Investigation of Interference Patterns in Newton's Rings

Jacopo Siniscalco
Diego Barlettani

January 27, 2019

Abstract

0.1 Introduction

The unification of electricity and magnetism under the cover of a single phenomenon can be considered to be one of the greatest achievements of physics, leading to a theory capable of explaining countless phenomena, previously thought uncorrelated, through a single model of electric and magnetic fields. Most importantly, perhaps, it provided a theoretical framework explaining the origin of the wave properties of light, such as diffraction and interference.

This last effect is perhaps one of the most bizzare features of waves - it allows for two different beams of light to interact with each other, causing the intensity of the superposition of the two waves to vary based on their relative phase.

(a) Constructive interference (b) Destructive interference

Figure 1: Interference of waves can be either constructive (resulting in a larger wave) or destructive (resulting in a smaller wave)

As is shown in Figure 1, waves superimpose by adding their amplitude at every point in space. If two or more waves are in phase with one another - their peaks and troughs match up - the resultant wave will have a larger amplitude. Conversely, the resultant wave will have a smaller amplitude if the input waves are out of phase, as the negative amplitude of one wave will cancel out the positive amplitude of the other when added together.

It is quite common for interesting inteference phenomena to occur in optical systems that split light amongst various paths of different length before recombining the beams. Each ray will have a certain phase shift dependent on the path taken, causing patterns of varying intensity in places where the rays recombine. One such optical system consists of a curved glass lens placed on top of a flat glass plate. Shining collimated light on top of this setup gives rise to an interference pattern composed of a series of concentric rings, alternating in intensity, which is commonly referred to as Newton's rings.

These rings are caused by the light rays passing through the lens and reflecting off the glass plate, interfering with the rays moving downwards. As can be seen in Figure 2, there is a small air gap between the curved surface of the lens and the glass plate. The height of this gap depends on the radial distance from the center. Specifically, if one considers the lens to be a sector of a sphere with radius R, one can obtain a relation between the height of the air gap y and the

distance x from the center as seen in Equation 1.

$$\begin{aligned} R^2 &= (R-y)^2 + x^2 \\ R^2 &= R^2 - 2Ry + y^2 + x^2 \end{aligned} \tag{1}$$

However, for a small lens, we expect the gap y to be orders of magnitude smaller than R (a few wavelengths versus a metre), and the term y^2 can therefore be ignored. Rearranging and simplifying, we obtain Equation 2.

$$2Ry = x^2 \tag{2}$$

Total destructive interference will occur at a point where the returning light beam has a phase shift of π compared to the incident beam on the surface of the lens. Since the returning beam obtains an extra phase shift of π when reflecting off the glass plate (due to the glass having a higher index of refraction than air) [1], the total path length required for a dark band must be an integer number of wavelengths. Since the beam travels twice along a certain path (once going down, and once going up), one requires the air gap y to be a multiple of half the wavelength λ. Therefore, the relation in Equation 3 is obtained between the square of the radius x of the nth dark band and the wavelength.

$$x^2 = \lambda R n \tag{3}$$

This paper aims to investigate the validity of this relationship by recreating the setup and measuring the radii of the rings produced through the use of digital image recognition algorithms and data manipulation to extract cleaner and more precise data from the setup. A monochromatic sodium lamp was used as the light source, and a microscope and smartphone camera were used to collect data from the interference pattern. The images gathered were then processed through OpenCV, an open-source image recognition library for Python/C [2]. The radii obtained from the image data were then used to calculate a theoretical value for the curvature of the lens used, which was then compared to the true value measured with a spherometer.

