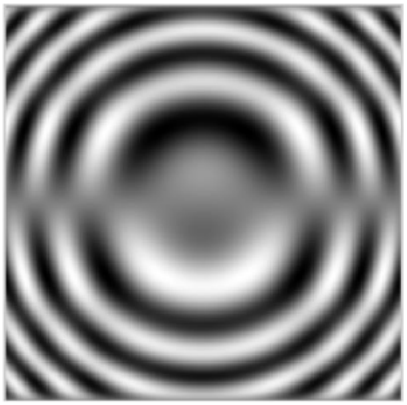


Figure 4.2: After applying HT once

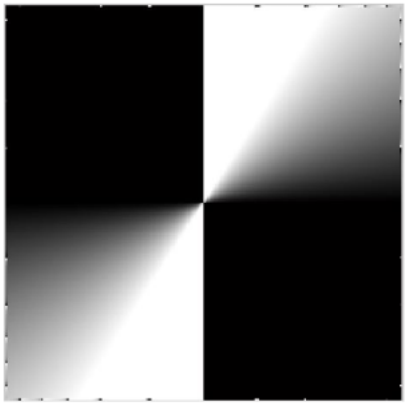


Figure 4.3: Fringe orientation map

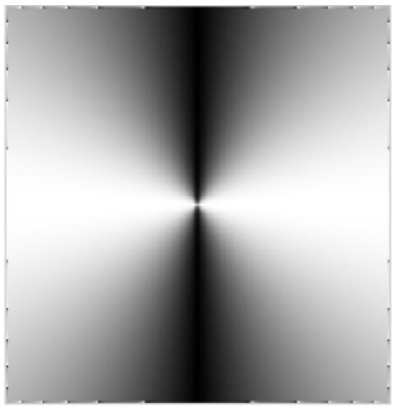


Figure 4.4: Cosine map

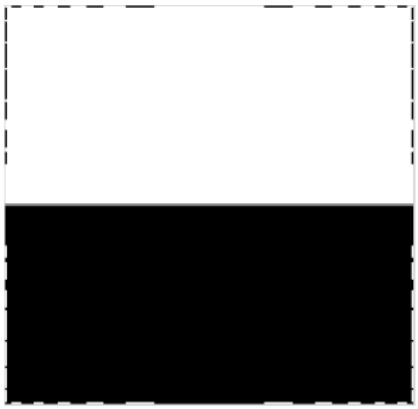


Figure 4.5: Signum map

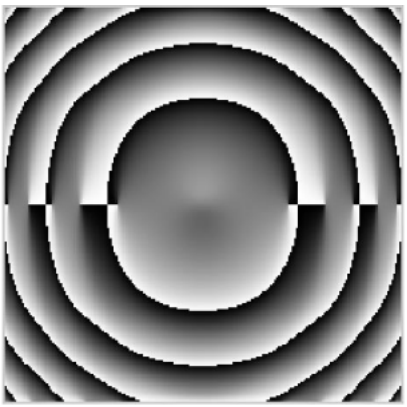


Figure 4.6: Wrapped phase recovered by HT method

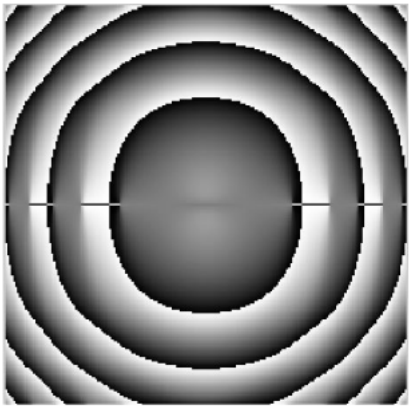


Figure 4.7: After removing π jumps

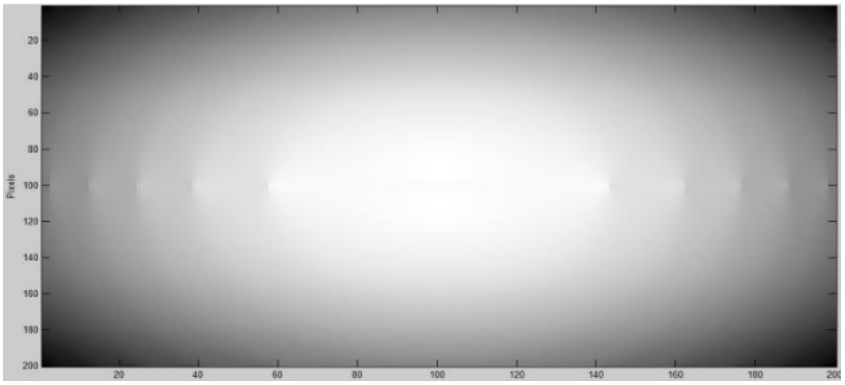


Figure 4.8: Intensity coded 2D phase map

mple multiplication of the data matrix of the phase map with a data matrix whose lower half is entirely constructed of data points (-1) i.e., 'black' and the upper semicircle is entirely made up of data points (+1) i.e., 'white'. This tailor-made matrix is called the signum map as the signum function is utilized to construct the matrix. It is necessary to mention in this context that in the wrapped phase map the black portion represents $-\pi$ and the white portion represents $+\pi$. The cosine of the fringe orientation θ is calculated and is shown in Fig-4.4 where black represents 0 and white 1. The signum of the difference map shown in Fig-4.5 is thus used as a mask where black represents (-1) and white (+1) as mentioned earlier. Hence this phase map mask is used to remove the sign ambiguity in the calculated phase shown in Fig-4.7 (phase map without the π jumps. Fig-4.8 is the 2D intensity coded phase map of the circular fringe pattern. The 3D phase map of the fringe pattern

has been provided below obtained after correcting the π jumps and also unwrapping the wrapped phase map. Unwrapping is one of the essential tools of image processing which will be discussed in the next chapter in greater detail.

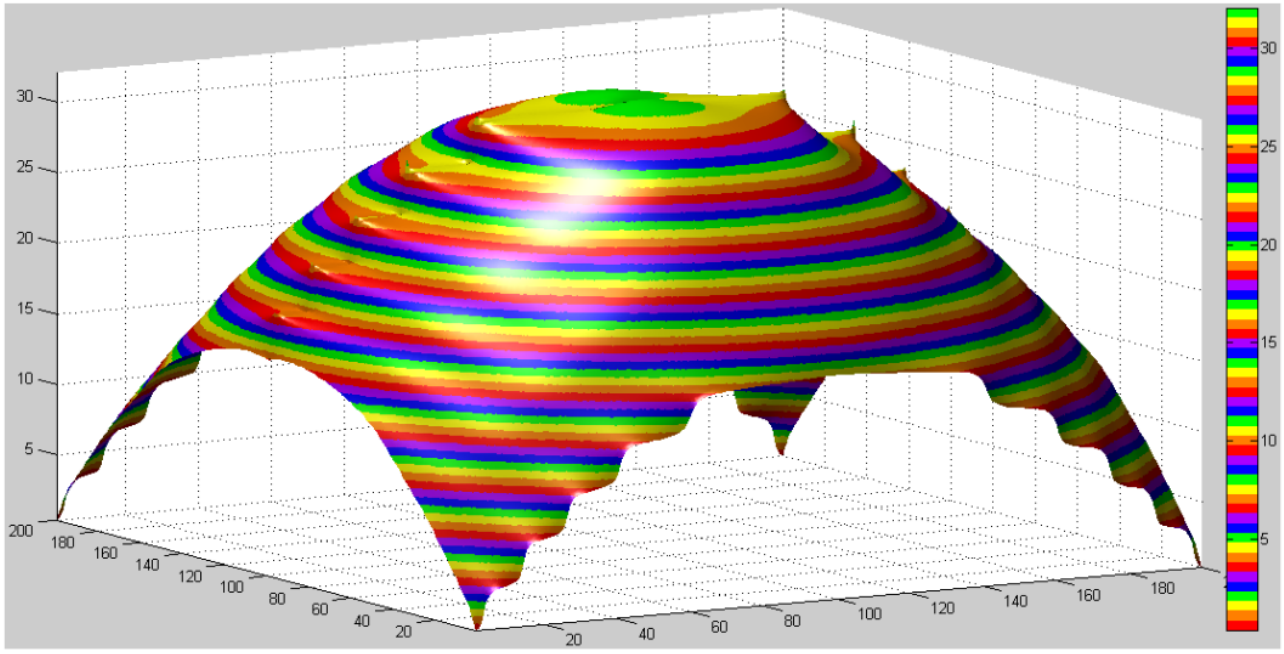


Figure 4.9: Recovered 3D phase map of the circular fringe pattern

Although the fringe pattern is extremely smooth there are some irregular peaks along the ridges of the 3D map the reason behind this has been discussed in the following section.

4.4 Generation of Noise Even in Simulated Fringes!!!

The peaks along the ridges of the 3D phase map are due to some missing data resulting as a side-effect after the application of hilbert in MATLAB. The reason for this is that when we choose the (imaginary) **HT** part of the analytical signal generated by MATLAB some of the original data is invariably lost. Also, the MATLAB software doesn't evaluate hilbert for the entire $-\infty \rightarrow +\infty$ range, it takes some finite value as the extreme limits and does the calculation. This is true not only for the hilbert operation but in the case of other operations as well, for instance when we generate the signum map by differentiating the cosine map (which is the primary source of error) itself. This can be easily shown by drawing the 3D map of the cosine map given in Fig-4.10

It can be easily noticed from the plot in Fig-4.10 that where the cosine map in Fig-4.4 was smooth to all appearances the 3D plot presents sharp peaks near the edges. However, these peaks can be easily smoothened out to some extent by filtering the cosine map itself. The 3D surface plot in Fig-4.11 is that of the filtered cosine map. It can be clearly seen that the top peaks of the original cosine map have vanished after filtering but the bottom striations still remain though reduced to some extent. Now in order to have a perfectly well-recovered phase map, it is necessary to have a perfect signum map and hence by extension demanding perfection in its source – the cosine map(itself) and the fringe orientation map. Instances of noisy signum maps and hence the source of noise originating in the ultimate phase map have been discussed in the upcoming chapter.

I have discussed the basic algorithm and its *modus operandi* with the help of a simulated circular fringe in great detail in the previous sections. The following section is devoted to implementing the **HT** based algorithm on the actual experimental fringes and the results obtained therefrom. The analysis of these fringes and techniques which we implement to improve the original algorithm are discussed in the sub-sections 4.5.1 & 4.5.2 respectively.

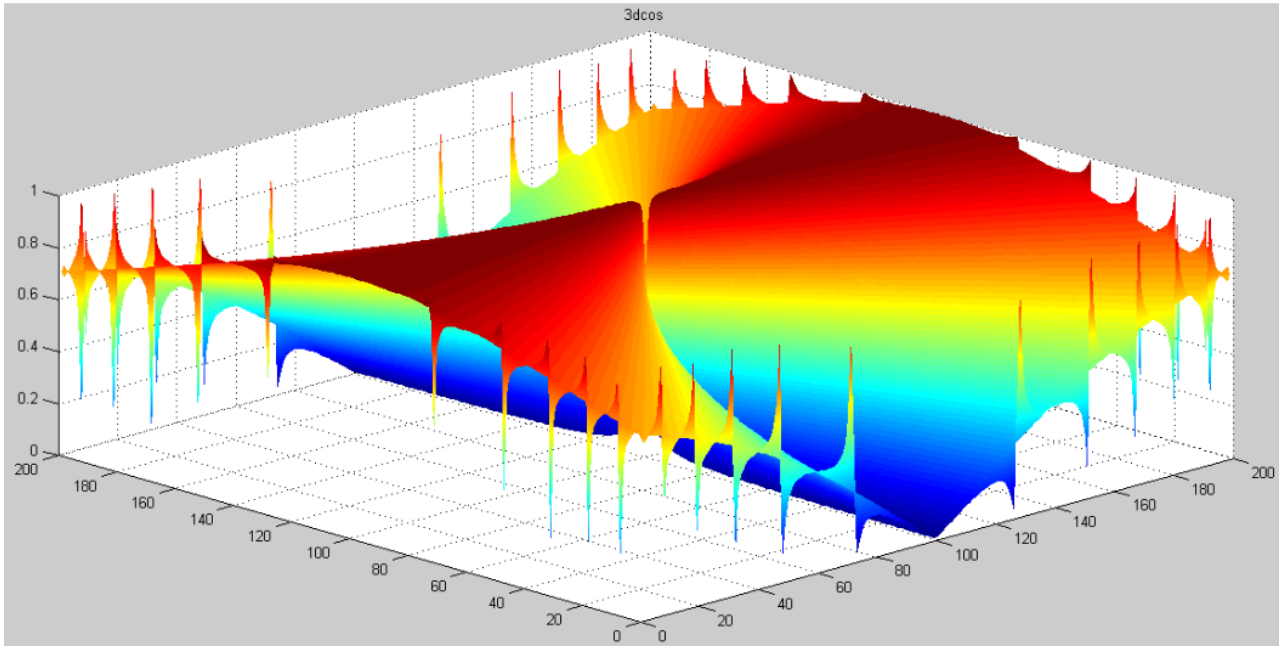


Figure 4.10: 3D surface map of the cosine map

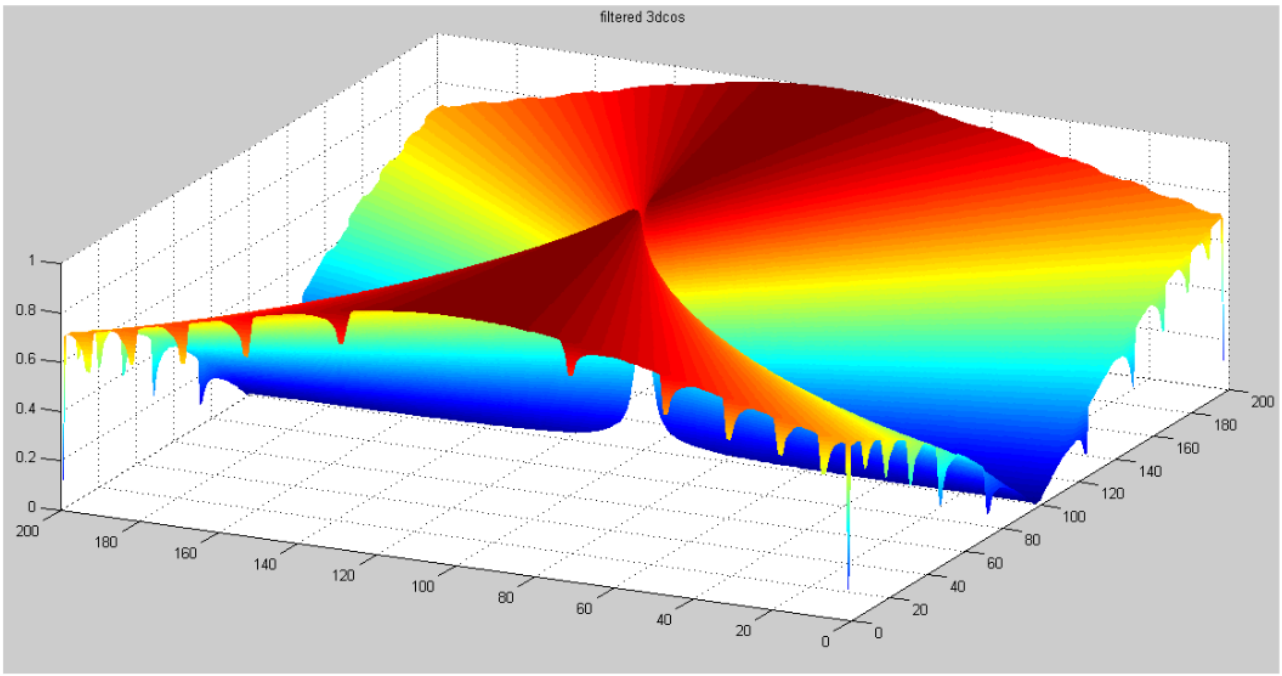


Figure 4.11: 3D surface map of the filtered cosine map

4.5 RESULTS – Experimental Fringes and Their Analysis

The following two figures are the fringe patterns experimentally obtained in the lab from the Michelson interferometric setup. The white boundary region in Fig-4.12 is Fig-4.13, the reason behind cutting out that portion is that the code perfected for analyzing the fringe patterns asks for two choices – one for open fringe patterns and the other for closed fringe patterns. At first I have presented the wrapped phase map, unwrapped phase map and finally the 3D phase map for both figures 4.12 and 4.13. It can be clearly seen that unlike the output for the code generated fringes the outputs over here are extremely rough owing to the presence of noise in the experimental fringes. Although the code used for analyzing the fringes filter the images prior to carrying out the other major operations, I have provided relatively unprocessed images over here which are sufficient for appreciating the phase map of the fringe patterns from where we can pass an educated verdict on the

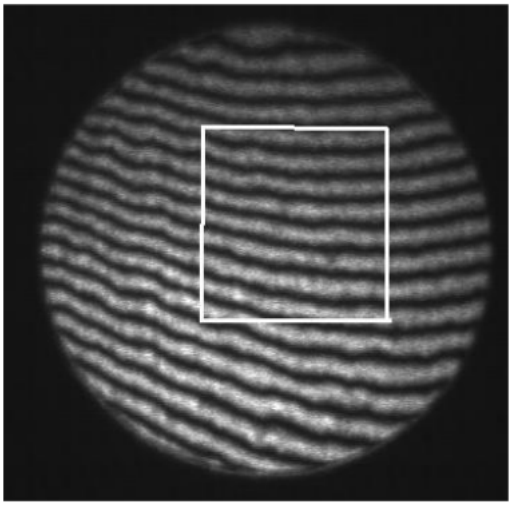


Figure 4.12: Experimentally obtained Interferrogram

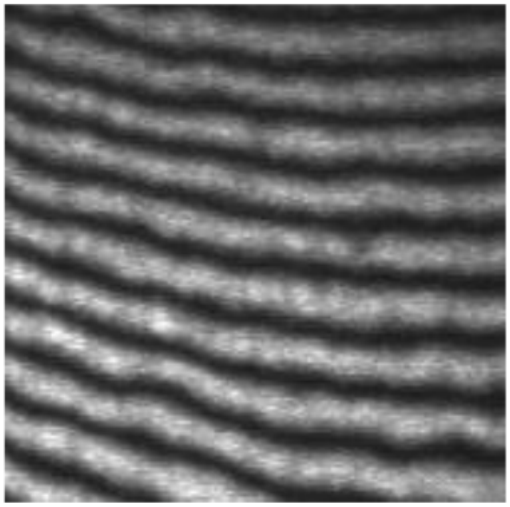


Figure 4.13: The selected portion of the fringe pattern

Figure 4.14: Wrapped phase map

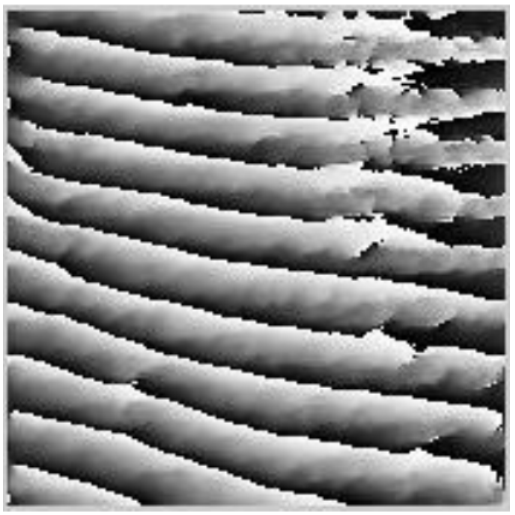


Figure 4.15: Wrapped phase map for the truncated portion

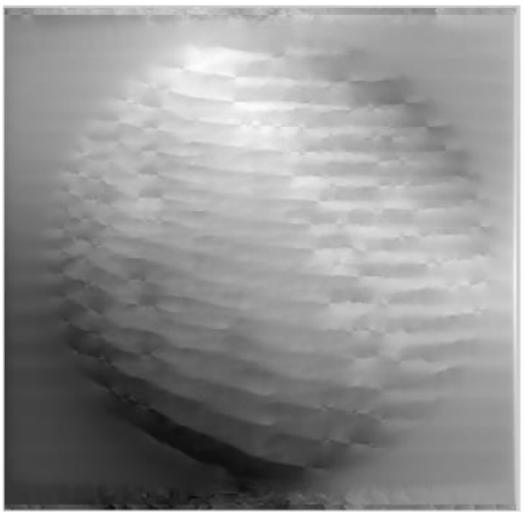


Figure 4.16: Unwrapped phase map

Figure 4.17: Unwrapped phase map for the truncated portion

profile of the material surface under study which in this case is the surface of the movable mirror of the Michelson interferometer.

The reason I have provided both the complete and truncated fringe patterns is to exemplify that having recorded an interferogram we can easily select our region of interest and then analyze that particular portion only. This is particularly relevant for the cases of non-destructive testing of materials, metrology etc., where we are required to find out the defects in the surface of the material under consideration.

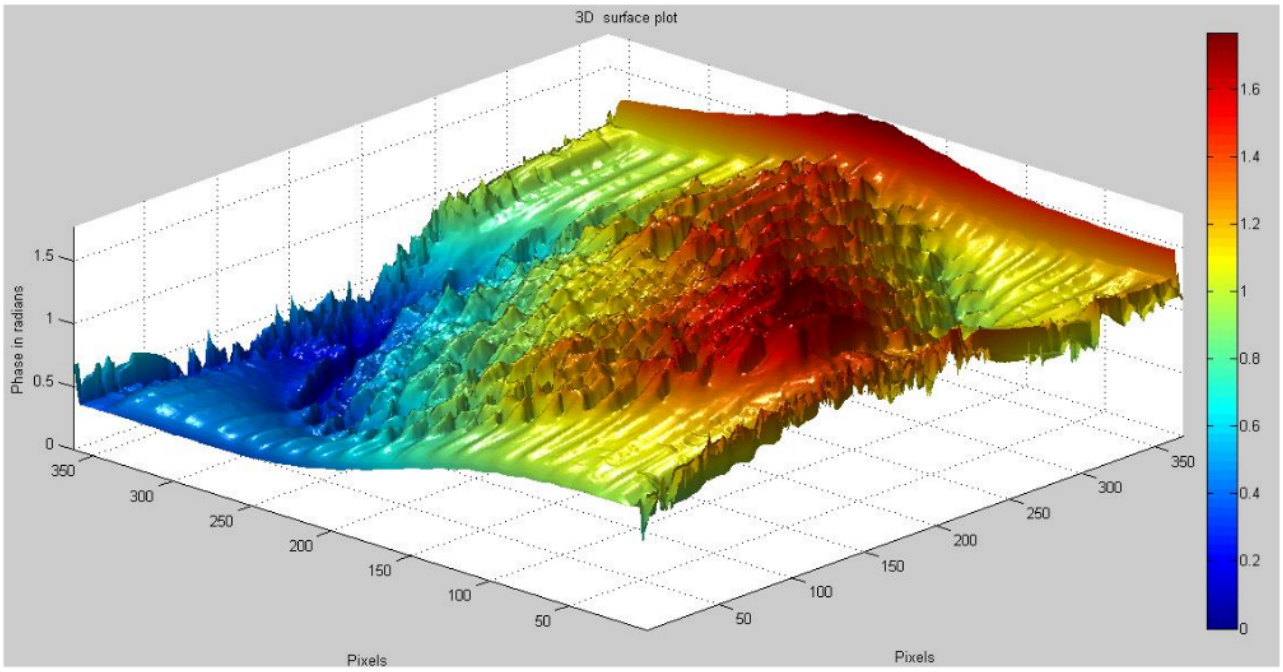


Figure 4.18: 3D surface plot of the entire phase map

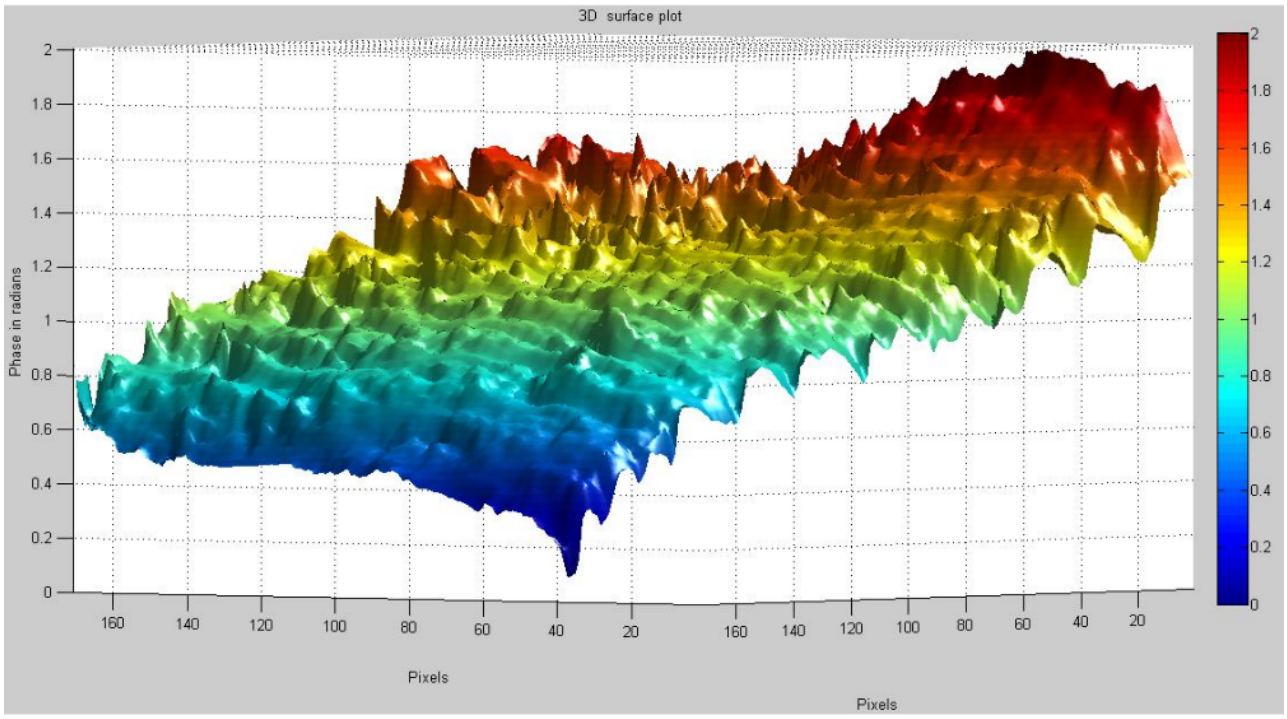


Figure 4.19: 3D surface plot of the phase map for the truncated region

Evidently from the Fig-4.19 we have a more detailed source of phase information about the specified region than what we would have obtained from its predecessor in Fig-4.18.