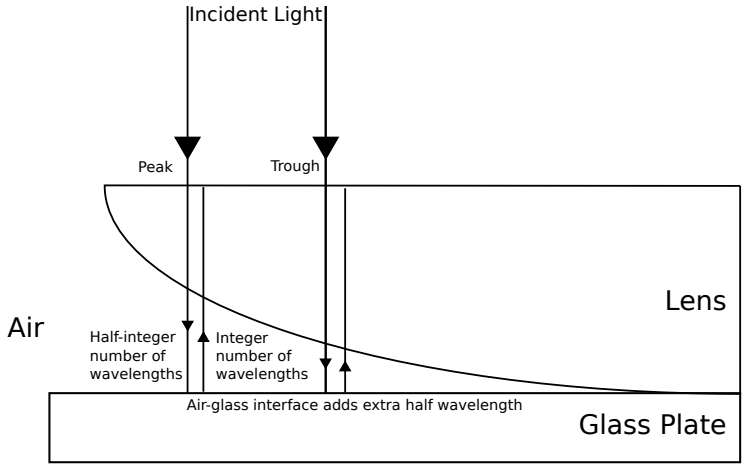

Figure 2: Geometry of the interface between the lens and the glass

0.2 Method

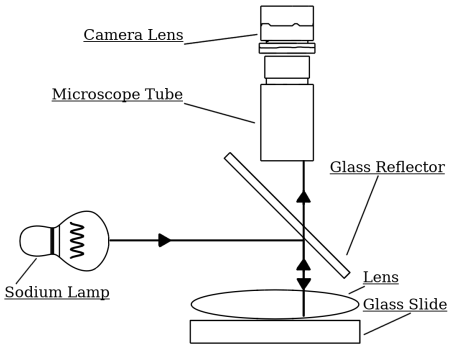

Figure 3: Schematic diagram of experimental setup

0.2.1 Experimental Method

To obtain the required interface between materials of differing refraction, a small convex lens was placed on top of a glass slide. This setup was then placed underneath another oblique glass slide to act as a half-mirror reflector, redirecting light towards the lens but also allowing for observation of the setup from above. A monochromatic sodium lamp (with wavelength of 589.3 nm) was used to direct light of a particular wavelength towards the setup, as seen in Figure 1. The sodium lamp required some time to reach maximum intensity as it heated up, and was therefore turned on as early as possible.
Because of the small size of the interference pattern expected from the setup, magnification was required to observe the result. A low-magnification microscope was therefore used to observe the lens from exactly above. To simplify the act of finding the pattern, which only occurs near the center of the lens, it is reccomended to use the microscope to first find the edge of the lens, using these region both as reference and to focus the lens to the right distance. Once the edge is found, the microscope can be slowly moved radially inwards, until the ringed interference pattern is found.

Once the microscope field is centered on the interference pattern, final adjustments to the focus of the instrument can be carried out, to obtain a clear image. Following this, the image can be captured. While ideally the data would be recorded directly by using a camera eyepiece, it is possible to also photograph the field of view manually, by holding a digital or smartphone camera over the microscope eyepiece. It was found that clearer images were obtained by steadying the camera a few millimeters above the eyepiece, without letting the two lens make contact.

After the images of the pattern had been aquired, manual measurements of

some distances had to be taken to calibrate the pixel distances to real-world distance. Originally the radius of the innermost dark band was measured for these purposes, but it was found that more accurate measurements were possible by instead measuring the difference between the radii of the first and second ring, as these boundaries were better defined compared to the more "fuzzy" inner ring. The manual measurements were taken through the use of a vernier scale attached to the microscope.

A final step required measuring the true curvature of the lens, which was used to gauge the accuracy of the measurements taken. The radius was measured through the use of a spherometer, which measured the height difference of a portion of the lens versus a flat surface. These measurements provided the height and length of a sector of the circle traced by the lens, from which the radius of the circle can be extrapolated.

0.2.2 Computational Method

Several images were gathered from a single observation, and were manually inspected to select one with good clarity and resolution, and minimal noise. The selected image was then manually edited, cropping it to only include the field of view of the microscope, and increasing the image contrast. The image was also scaled to a lower resolution, to improve computational efficiency at the cost of a minor decrease in precision.

The image was then fed to the image recognition algorithm (Appendix A), that converted the 16-bit rgb channels to a single 8-bit intensity value for each pixel. A gaussian blur was then applied by convolving the pixel values with a gaussian matrix. This step was applied to reduce high-frequency noise in the image, which is very common for pictures taken with small cameras and short exposures.

A Hough transform, provided by the OpenCV library, was then applied to the processed image to detect any circles. This algorithm first attempts to find edges in the image by locating pixels with an intensity above a certain threshold, and then creating paths extending from these points to all other pixels brighter than a second, lower threshold. These edges are stored in a buffer, and a separate algorithm attempts to fit a collection of points onto a parametric equation of a circle [2].