4.5.1 The Analysis–I(*Ugly*)

Recall that while naming this chapter I had referred to the fringe analysis as being ugly. A detailed discussion on this aspect of the analysis and post-processing is presented in this and the following subsection. First, let us inspect the signum map generated for our experimental fringes and the subsequent unwrapped phase map originating from it using the original simplistic algorithm.

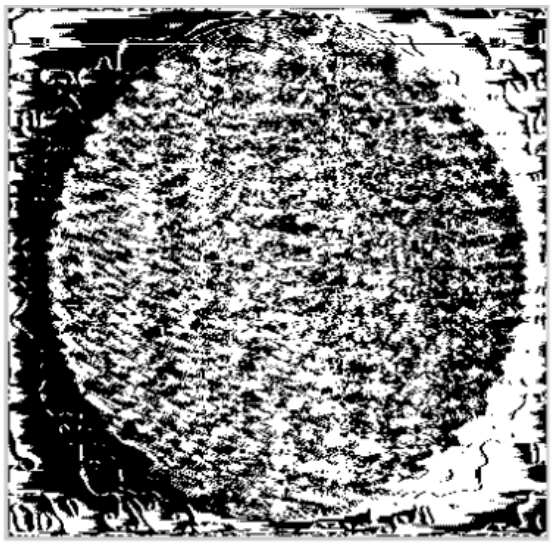


Figure 4.20: The signum map

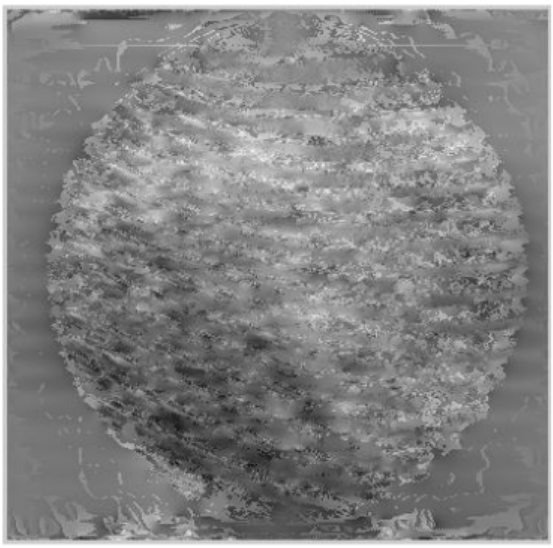


Figure 4.21: Corresponding unwrapped phase map

It is important to mention in this context that in the original algorithm suggested in Paul Kumar et al.(2011) they used the **differentiation operation** on the cosine map to obtain the signum map but differentiating the cosine map yields this 'ugly' noisy image from which it is hardly possible to get a smooth and acceptable phase map, which is our ultimate goal. Indeed the phase map obtained by using this signum map(Fig-4.20) is quite unlike the one in Fig-4.18 obtained from our modified algorithm.

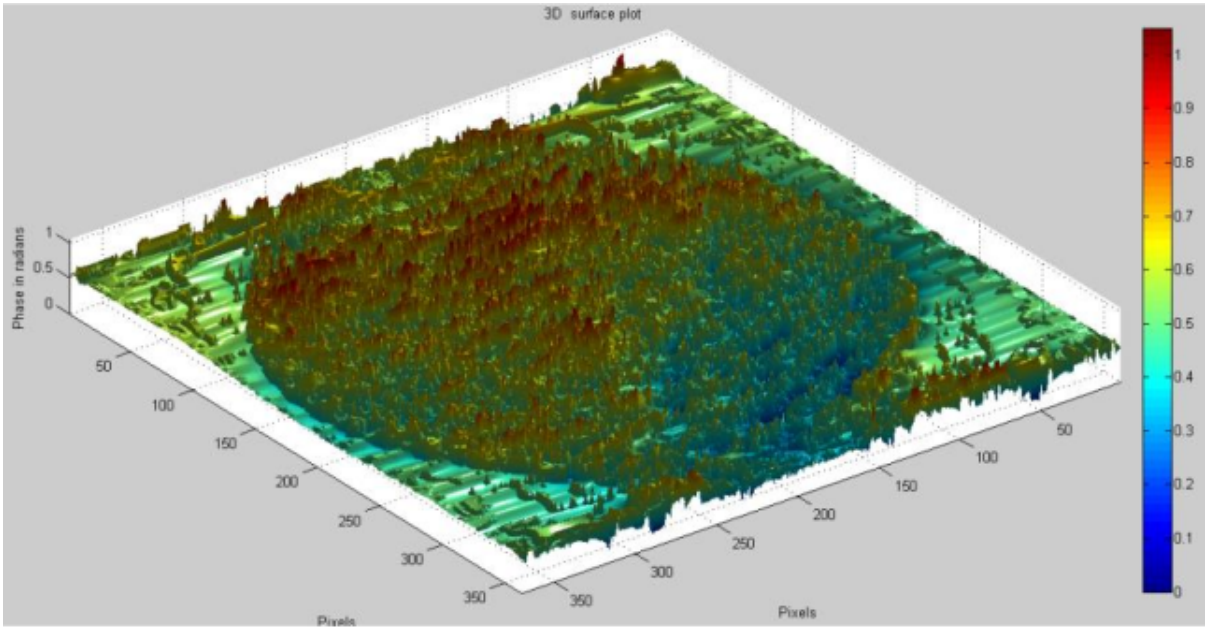


Figure 4.22: 3D surf plot after masking with the signum map in Fig-4.20

This particular drawback was not evident in their paper as they used a microlens array for their

analysis where the pattern on account of being unidirectional in terms of fringe orientation didn't leave any room for errors creeping into the output images. We will try to remove the noise introduced due to these errors by changing the 4^{th} step of the algorithm(please refer to the flowchart presented in section-4.2) as discussed in the next section.

4.5.2 The Analysis–II(*Not so ugly!!!*)

In order to adjust for the mal-effects of the errors mentioned above, we attempt to attain the signum map in a different way where instead of differentiating we **HT** the cosine map to get the signum map. This approach is perfectly logical as every image (fringe pattern) is basically a distribution of intensity where once we have removed the bias intensity terms we are left with the cosine term, now when we differentiate a cosine-term we get its sine but that is the same result we get when we apply **HT** to a cosine-term as well. So basically in terms of mathematics as related to its application based results in optics, we are not generating anything different(albeit a very crude justification and comparison). On trying out this approach the quality of the signum map was greatly enhanced as a result of which our final phase map was much better as compared to the previous one(obtained from the original algorithm). Now our experimentally obtained fringes were of very low quality and also quite simple. Hence, our new approach of obtaining the signum map cannot be said to have passed the acid test unless we verify it for some arbitrary fringe pattern whose orientation is pretty haphazard and unconventional. And if our approach works on this complex fringe pattern we can claim with absolute certainty that this is indeed a better way to generate the signum map *en route* to find the phase map. The following images bear witness to our claim. **Please note that the figures in section-4.5 are also due to our modified algorithm and not the original one**.

The image used to demonstrate our claim was not obtained in our lab as our experimental setup was not conducive to producing a similar arbitrary fringe pattern, it was downloaded from internet resources arbitrarily with the sole object of checking the credibility of our proposition. The set of images presented over here are the ones obtained implementing our correction to the analysis algorithm. The outputs for both the algorithms till the cosine map is the same but thereafter the outputs are different(as expected, obviously!), both the outputs have been provided side by side for comparison in the next page to illustrate the superiority of our approach over the previous approach.

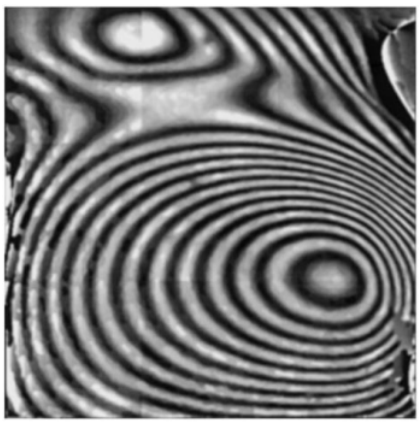


Figure 4.23: Original Interferogram

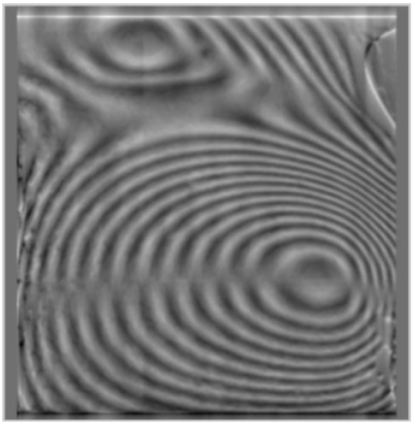


Figure 4.24: After applying HT

Figure 4.25: Fringe orientation map

It can be seen very clearly from the images presented below that the signum map obtained from the modified algorithm is much clearer compared to the one obtained from the original algorithm. In fact, the signum map obtained from the basic algorithm is more nearer to a spread of random noise bearing only a faint resemblance to the fringe orientation of the original image. Now the wrapped phase map, which has to be cured of its sign ambiguity featuring π jumps has been processed to yield the following subsequent images as obtained from both the algorithms.