The detected circles were then filtered by distance to the center of the image, and the center of one of the detected circles was chosen to be the center of the concentric rings in the interference pattern. Initially, pixel intensities were plotted from this center to the edge of the image, but this approach was prone to low-frequency noise, such as dark spots caused by dirt on the lens of the microscope. To lessen the effect of noise, a polar transform was applied to the image, centered around the detected circle. Under such a transformation, any concentric circles would be visible as vertical bars instead. The intensities of each column of pixels were then calculated and plotted, leading to a much more robust estimate of ring radii, as can be seen in Figure 4 (the top row shows the image before the polar transform, and the bottom is after - note the much

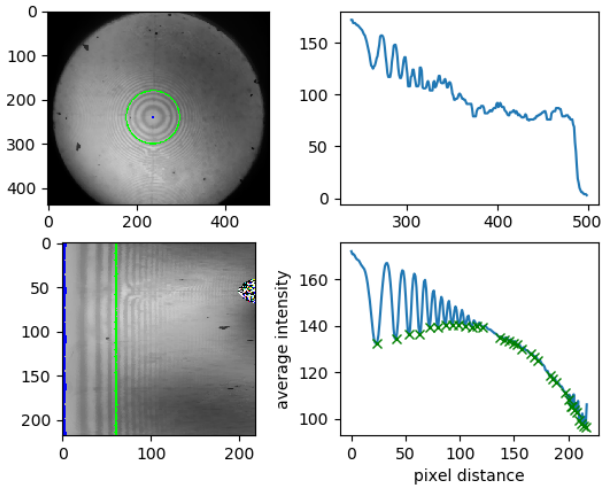

Figure 4: Output of image recognition algorithm

greater regularity in the intensity pattern). The polar transform also makes it easier to determine the success of the circle detection algorithm, as a misaligned center will produce oscillating, rather than straight, vertical lines, which are easy to notice visually.

Local minima were then found in the data using SciPy, an open-source Python library for scientific computing [3]. The radius of each minimum was recorded, and a section of these radii corresponding to clean oscillations in the graph were manually selected. The difference between the first and second value were then used to obtain a pixel-to-mm proportion by comparing them to the measured calibration value, and all pixel distances were multiplied by this factor to convert them to real-world distances.

0.3 Results

As seen in Equation 3, one would expect the square of the radii of each ring to be linearly dependent on the index of the ring. Figure 5 portrays a graph of the square of the real-world radius of the minima obtained from the circle detection algorithm versus their index. As can be seen, the data has a very good linear correlation, with minimal variance arising from the fit to a line. However, the greatest source of error comes from the calibration itself, and multiplies all the data points equally - therefore, this error is not evident on the actual graph, and in fact is considerably greater than the variance calculated.

Using the relation described in Equation 3, one can easily find an estimate for the radius of curvature, yielding a value of 553 ± 3 mm without consideration of the larger error source.

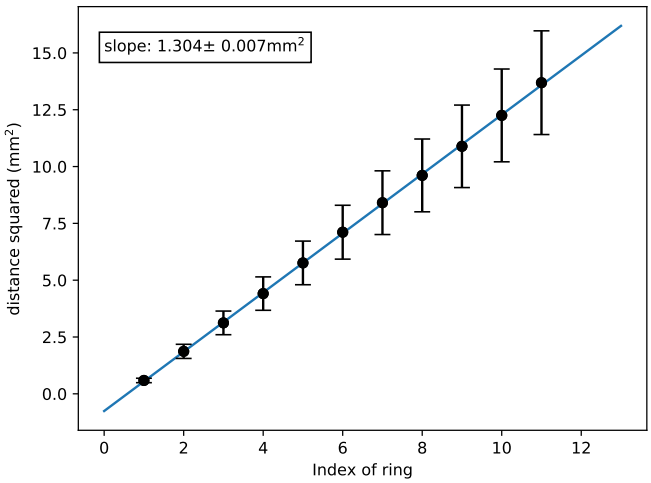

Figure 5: Index of ring versus radius squared

The precision of the calibration measurements, in theory, are dictated by the vernier scale's precision of ±0.07 mm (the precision for a single value would be ±0.05 mm, but this error occurs at both ends of the measurement, and must be added in quadrature) There may, however, be some amount of unquantified error due to the lack of proper definition for the bands. The slope of the line can be converted from mm^2 to $pixel^2$ by dividing by the square of the mm/pixel ratio found during calibration (0.033mm/pixel ± 0.003). The square of the ratio can then be multiplied back, this time applying the error analysis (Equation 4), to obtain an error on the slope due to the calibration of ±0.2 mm^2. The final estimate for curvature was therefore 550 ± 90 mm.

$$\begin{aligned} \text{slope in mm}^2 &= (\text{slope in pixels}^2)(\text{mm/pixel})^2 \\ \text{error in slope} &= 2(\text{mm/pixel})(\text{error per pixel}) \end{aligned} \tag{4}$$

The curvature of the lens was then measured manually, with a spherometer. The distance d between a leg and the sagitta (central screw) of the spherometer was measured using a ruler to be 15 ± 0.7 mm. The relative height h of the lens was found to be 0.24 ± 0.005 mm. From this data, the radius of curvature R can be calculated according to Equation 5 [4].

$$R = \frac{d^2}{2h} + \frac{h}{2} \tag{5}$$

The measured radius of curvature was 470±40 mm. Errors were propagated, as usual, by summing the product of error in the variables and the partial derivatives of each variable in quadrature (Equation 6) [5].

$$\Delta F(x_1, ..., x_n) = \sqrt{\sum_i (\Delta x_i \frac{\partial F(x_1, ..., x_n)}{\partial x_i})^2} \tag{6}$$

There is a significant area of overlap between the two values, suggesting the approach used to model the phenomenon is valid, and also validating the computerised approach to image analysis that was employed.

0.4 Analysis

The calculated and measured values for radii of curvature overlap, implying validity of the theoretical treatment of Newton's rings. However, both values had considerable errors, and improvements on procedures may be applied to greatly increase precision. The greatest source of error in the theoretical measurement come from the calbiration step, resulting in a value an order of magnitude greater than what would be expected from the linear regression alone (± 90 mm versus only ± 3 mm). This discrepancy is to be expected, as the precision of the computer-generated data is obviously considerably superior to what can be achieved from the manual measurements done during the calibration steps, especially with the addition of manipulations aiming to reduce data noise.

An obvious solution would be to use a digital microscope, as that would allow pixel measurements to be automatically calibrated. However, such equipment may be expensive and complex, so alternatives have been proposed, such as laying a small marker of known size on the glass plate, under the lens. Such a marker may be as simple as a sliver of paper, or a frame around the center of the lens. The marker could easily be picked up by image recognition and its pixel length correlated to its real length, allowing for automatic calibration of the algorithm. Depending on the system used to fabricate or measure the marker, the precision of the calibration step may also be greatly improved and made more reliable.

Alternatively, several calibration measurements may be taken, and the calculated conversion values averaged to increase precision of the final value. However, the use of image recognition in this investigation aimed specifically to reduce the number of readings taken by a human, to increase both precision and efficiency, and such a procedure would be against the spirit of the computerised approach.

While fairly accurate, the data obtained from the image itself may also be improved. The resolution of the image on which the algorithm was applied had to be downscaled to allow for responsive computational times (on the order of a few seconds), but higher resolutions could be used to allow for more viable datapoints. The main factor limiting data extraction, however, was the quality of the image itself. The pictures were taken using a smartphone camera that was stabilised by hand on the microscope eyepiece, leading to suboptimal quality. The low exposure time led to a darker and less defined image, especially further from the focal point, which in part explains why the rings further from the center were harder to detect. Furthermore, dirt and distortions on the lens are visible, and very hard to completely filter out digitally. These effects, combined with blurring and "splotching" caused by imperfect stabilization, eventually overwhelm the regular data patterns that indicate the presence of a ring. Simply improving the quality of the camera used, and employing a mechanical (rather

than biological) stabilization system would contribute to improve data quality.