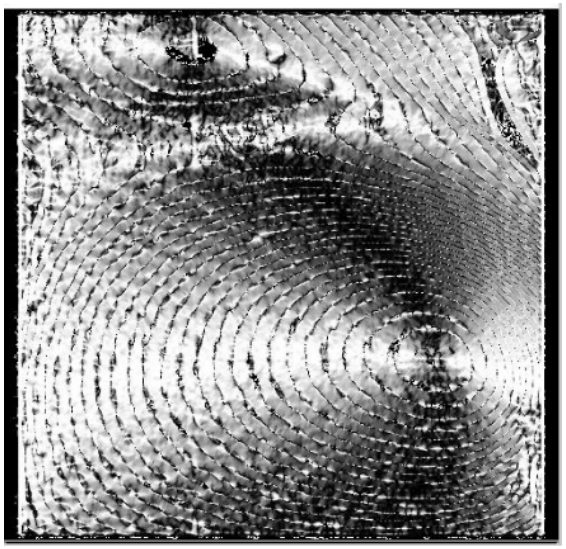


Figure 4.26: Cosine map

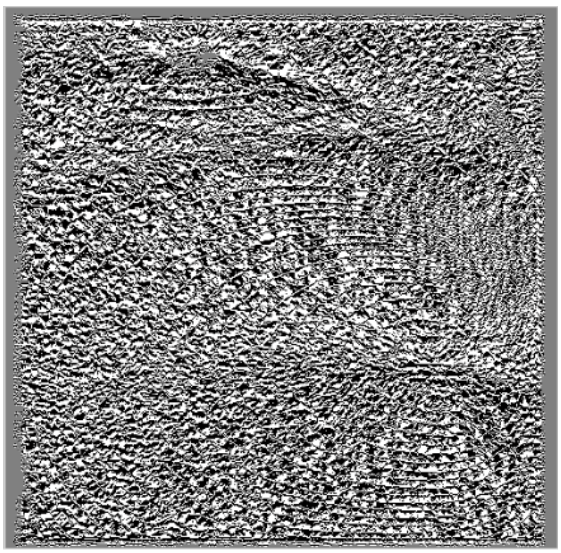


Figure 4.27: Signum map obtained from original algorithm

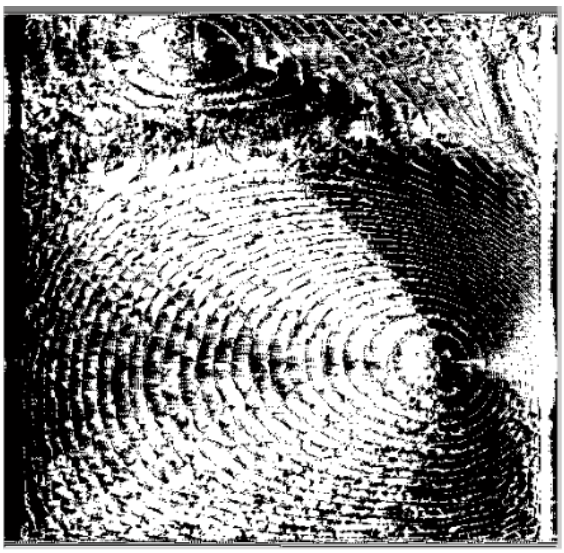


Figure 4.28: Signum map obtained from modified algorithm

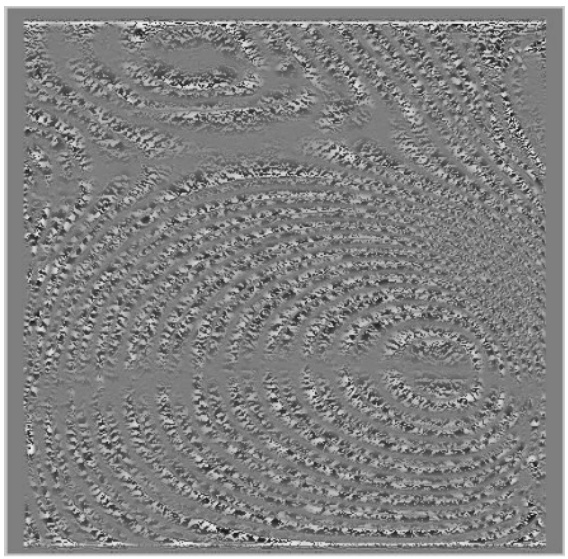


Figure 4.29: Corrected phase map as given by the original algorithm

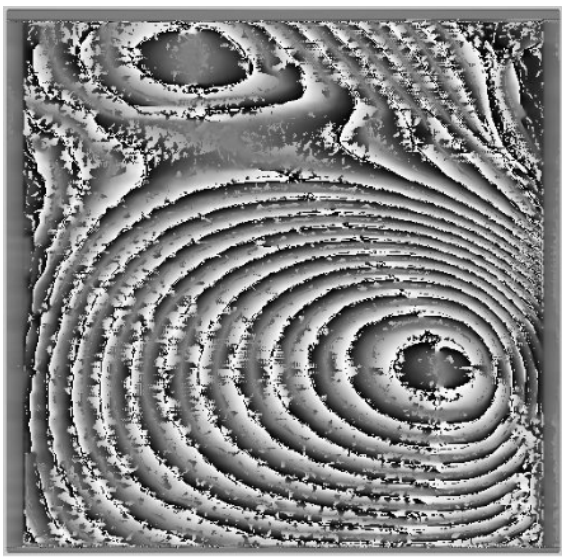


Figure 4.30: Corrected phase map as given by the modified algorithm

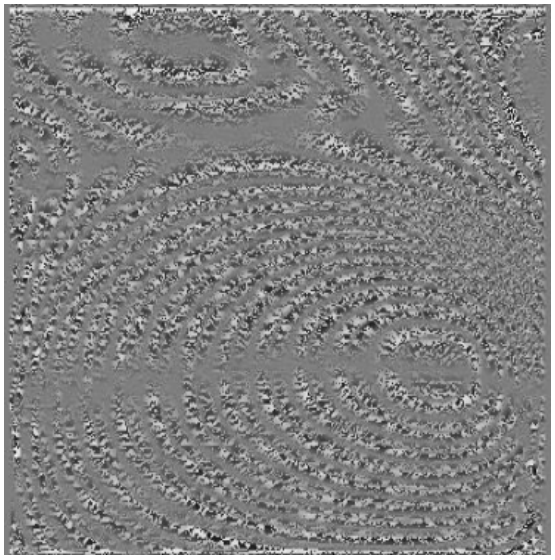


Figure 4.31: Corrected phase map as given by the original algorithm

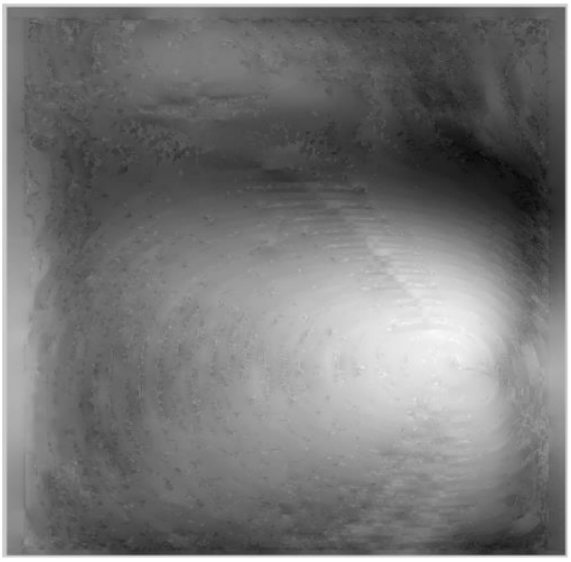


Figure 4.32: Corrected phase map as given by the modified algorithm

It is evident from the preceding figures that the troublesome noisy signum map obtained from the original algorithm has filtered down its error to completely ruin the subsequent corrected and unwrapped phase maps. Contrary to which the modified algorithm yields significantly smoother and noise-reduced corrected and unwrapped phase maps. Thus it may be said that the 'ugliness' in our analysis has been remedied satisfactorily and we can now carry out the business of phase extraction much more efficiently using our modified algorithm. In conclusion of this chapter, I would like to highlight that as always in the domain of science any theory can claim to be 'GOOD' purely based on the argument that it is logically consistent in terms of mathematics, but to come home & dry it must be verified experimentally. And if it falls short on that criteria and the experimental results turn out to bear an 'UGLY' contradiction with the existing theoretical framework then in order to circumnavigate that stumbling block we have to meander around in a different way and find novel solutions to achieve our goal as done on this occasion.

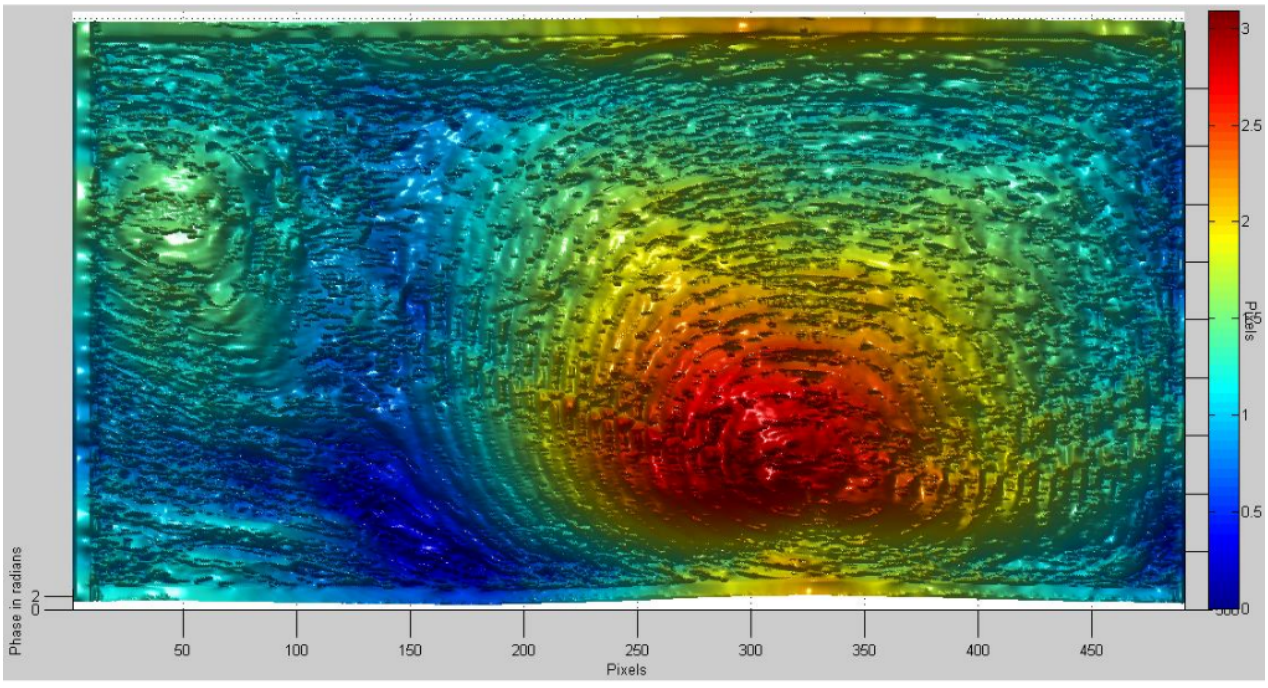


Figure 4.33: 2D intensity coded phase map of the fringe pattern

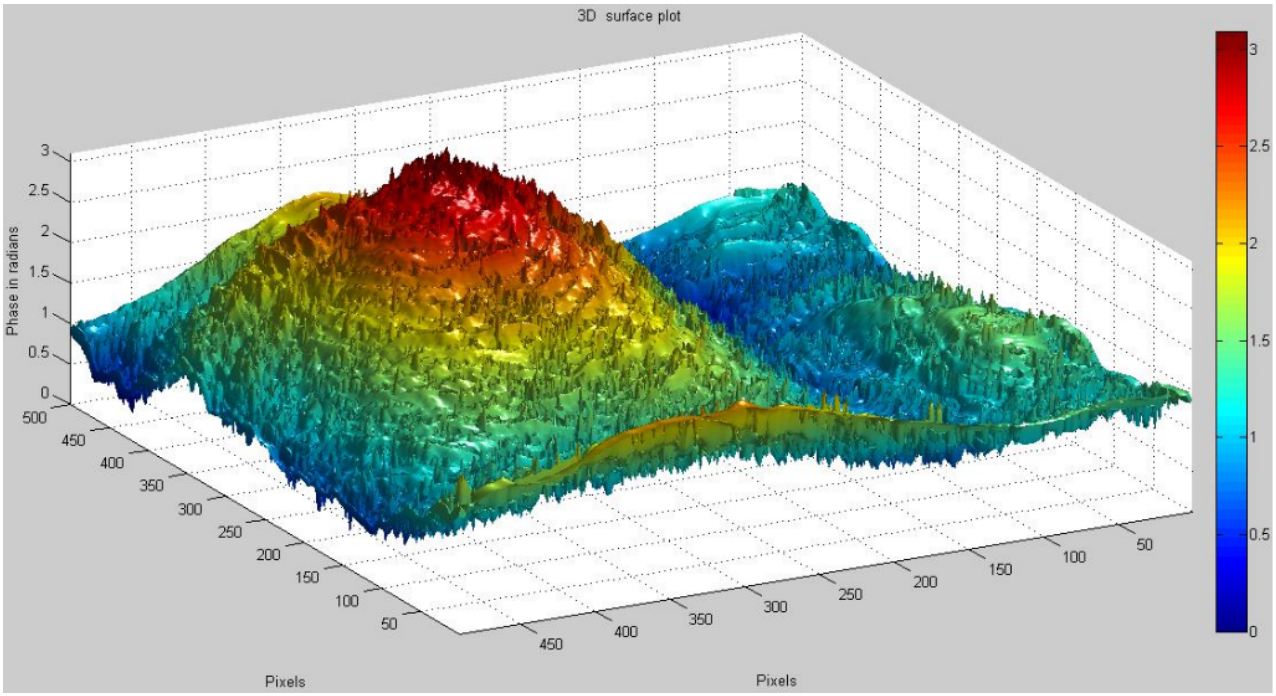


Figure 4.34: 3D surface plot of the fringe pattern

The figures 4.33 & 4.34 depict the final phase maps of the fringe pattern as obtained from the modified algorithm for the fringe pattern presented in Fig-4.23. It can be seen from the figures 4.33 & 4.34 that the images are still not entirely free from noise but in order to present absolutely pristine phase maps we would require source images which are better processed and have undergone significantly more filtering(which in turn corrupts the potential information stored in phase maps). However, these advanced processing and filtering techniques are beyond the scope of this project and its mandated time constraints. Although in spite of that I have done some work on processing, filtering etc on the source images to enhance the quality of our results which are discussed in greater detail in the next chapter.

Na karmaṇāmanārambhānnaiṣhkarmyaṁ puruṣhośnute।
na cha sannyasanādeva siddhiṁ samadhigachhati ।।4।।

— Karmayoga, Bhagavad Gita

"A person does not attain freedom from action and its influence by not undertaking actions, nor does he attain perfection i.e., Moksha or Liberation by renunciation alone."

5 IMAGE PROCESSING (TOOLS AND TECHNIQUES)

Having presented the results from our modified algorithm in the previous chapter it would have been perhaps the obvious choice to go on to mention the applications of the technique developed and conclude the project report. However, it is impossible to conclude this report without elaborating on one of the major aspects of 'action' undertaken in the project, namely image processing. Without carrying out the necessary operations of filtering and phase unwrapping it would have hardly been possible to extract the phase maps. Before going on to discuss the modes of filtering used, a brief description of the primary sources of noise relevant to this project has been put forward first.

Salt and Pepper noise – *Salt-and-pepper noise is a form of noise sometimes seen on images. It presents itself as sparsely occurring white and black pixels. An effective noise reduction method for this type of noise is a median filter or a morphological filter.*

Noise from digital cameras – *In low light, correct exposure requires the use of long shutter speeds, higher gain or both. In most cameras, longer shutter speeds lead to increased salt-and-pepper noise due to photodiode leakage currents. As we used a digital camera to capture our fringe patterns this is a prime source of noise in our fringe patterns.*

In the following section a discussion of filtering has been provided in detail followed by some other techniques used to enhance the image quality of the fringes we obtained experimentally.

5.1 Filtering Images

In signal processing, a filter is a device or process that removes from a signal some unwanted component or feature. Filtering is a class of signal processing, the defining feature of filters being the complete or partial suppression of some aspect of the signal. Most often, this means removing some

frequencies and not others in order to suppress interfering signals and reduce background noise. However, filters do not exclusively act in the frequency domain; especially in the field of image processing many other targets for filtering exist. Correlations can be removed for certain frequency components and not for others, without having to act in the frequency domain.

There are many different bases of classifying filters and these overlap in many different ways in other words there is no simple hierarchical classification. Filters may be of various types but only gaussian and median filtering are discussed over here as those are the ones discussed over here.

5.1.1 The Gaussian Filter

In electronics and signal processing, a Gaussian filter is a filter whose impulse response is a Gaussian function (or an approximation to it). Gaussian filters have the properties of having no overshoot

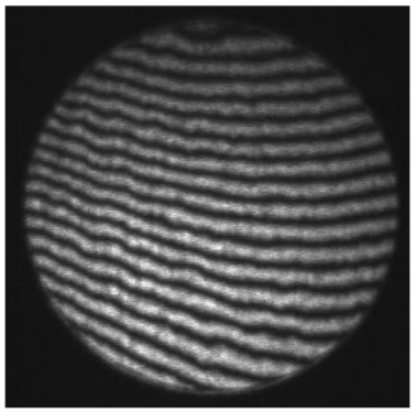


Figure 5.1: Original image

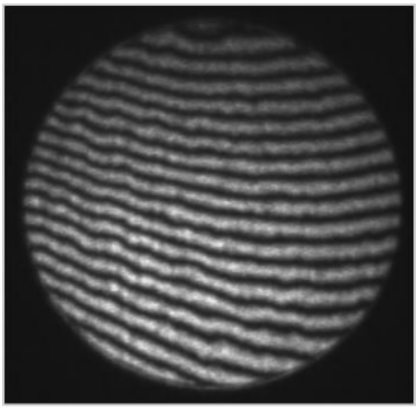


Figure 5.2: G-filtering ($\sigma = 0.5$)

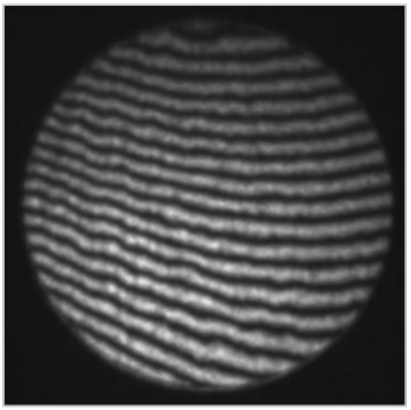


Figure 5.3: G-filtering ($\sigma = 1$)

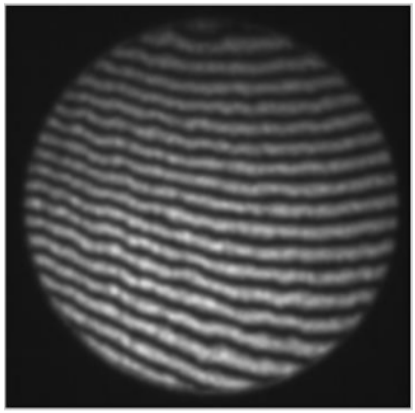


Figure 5.4: G-filtering ($\sigma = 1.5$)

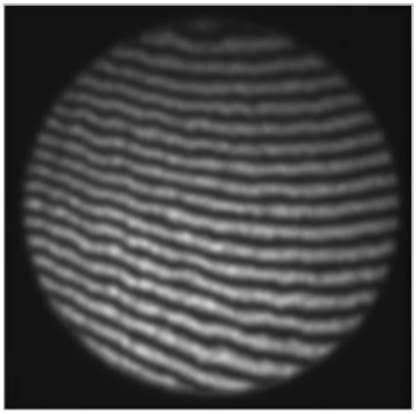


Figure 5.5: G-filtering ($\sigma = 2$)

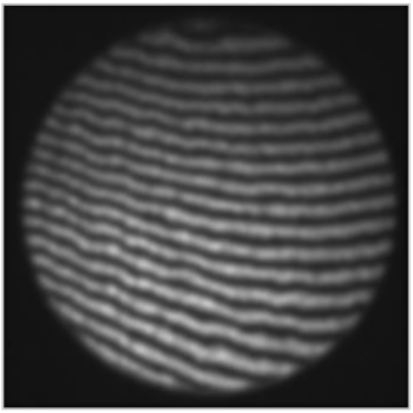


Figure 5.6: G-filtering ($\sigma = 2.5$)

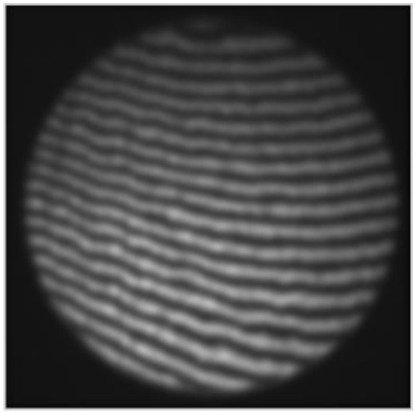


Figure 5.7: G-filtering ($\sigma = 3$)

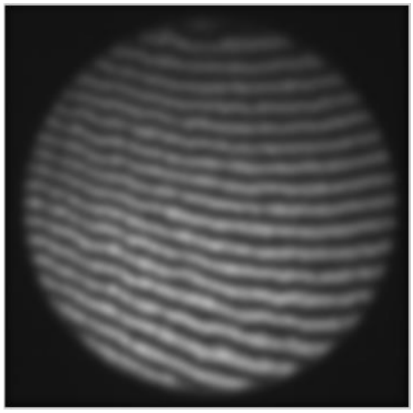


Figure 5.8: G-filtering ($\sigma = 3.5$)

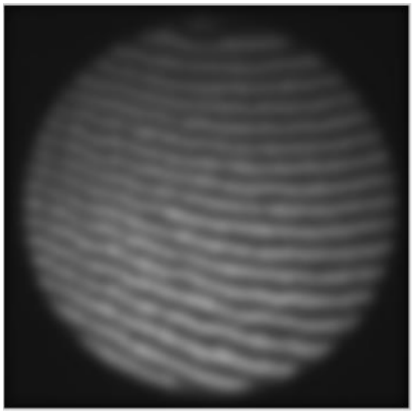


Figure 5.9: G-filtering ($\sigma = 5$)

to a step function input while minimizing the rise and fall time. This behaviour is closely connected to the fact that the Gaussian filter has the minimum possible group delay. It is considered the ideal time-domain filter, just as the *sinc* is the ideal frequency-domain filter. Mathematically, a Gaussian filter modifies the input signal by convolution with a Gaussian function. The functioning of the Gaussian filtering has been illustrated with the help of our experimentally obtained fringe pattern Fig-5.1. Filtered images for different values of standard deviation have been presented in figures 5.2 $\rightarrow$ 5.9.

As can be seen from the above images with a continuous increase in values of the standard deviation the image is extremely blurred out. Indeed for values of $\sigma = 5$ the original image loses a significant amount of its distinguishing features and is merely a blurry blob. Hence the value of the standard deviation must be chosen with great care so as not to compromise the individuality of the image in order to remove the noise present in the fringe pattern. There cannot be any fixed ideal value of the standard deviation and has to be fixed depending on factors such as the clarity and resolution of the captured image. But of course, for our purposes, median filtering is a much better option the reasons behind which are discussed in the next subsection.

5.1.2 The Median Filter

The median filter is a nonlinear digital filtering technique, often used to remove noise. Such noise reduction is a typical preprocessing step to improve the results of later processing (for example, edge detection of an image). Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise. Now if observed carefully it can be seen from the figures in the previous section(Fig-5.1 $\rightarrow$ Fig-5.9) that the gaussian filter is not able to preserve the edges at all i.e., why we need to implement median filtering.

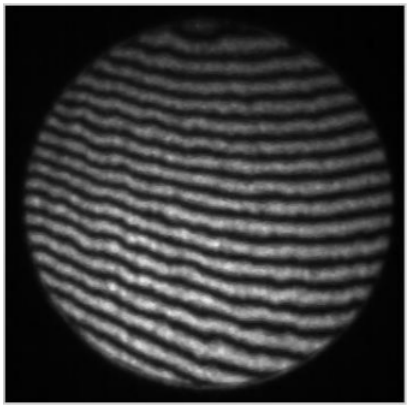


Figure 5.10: M-filtering (WS 3×3)

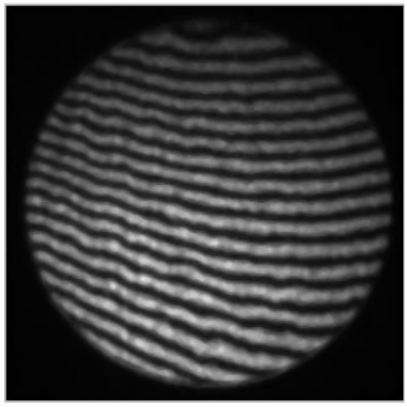


Figure 5.11: M-filtering (WS 5×5)

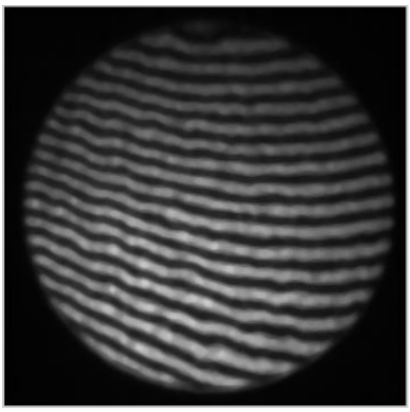


Figure 5.12: M-filtering (WS 7×7)

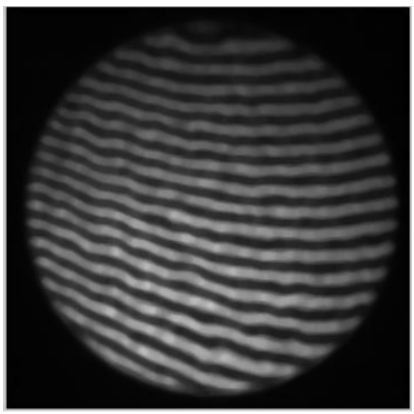


Figure 5.13: M-filtering (WS 9×9)

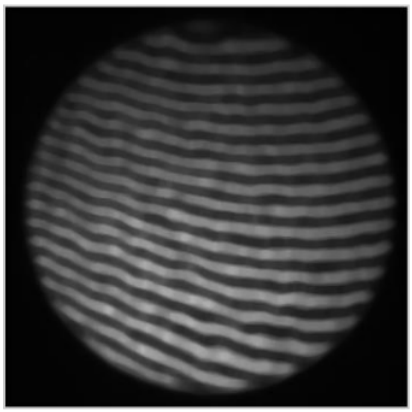


Figure 5.14: M-filtering (WS 11×11)

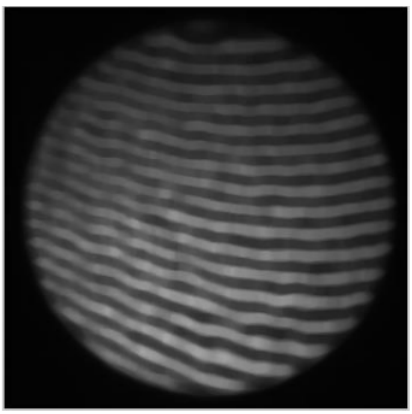


Figure 5.15: M-filtering (WS 13×13)

The functioning of median filtering has been explained as follows. The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of neighbouring entries. The pattern of neighbours is called the "window", which slides over the entire signal entry by entry. For 2D (or higher-dimensional) signals such as images, complex window patterns are possible (such as "box" or "cross" patterns). The window pattern used by us is the box. Note that if the window has an odd number of entries, then the median is simple to define: it is just the middle value after all the entries in the window are sorted numerically, this is the reason why we have chosen **window sizes(WS)** 3×3, 5×5 and 7×7 respectively in our code. Thus a Median filter replaces the pixel at the centre of the filter with the median value of the pixels falling beneath the mask. For an even number of entries, there is more than one possible median, which is inconvenient to deal with. The images obtained after median filtering with different window sizes have been presented in the figures(5.10 $\rightarrow$ 5.15).

It can be seen very clearly from the above figures that unlike gaussian filtering the median filtering preserves the edges of the fringe pattern. Of course, as we keep on increasing the window size(WS) the image keeps getting blurred in a trend similar to the effect of increasing σ on gaussian filtering. In the previous chapter-4 where our experimental results have been presented, we chose the optimum standard deviation and window size for the applied filters and then extracted our phase maps.