0.5 Conclusion

0.6 Appendix A - Image recognition algorithm

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from scipy.signal import argrelmin
import sys

#read input image, resize it if needed, and apply a basic gaussian blur to remove
#high-frequency noise

img = cv2.imread(sys.argv[1],0)
img = cv2.resize(img,(0,0), fx=1,fy=1)
img = cv2.medianBlur(img,5)

#rescale color space from rbg to bw
cimg = cv2.cvtColor(img,cv2.COLOR_GRAY2BGR)
bwimg = cv2.cvtColor(img,cv2.COLOR_GRAY2BGR)

#apply the hough transform to obtain detected circles in the image
#circles should be an array with each element of the form [xpos, ypos, radius]

circles = cv2.HoughCircles(img,cv2.HOUGH_GRADIENT,5,50,param1=110, param2=80,minRadius=40,maxRadius=0)
circles = np.uint16(np.around(circles))

#obtain the center of the image for size calculations
center = np.array(np.shape(img)) / 2.0
#print(center)

# iterate through all circles found, and draw those within a certain radius of the center of the image
# to filter out the stuff we don't want
# also choose one of these circles to be the one we use, and save it's center point
#-- This is a bit of a bodge, as it simply
# pulls a random circle - in theory, this is the only one that should be found,
#but this isn't always certain depending on noise

trueCenter = None
for i in circles[0]:
    if((np.abs(i[1] - center[0]) ** 2 + np.abs(i[0] - center[1]) ** 2) ** 0.5 < 35 ):
        trueCenter = i
        # draw the outer circle
        cv2.circle(cimg,(i[0],i[1]),i[2],(0,255,0),2)
        # draw the center of the circle
        cv2.circle(cimg,(i[0],i[1]),2,(0,0,255),3)



#transform to polar and draw image - do this twice, once for the image to display,
```

```
#and once for the one to do the math on (which should not have debug graphics -
#eg the center of the circle drawn)
size = int(np.shape(img)[0]/2)
dst = cv2.warpPolar(cimg,(size,size), (trueCenter[0],trueCenter[1]), maxRadius=size, flags = cv2.WARP_POLAR_LINEAR)
dst2 = cv2.warpPolar(bwimg,(size,size), (trueCenter[0],trueCenter[1]), maxRadius=size, flags = cv2.WARP_POLAR_LINEAR)

#plot intensities and figures
fig = plt.figure()

plt.subplot(2, 2, 1)
plt.imshow(cimg)

plt.subplot(2, 2, 2)
#plot pixel intensities from center to the side of the image (bad way)
plt.plot(range(0,len(bwimg[trueCenter[1]]))[trueCenter[0]:],[i[1] for i in bwimg[trueCenter[1]]][trueCenter[0]:])


plt.subplot(2, 2, 3)
plt.imshow(dst)

plt.subplot(2, 2, 4)
#Find the average value of each column in the transformed image,
#and plot that value instead (better way)
plt.xlabel("pixel distance")
plt.ylabel("average intensity")

average_val = [sum([dst2[j][i][1] for j in range(0,len(dst[i]))]) / len(dst[i]) for i in range(0, len(dst))]
plt.plot(range(0,len(dst)), average_val)

#find local minima, and plot those on the graph. also print them (pixel distances)
mins = argrelmin(np.array(average_val), order=1)
plt.plot(mins[0][1:],[average_val[i] for i in mins[0][1:]], 'gx')


[print(i) for i in mins[0][1:12]]

plt.show()
```

Bibliography

[1] Yu, N., Genevet, P., Kats, M.A., Aieta, F., Tetienne, J., Capasso, F., Gaburro, Z. (2011). *Light propagation with phase discontinuities: generalized laws of reflection and refraction.* Science, 334 6054, 333-7.

[2] Bradski, G. (2000). *The OpenCV Library.* Dr. Dobb's Journal of Software Tools, 2236121

[3] Jones E, Oliphant E, Peterson P, et al. *SciPy: Open Source Scientific Tools for Python*, 2001-, http://www.scipy.org/ [Online; accessed 2019-01-27]

[4] Khan, S.A. (2013) *Coordinate Geometric Approach to Spherometer*,8 Sep 2013, arXiv:1309.1951v1 [physics.gen-ph]

[5] Yusaf, F. (2011). *First Year Laboratory Physics.* London: King's College London