However, it must be noted in this context that for image processing we always use low pass filters, which block the high-frequency content of the image and as high-frequency components correspond to boundaries of the objects, the resulting blurring takes place near the edges of the images.

5.2 Phase Unwrapping

It may be recalled that the expression for the phase is given by an arctangent function as a result of which the amplitude of the phase can assume any value, which in general exceeds the range $[-\pi, +\pi]$. Now in the cases where the phase exceeds this range of values, it will be *wrapped* so that it can be accommodated within the normal range $[-\pi, +\pi]$. For the instances where this so called *wrapping* does take place the wrapped phase will contain one or more 2π jumps. It is interesting to note that actually in the case of digital signal processing the quadrant arctangent function is employed to calculate the phase map. The **four quadrant arctangent** $\arctan 2$ is calculated using the following equation:–

$$\arctan 2(a,b) = \begin{cases} \tan^{-1}\dfrac{a}{b} & a > 0 \;\;\&\;\; b > 0 \;\; \text{first quadrant} \\ \tan^{-1}\dfrac{a}{b} + \pi & a > 0 \;\;\&\;\; b < 0 \;\; \text{second quadrant} \\ \tan^{-1}\dfrac{a}{b} - \pi & a < 0 \;\;\&\;\; b < 0 \;\; \text{third quadrant} \\ \tan^{-1}\dfrac{a}{b} & a < 0 \;\;\&\;\; b > 0 \;\; \text{fourth quadrant} \end{cases} \tag{5.1}$$

where a and b are real numbers.

Suppose we express the wrapped phase as $x_W(n) = W[x(n)]$ where $x(n)$ is the original continuous phase signal, $W[\,]$ is the phase unwrapping operation and $x_W(n)$ is the wrapped phase signal. Now the 2π jumps that are present in the wrapped phase signal must be removed in order to restore the phase signal that is free from 2π jumps. The basic phase unwrapping process can be explained by splitting the task down into the following steps.

i) Start with the second sample from the left in the wrapped phase signal $x_W(n)$.
ii) Evaluate the difference between the current sample and its directly adjacent left-hand neighbour.

iii) If the difference between the two adjacent samples described in **ii** is larger than $+\pi$, then subtract 2π from the current sample and so on.
iv) If the difference between the two adjacent samples described in **ii** is smaller than $-\pi$, then add 2π to the current sample and so on.
v) If all the samples in $x_W(n)$ are processed then terminate the program or else repeat the cycle from step **ii**.

The unwrapping process can be mathematically expressed as,

$$x_U(n) = U[x_W(n)] = x_W(n) + 2\pi k \tag{5.2}$$

where $U[\,]$ is the phase unwrapping operation, $x_U(n)$ is the unwrapped phase signal and k is an integer chosen appropriately.

It must be mentioned over here that there are two daunting challenges for phase unwrapping namely the presence of noise in the wrapped phase map and the wrapped signal being under sampled. The effects of these factors are discussed in the following paragraphs.

In order to perform the phase unwrapping process, the difference between a sample and the preceding sample (directly adjacent on its left) is calculated. When this difference is larger than $+\pi$, or smaller than $-\pi$, a phase wrap is detected. Once a phase wrap is detected, the value of 2π is either added or subtracted, to/from this sample and also from all the further samples to the right-hand side of it. Now consider that the phase wrap that has just been detected can be either a "genuine phase wrap", or might also be a "fake wrap" that has been produced by noise in the signal. It is this distinction between true phase wraps and apparent phase wraps(that have been caused by noise) that make the practical phase unwrapping problem such a challenging task.

In the case where a genuine wrap has been detected, the phase unwrapping process continues its operation successfully. The existence of a fake wrap will affect the unwrapping of the sample, where the fake wrap is located, and will also affect all the samples to the right of the wrap. This makes phase unwrapping such a delicately intricate task, as any error in the determination of only a single phase wrap may affect a significant part of the entire signal because this error then propagates to the rest of the samples to the right-hand side of the problematic "detected" wrap. Thus clearly from the above portion it is evident how the presence of noise in the image destroys the concept of recovering "true phase" by phase unwrapping due to the presence of said "fake wraps".

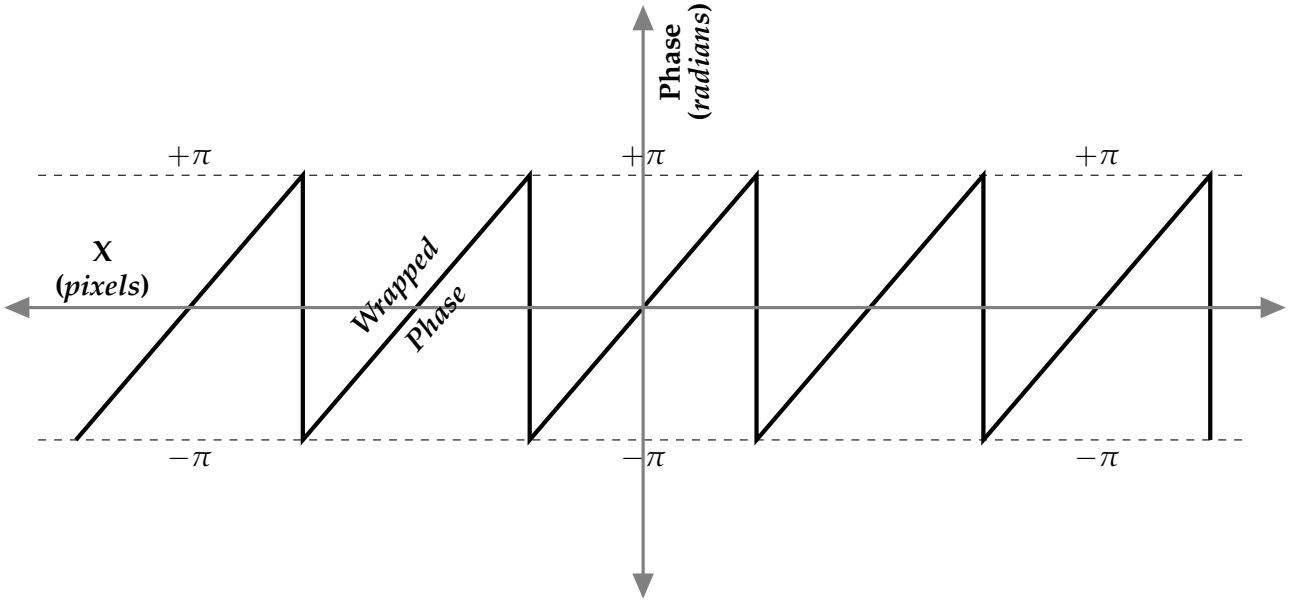


Figure 5.16: Wrapped phase oscillating in between $-\pi \rightarrow +\pi$

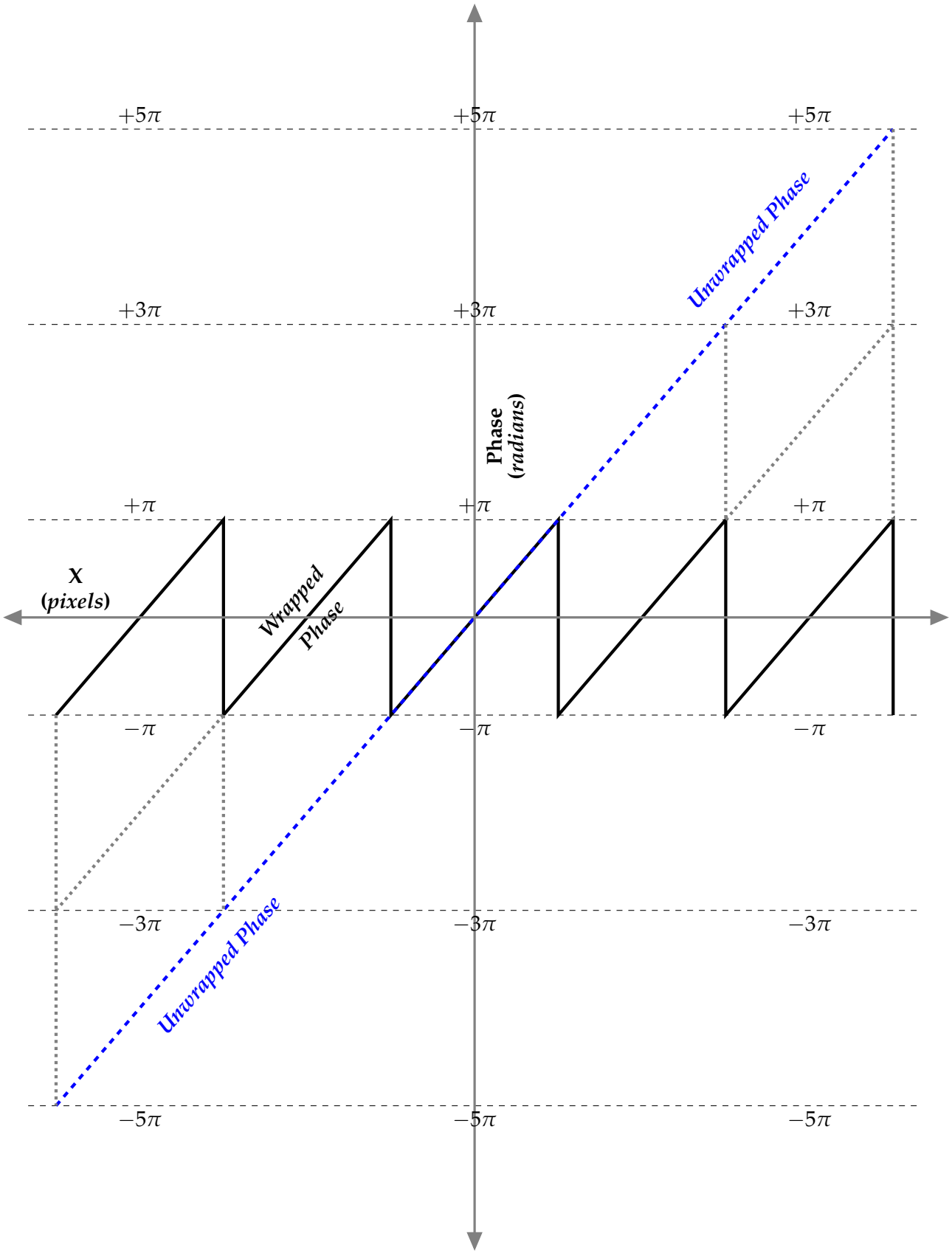


Figure 5.17: Unwrapping the phase map

In order to unwrap the image, we have used Itoh's 2D phase unwrapping algorithm(Itoh (1982)). There are two main methods by which the Itoh 2D phase unwrapping may be implemented. The first method involves unwrapping the rows in the wrapped image sequentially (one at a time). This

produces an intermediate image that is only partially phase unwrapped. Next, we perform a similar process, but this time unwrap all the columns within the partially unwrapped image. The second method of implementing Itoh's unwrapping simply works the other way around. In other words, it involves first unwrapping all the columns within the wrapped phase image, one at a time. And this again produces a partially phase unwrapped image. Then we sequentially unwrap all rows of the partially unwrapped image. Fig-5.16 & Fig-5.17 depict a schematic diagram to show the wrapped phase and exactly how the process of unwrapping is carried out in practice respectively.

6 APPLICATIONS – A PROPOSITION FOR DETECTING MALARIA

6.1 A Brief Discussion on Malaria Infected RBCs

In order to better understand our proposition, it is essential to understand exactly what the nature of Malaria infection is and how it affects the red blood cells(RBCs). I have attached the blood sample figures as obtained from internet sources in order to explain the nature of the disease and why finding the phase map of the RBC can possibly lead to the detection of the disease.

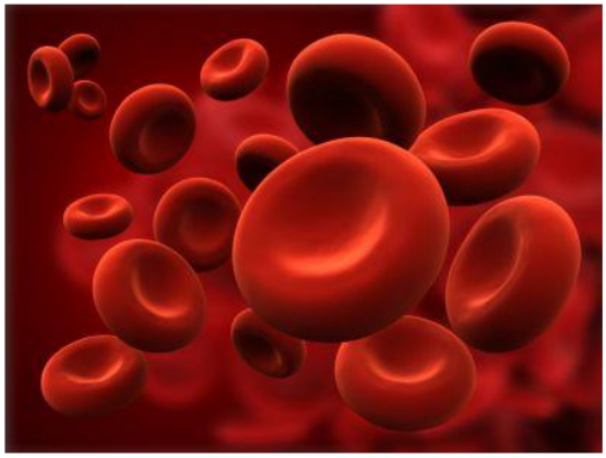


Figure 6.1: Red blood cells of a healthy person

Figure 6.2: Red blood cells of a Malaria infected person

Image courtesy– The National Geographic Society

It can be easily seen from the two above figures that on being infected by the disease the surface of the RBCs tend to get deformed as a result of which tiny bulges start appearing on their surfaces. Now obviously a phase map will display the irregularities occurring on the surface of the RBCs. The **HT** method can detect the phase map from any intensity coded figure be it recorded by microscope or anything else. Also as this technique of phase map evaluation is based on the ideology of phase extraction from a single recorded image the aspect of time consumption is managed very effectively as far as the detection of the disease goes. We propose that a database be made of blood samples from regions that are prone to be affected by the disease and then our proposition be tested out to check out its applicability. The proposition is discussed in greater detail in the next section.

6.2 Detection of Malaria ?

As mentioned in the previous section we propose a new and advanced method of detecting diseases that affect the human RBCs in this section citing the example of Malaria. The analysis of image patterns using **HT** can be carried out for serving many important purposes like detection of Malaria and similar diseases. The reason is that as **HT** demands the requirement of only one fringe pattern a simple microscope captured image of the blood sample will be enough for accurately detecting the traits of Malaria present in the sample. The basic logic behind this is that when a person gets infected by Malaria the seeds of the disease anchor themselves in the RBCs and after a certain point of time the RBC starts swelling up as a result of which the RBC finally bursts. Now, these red blood cells which have started swelling up will obviously have a different phase map due to the unevenness of the surface of the red blood cells as compared to the blood cells of a perfectly healthy person. Now having found the phase profile of both kinds of blood samples we can simply superpose the to phase maps in order to ascertain the measure of the bulging of the RBCs which will ultimately result in pinpointing the extent of the infection of the disease. Thus this can be treated as a very novel and effective way of detecting a very fatal tropical disease. Diagnosis of malaria must be rapid, accurate, simple to use, portable and low cost, as suggested by the World Health Organization (WHO). Despite recent efforts, the gold standard remains the light microscopy of a stained blood film. This method can detect low *parasitemia* and identify different species of *Plasmodium*. However, it is time-consuming, it requires well-trained microscopists and good instrumentation to minimize misinterpretation, thus the costs are considerable. Moreover, the equipment cannot be easily transported and installed, whereas our **HT** based proposition is a much better way of detecting the disease as well as being less time-consuming.

After downloading some patterns of blood samples infected with Malaria and applying our algorithm to them we find that the resulting phase maps were of very rough quality. The reason mainly being the quality of the image downloaded from the internet was extremely poor. However, an experimental setup can be set up in our biotechnology department for recording the speckle patterns of a malaria-infected blood sample. And after those superior images are adequately processed we have no doubt that our algorithm will provide extremely useful results. Although our output images were very poor still I have attached the resultant images to support the potential applicability of our proposition, pending the previously mentioned adjustments and enhancements which are beyond the scope of our present project.

If we subtract the phase map of RBC from an infected person(Fig-6.8) by the phase map of RBC of an uninfected person we will only have the phase map of the symptomatic bulges on the surface of the infected RBC which will give a good measure of the extent of the infection of the disease. However, this is just a proposition the ultimate verification of which is beyond our scope as we do not possess the necessary tools for carrying out a biologically sensitive experiment of collecting the blood samples. For instance, even the slightest amount of adulteration to the sample will be registered as a bulge in the phase map which may not be due to the infection of Malaria at all. Thus we need an extremely well-sanitised environment for carrying out this methodology of detection of Malaria by **HT** method so that we can claim beyond any doubt that the diagnosis suggested from the results of the test are perfect. Also it will help us to ascertain what adaptations and modifications

are needed to make it practicable to use on a regular basis without compromising on the expected medical standards.

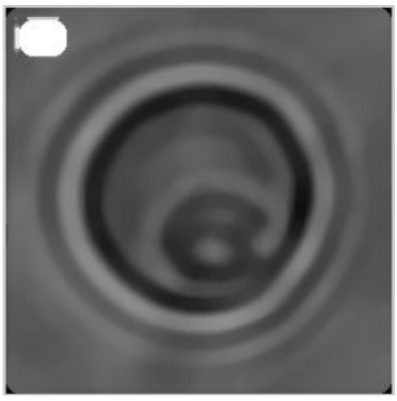


Figure 6.3: Infected RBC

Figure 6.4: Fringe orientation map of infected RBC

Figure 6.5: Cosine map of infected RBC

Figure 6.6: Signum map of infected RBC

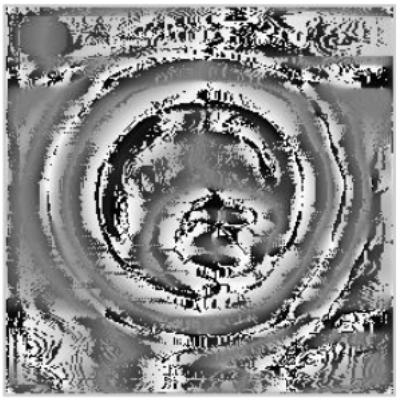


Figure 6.7: Wrapped phase map of infected RBC

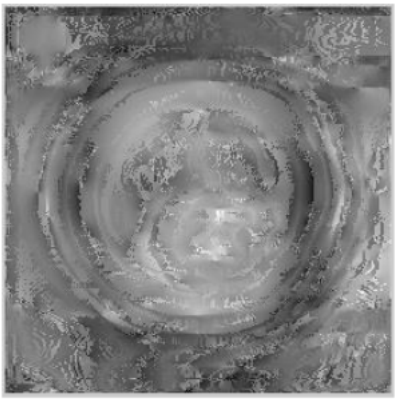


Figure 6.8: Unwrapped phase map of infected RBC

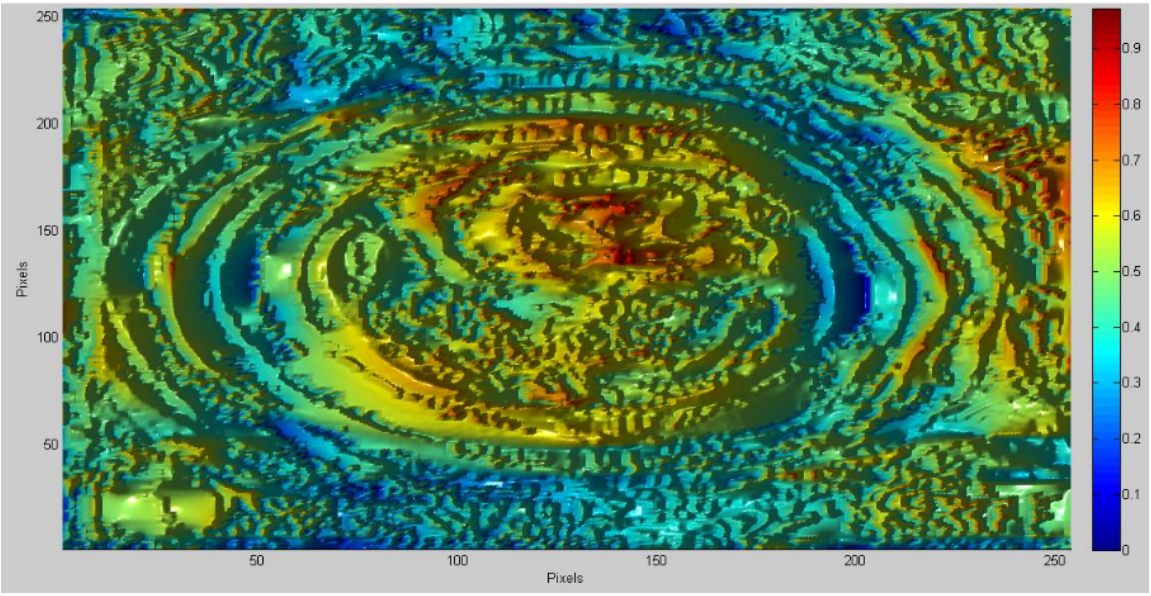


Figure 6.9: Intensity coded 2D map of the infected RBC(the red portions correspond to bulges in the surface of the RBCs owing to the infection)

7 CONCLUSION

Throughout the project, we have concentrated on the methodology of phase extraction by **HT** method. Our primary motivation behind this was to establish an alternative to the time-consuming processes of phase-shifting interferometry. This project demonstrates why it is imperative to discard the old processes and switch over to this better and improved technique of using **HT** dependent algorithm for estimating the phase map of the objects under investigation. Although our experiment was conducted on a simple Michelson interferometric setup nevertheless it demonstrates the effectiveness of the method beyond any doubt, which was the ultimate aim of the project. Complex experimental setups may be designed with the object of finding the phase map of a material surface which will have deeper consequences in fields such as metrology, non-destructive testing of surfaces etc. But once having obtained the required interferogram we can simply invoke our algorithm to extract the phase map and thus analyse the defects etc on the surface of a material sample.

Moreover, during this project, we were successful in determining the necessary modifications to the original algorithm to make it an even more effective tool than before. As has already been mentioned in Chapter-4 how in the course of our investigations we found out that the original algorithm in Paul Kumar et al.(2011) was not consistently applicable to all kinds of fringe patterns. Indeed it was a specialized case where the phase map obtained was based on the designing of the signum map by differentiating the cosine map. However, as we have shown the method of constructing the signum map is much more effective when done by using the **HT** on the cosine map and then extracting the imaginary values of the obtained function. It is our sincere belief that this change although small in itself has a large consequence in terms of results obtained by its implementation. And indeed that is what has been demonstrated in Chapter-4 where we have compared the output images as obtained from both the algorithms and the difference is clearly visible.

In conclusion, it would be apt to mention that the ultimate endeavour of any branch of science is to reach forward towards a greater form of truth from a comparatively lesser one. Thus it is quite plausible that a more elegant approach for evaluating the phase maps will be perfected in the coming future. Thus I conclude this project report on this note of striving towards one form of truth to another, or rather from darkness towards 'light' as our ancient texts say because that is the defining

quality of science, to treat every emerging idea as light and receding ideology as darkness.

Asato mā sad gamaya |
Tamaso mā jyotir gamaya |
Mrtyor mā aḿrtam gamaya |
Om shantih shantih shantih | |

— Brihadaranyak Upanishad

Lead us from the untruth to the Truth.
Lead us from darkness to light.
Lead us from oblivion to immortality.

APPENDIX–I

MATHEMATICA CODES USED IN THE PROJECT

Code for performing Hilbert Transform on functions

```
HilbertTransform[f_, x_, y_, assum___?OptionQ] :=
 Integrate[f/(x - y), {x, -Infinity, Infinity},
   PrincipalValue -> True, assum]/Pi

HilbertTransform[f_, x_, y_, a_, assum___?OptionQ] :=
 Integrate[f/(x - y), {x, -a, a},
   PrincipalValue -> True, assum]/Pi

HilbertTransform["Input the function with x in the argument", x, y]
```

Code to perform Discrete Hilbert Transform on function which behaves halfway through as a triangular function and then as a square wave, as an aid to illustrating how MATLAB's hilbert command performs(as MATLAB'S hilbert command performs discretely)

```
HilbertSpectrum[0] := {};
HilbertSpectrum[n_Integer?Positive] :=
 With[{nhalf = Quotient[n - 1, 2]},
 Join[{0}, ConstantArray[-I, nhalf], If[EvenQ[n], {0}, {}],
   ConstantArray[I, nhalf]]];

Hilbert[data_?VectorQ, padding_Integer?NonNegative] :=
 Module[{fp = FourierParameters -> {1, -1}, n = Length[data], m,
   paddeddata}, m = n + padding;
   paddeddata = PadRight[data, m];
   Re@InverseFourier[HilbertSpectrum[m] Fourier[paddeddata, fp], fp][[;;
     n]]]

With[{n = "Choose the no.of points"},
data = N@Join[Table[TriangleWave[3 k/n], {k, n}],
   Table[SquareWave[3 k/n], {k, n}]];]
ListLinePlot[{data, Hilbert[data, 256]}, GridLines -> Automatic,
 PlotStyle -> {Thickness[0.0062]}, ImageSize -> {600},
 PlotMarkers -> {Automatic, Large}, Frame -> True,
 FrameLabel -> {x, f (x)}, LabelStyle -> Directive[Black, Large],
 AxesStyle -> Thickness[0.002]]
```

APPENDIX–II

MATLAB CODES USED IN THE PROJECT

Code in MATLAB for Phase Extraction by the HT Method

```
clc
clear all
% clears your workspace and command window, so you can start fresh
% loading the input image
reading =input('Enter the location of fringe pattern :');
% Generates circular fringes
if reading == 1
   x=linspace(-4,4,200);y=x';
   [X,Y]=meshgrid(x,y);
   r=X.^2+Y.^2;
   z=cos(r+30);
   image=z;
   clear z r X Y x y;
else
   image = (imread(reading));
end
normimage= mat2gray(image);
% Normalizing the image
[A, B, C] = size(image);
if C~=1
   imagegray = rgb2gray(normimage);
% 3D to 2D conversion
else
   imagegray = normimage;
end
% Option to chose no of times filter is applied & window size of
% filter that is applied
l=input('Enter the Number of times for Filtering :');
y=input('Enter the length of the averaging window \n(3 or 5 or 7 or 9 or 11 or 13 or 15 or
    17 or
19):');
K=input('Enter type of fringe pattern \n(close=0,open=1):');
for k=1:l
   imagegray = medfilt2(imagegray,[y,y]);
end
V = imagegray;
for i=1:1:A
   VV=V(i,:);
   VV=double(VV-mean(VV));
% bias intensity is removed
   g(i,:)=VV;
% Applying Hilbert transform
   HH=hilbert(VV);
   IM=imag(HH);
% Applying arctangent funtion
   T=atan2(IM,VV);
   b(i,:)=T;
```

```
   NN(i,:)=IM;
   DD(i,:)=VV;
end
% finding of theta
[fx,fy]=gradient(double(g));
theta=atan(-fy./(fx+10^(-7)));
M=cos(theta);
for i=1:1:A
   MM=M(i,:);
% Modification of the original algorithm where we use HT instead of
% differentiation for obtaining the Signum map
   MHH=hilbert(MM);
   MIM=imag(MHH);
   TM=atan2(MIM,MM);
   Mb(i,:)=TM;
end
VQ=sign(Mb);
if K==0
   P=b.*VQ;
else
   P=b;
end
% Unwrapping the wrapped phase
[m, n]=size(P);
for i =1:m
   for j= 1:n
      if (j==n)
         a=0;
      else
         a=P(i,j+1)-P(i,j);
      end
      if (a>=pi)
         a=a-2*pi;
      elseif(a<=-pi)
         a=a+2*pi;
      end
      if (j==1)
         b=0;
      else
         b=P(i,j)-P(i,j-1);
      end
      if (b>=pi)
         b=b-2*pi;
      elseif(b<=-pi)
         b=b+2*pi;
      end
      if (i==m)
         c=0;
      else
         c=P(i+1,j)-P(i,j);
      end
      if (c>=pi)
         c=c-2*pi;
      elseif(c<=-pi)
         c=c+2*pi;
      end
      if (i==1)
         d=0;
      else
         d=P(i,j)-P(i-1,j);
      end
```

```
      if (d>=pi)
         d=d-2*pi;
      elseif(d<=-pi)
         d=d+2*pi;
      end
      R(i,j)=(a-b+c-d);
      clear a b c d;
   end
end
r=dct2(R);
clear R;
for k=1:m
   for l=1:n
      q(k,l)=r(k,l)/(2*cos(pi*k/m)+2*cos(pi*l/n)-4);
   end
end
clear r;
Q=2*idct2(q);
clear q;
a = input('Enter lambda value in nm:');
lam = a/1000;
S=Q*lam/(4*pi);
S=S-min(min(S));
if K == 0
   figure,imshow(V)
   title ('filtered image')
   figure,imshow(theta)
   title ('Fringe orientation map')
   figure,imshow(cos(theta))
   title ('Cosine map')
   figure,imshow(VQ,[])
   title ('Signum map')
end
figure,imshow(P,[])
title ('Wrapped Phase map recovered by HT')
figure,imshow(S,[])
title ('Unwrapped Phase map recoverd by HT')
figure
surf(S,'FaceColor','interp', 'EdgeColor','none', 'FaceLighting','phong')
view(-30,30), camlight left, axis tight
title('3D surface plot')
xlabel('Pixels'), ylabel('Pixels'), zlabel('Phase in radians')
colorbar
Alternate code for unwrapping
% Unwrapping using the Itoh algorithm
image1_unwrapped = PP2;
for i=1:p
   image1_unwrapped(i,:) = unwrap(image1_unwrapped(i,:));
end
%Then unwrap all the columns one-by-one
for i=1:q
   image1_unwrapped(:,i) = unwrap(image1_unwrapped(:,i));
end
figure, colormap(gray(256)), imagesc(image1_unwrapped)
title('Unwrapped noisy phase image using the Itoh algorithm: the first method')
xlabel('Pixels'), ylabel('Pixels')
lam=0.532;
W=image1_unwrapped; %*4*pi/lambda;
```

APPENDIX–III

MATLAB CODES USED FOR EXPERIMENTAL SETUP AND THOSE RELEVANT FOR STEP VARYING TECHNIQUES IN INTERFEROMETRY

Code in MATLAB for Calibrating the CCD Camera

```
pixel=[524 608 644];
wavel=[508.6 480 467.8];
p=polyfit(pixel,w1,1);
x=100:1:724;
y=p(1)*x+p(2);
plot(x,y,pixel,wavel,'*')
hold on
plot(124,643.8,'*m')
hold off
title('pixel vs wavelength')
xlabel('pixel')
ylabel('wavelength in nm')
grid on
```

Simulating effect of Reflection on Phase Change

```
w1=data(:,1);
n=data(:,2);
k=data(:,3);
N:no.of data base point
for i=1:1:N
    phi(i)=atan2(2.*k(i),1-n(i)^2-k(i)^2);
end

plot(w1,phi)
grid on
k=1;
for i=1:length(w1)
    if w1(i)>=400
    if w1(i)<=750
        w1k(k)=w1(i);
        phi1(k)=phi(i);
        k=k+1;
    else
    end
else
end
end

figure,plot(w1k,phi1)
grid on
xlabel('wavelength')
ylabel('non linear phase change')
title('Simulated plot of total phase')
b=1;
```

```
po=polyfit(w1k,ppp,b);
phs=ppp-po(1).*w1k-po(2);
figure,plot(w1k,phs)
grid on
xlabel('wavelength')
ylabel('non linear phase change')
title('Simulated plot of non linear part')
grid on
```

Step Height Measurement and Non-Linear Phase Change due to Reflection

```
frame=input('enter the name of the frame with path','s');
type='.png';
for i=1:1:8
    a=[frame num2str(i) type];
    j=i;
    b(:,:,j)=double(imread(a));
end

I=b(:,:,:);
clear a b
[p,q,r]=size(I);
num=-I(:,:,1)-5*I(:,:,2)+11*I(:,:,3)+15*I(:,:,4)-15*I(:,:,5)-11*I(:,:,6)+5*I(:,:,7)+I(:,:,8);
den=I(:,:,1)-5*I(:,:,2)-11*I(:,:,3)+15*I(:,:,4)+15*I(:,:,5)-11*I(:,:,6)-5*I(:,:,7)+I(:,:,8);

for vp=1:1:416
    for i=1:1:724
        wp(i)=atan2(num(vp,i),den(vp,i));
    end
    unph=unwrap(wp)';
    pixel=[124 524 608 644];
    w1=[643.8 508.6 480 467.8];
    p=polyfit(pixel,w1,1);
    x=1:1:724;
    y=p(1)*x+p(2);
    sigma=(10^9)./y;
    sgma=sigma';
    c=polyfit(sgma,unph,1);
    unwp=c(1)*sigma+c(2);
    z(vp)=c(1)/(4*pi);
end

vpix=1:1:416;
Zvar=z.*10^9;
figure,plot(vpix,Zvar,'k','linewidth',1)
Zvar1=medfilt1(Zvar,7);
step=Zvar(150)-Zvar(350)
figure,plot(vpix,Zvar1,'m','linewidth',1)
grid on
xlabel('pixel')
ylabel('height z')
title('height variation with vertical pixel')
frame=input('enter the name of the frame with path','s');
type='.png';
for i=1:1:8
    a=[frame num2str(i) type];
    j=i;
    b(:,:,j)=double(imread(a));
end
I=b(:,:,:);
```

```
clear a b
[p,q,r]=size(I);
num=-I(:,:,1)-5*I(:,:,2)+11*I(:,:,3)+15*I(:,:,4)-15*I(:,:,5)-11*I(:,:,6)+5*I(:,:,7)+I(:,:,8);
den=I(:,:,1)-5*I(:,:,2)-11*I(:,:,3)+15*I(:,:,4)+15*I(:,:,5)-11*I(:,:,6)-5*I(:,:,7)+I(:,:,8);
vp=150;
for i=1:1:724
    wp(i)=atan2(num(vp,i),den(vp,i));
end
unwp1=unwrap(wp)';
pixel=[124 524 608 644];
w1=[643.8 508.6 480 467.8];
p=polyfit(pixel,w1,1);
x=1:1:724;
y=p(1)*x+p(2);
sigma=(10^9)./y;
sgma=sigma';
plot(sgma,unwp1,'k','linewidth',1)
p1=polyfit(sgma,unwp1,1);
y1=p1(1)*sgma+p1(2);
frame=input('enter the name of the frame with path','s');
type='.png';
for i=1:1:8
    a=[frame num2str(i) type];
    j=i;
    b(:,:,j)=double(imread(a));
end

I=b(:,:,:);
clear a b
[p,q,r]=size(I);
num=-I(:,:,1)-5*I(:,:,2)+11*I(:,:,3)+15*I(:,:,4)-15*I(:,:,5)-11*I(:,:,6)+5*I(:,:,7)+I(:,:,8);
den=I(:,:,1)-5*I(:,:,2)-11*I(:,:,3)+15*I(:,:,4)+15*I(:,:,5)-11*I(:,:,6)-5*I(:,:,7)+I(:,:,8);
vp=350;
for i=1:1:724
    wp(i)=atan2(num(vp,i),den(vp,i));
end

unwp2=unwrap(wp)';
pixel=[124 524 608 644];
w1=[643.8 508.6 480 467.8];
p=polyfit(pixel,w1,1);
x=1:1:724;
y=p(1)*x+p(2);
sigma=(10^9)./y;
sgma=sigma';
plot(sgma,unwp2,'k','linewidth',1)
p2=polyfit(sgma,unwp2,1);
y2=p2(1)*sgma+p2(2);
hold on
d=unwp1-unwp2;
plot(sgma,d,'k','linewidth',1)
hold off
p11=polyfit(sgma,d,1);
y11=sgma*p11(1)+p11(2);
step1=p11(1)./(4*pi)
plot(sgma,y11)
hold off
grid on
xlabel('wavenumber(1/m)')
ylabel('phase variation(in radian)')
title('Total phase variation in gold')
```

```
d1=d-y11;
figure,plot(sgma,d1)
wv1=10^9./sgma;
figure,plot(wv1,d1)
hold on
order=8;
po=polyfit(wv1,d1,order);
ph=0;
for sr=1:order+1
    nph=po(yt).*(wv1).^(order+1-sr);
    ph=nph+ph;
end

plot(wv1,ph,'k','linewidth',1)
xlabel('wavelength(nm)')
ylabel('phase variation(in radian)')
title('Non linear phase variation for gold')
grid on
hold off
```

Variation of Wrapped and Unwrapped Phase

```
frame=input('enter the name of the frame with path','s');
type='.png';
for i=1:1:8
    a=[frame num2str(i) type];
    j=i;
    b(:,:,j)=double(imread(a));
end

I=b(:,:,:);
clear a b
[p,q,r]=size(I);
num=-I(:,:,1)-5*I(:,:,2)+11*I(:,:,3)+15*I(:,:,4)-15*I(:,:,5)-11*I(:,:,6)+5*I(:,:,7)+I(:,:,8);
den=I(:,:,1)-5*I(:,:,2)-11*I(:,:,3)+15*I(:,:,4)+15*I(:,:,5)-11*I(:,:,6)-5*I(:,:,7)+I(:,:,8);
vp=150;
for i=1:1:724
    wp(i)=atan2(num(vp,i),den(vp,i));
end

pixel=[124 524 608 644];
wavel=[643.8 508.6 480 467.8];
p=polyfit(pixel,wavel,1);
x=1:1:724;
y=p(1)*x+p(2);
sigma=(10^9)./y;
figure,plot(sigma,wp,'k','linewidth',1)
grid on
grid on
xlabel('Wavenumber(in 1/m)')
ylabel('Wrapped phase(in radian)')
title('Wrapped Phase Plot')
unph=unwrap(wp)';
figure,plot(sigma,unwp,'k','linewidth',1)
grid on
xlabel('Wavenumber(in 1/m)')
ylabel('Unwrapped Phase(in radian)')
title('Unwrapped Phase vs Wavenumber')
grid on
```

REFERENCES

[1] Characterization of micro-lenses based on single interferogram analysis using Hilbert Transformation – U. Paul Kumar, N. Krishna Mohan , M.P. Kothiyal

[2] Interferogram analysis using Hilbert Transform – U. Paul Kumar, N. Krishna Mohan and M. P. Kothiyal

[3] Mathematical methods for physicists and engineers – Arfken & Weber

[4] Hilbert Transforms: Volume 1 & Volume 2 (Encyclopedia of Mathematics and its Applications) – Frederick.W.King

[5] Digital Image Processing Using MATLAB (2nd Ed) by Gonzalez, Woods and Eddins

[6] Interferometry by Howard Steel

[7] Two – dimensional phase unwrapping – Theory, Algorithms and Software by Dennis C.Ghiglia & Mark D.Pritt

[8] Speckle Metrology by Rajpal S.Sirohi

[9] Simulation and Experiment in Laser Metrology – International Symposium on Laser Applications in Precision Measurements edited by Zoltan Fuzessy, Werner Juptner and Wolfgang Osten

[10] Optics by Eugene Hecht & A.R.Ganesan

[11] Fundamentals of Optics by Francis Jenkins & Harvey White

[12] Principles of Optics by Max Born & Emil Wolf

[13] Photo Archives of The National Geographic Society

[14] Discrete-Time Signal Processing – Oppenheim & Schafer