

MRES SUMMER PROJECT

Topographic Inference Engine for Interferometric Microscopy with Implications for Structural Colour

Author:

Mark Ransley

Supervisors:

Dr. Geraint Thomas

Dr. Phil Jones

UCL CoMPLEX

August 2013



“Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.”

Brian W. Kernighan and P. J. Plauger
The Elements of Programming Style

Abstract

Laser interferometry has long been an important technique in metrology, providing excellent sensitivity to longitudinal displacements albeit with limitations in the lateral axes. After reviewing the current and proposed applications of this concept to biological problems and practice, we consider the interpretation of interferograms (the output from an interferometric system), and conditions under which a detailed profile of a biological surface may or may not be extracted from one.

We present an algorithm that combines elements of raytracing with interference theory to generate interferograms from simulated surfaces and optics systems. Through optimising the surface parameters we are able to ascertain the geometries responsible for a given interferogram. Examining the output of a modified Michelson interferometer, we observe that the effect of diffraction on interferograms is larger than previously assumed, and investigate techniques for modelling this phenomenon within our algorithm.

Considering additional uses of the software, a chapter is dedicated to structural colour - where nano-structures give rise to interference effects seen in nature - including the techniques currently used when rendering such systems and the potential value this field may hold for technology.

To test whether our algorithm is capable of rendering structural colour, an angular spectral analysis is conducted on the iridescent tail feather of the male Indian peafowl, and the nano-structures believed to be responsible are modelled in the software.

Acknowledgements

Special thanks to my supervisors Dr. Phil Jones and Dr. Geraint Thomas for providing the initial stimulus for this project, as well as for the optics equipment and of course their time. Thanks also to Dr. Lewis Griffin for pointing me towards structural colour, answering a great number of my questions and lending me the spectroradiometer, to Mark Turmaine for an enlightening discussion on the current standard of Scanning Electron Microscopy, to Dr. Guy Moss for allowing me to use his microscope (sorry about the bulb), and to Dr. Bart Hoogenboom for his honesty and critique.

Finally I would like to thank the old man outside Kings Cross station for providing the peacock feathers, a true testament to the notion that London is a place where anything and everything can be found little more than a tube stop away.

Contents

Abstract	ii
Acknowledgements	iii
Nomenclature	vi
1 Interferometric microscopy	1
1.1 Classical interferometric microscopy	1
1.1.1 The Mach-Zehnder interferometer	2
1.1.2 Jamin-Lebedeff interference microscopy	3
1.1.3 Phase stepping algorithms	3
1.2 Single beam interference microscopy	4
1.2.1 Phase contrast	4
1.2.2 Differential interference contrast	5
1.2.3 Synthetic aperture 3D DIC	5
2 The parallel problem	8
2.1 CGI rendering techniques	9
2.1.1 Ray tracing	9
2.2 Interferometric ray tracing	10
2.2.1 Implementation	11
2.3 Optimisation	13
2.4 Diffraction	14
2.4.1 Raytracing diffraction	15
2.5 Conclusions	15
3 Rendering structural colour	16
3.1 Types of structural colour	16
3.2 Quantifying structural colour	17
3.3 Rendering structural colour	17
3.4 Structural colour in technology	18
3.5 Interferometric raytracing of structural colour	18
3.5.1 Methods	19
3.6 Modelling the colour structure	20
3.7 Conclusions	22

A Matlab Code**28**

Nomenclature

OPL	Optical path length
BS	Beam splitter
DIC	Differential image contrast
OCT	Optical coherence tomography
NA	Numerical aperture
ISAM	Interferometric synthetic aperture microscopy
CCD	Charge coupled device
SLM	Spatial light modulator
λ	Wavelength
ϕ	Phase
n_s	Refractive index (of sample s)
i, j, k, m	subscripts
i	$\sqrt{-1}$
v	Normalised vector
n	Surface normal vector
θ_i	Angle of incidence
θ_r	Angle of reflection
θ_o	Angle of observation

Chapter 1

Interferometric microscopy

In previous work ([Ransley \[2013\]](#), [Wyatt \[2012\]](#)) the prospect of a new class of 3D super-resolution microscope based on laser interferometry was discussed, and the theory of this instrument shall be developed further in Chapter 2. In said reports the techniques behind super-resolution microscopy and sub-nanometre sensitive interferometry were reviewed, and the demand for our proposed technique surveyed.

This introductory chapter will examine the applications of interferometry to biological research, the strengths and shortcomings of each technique, and the compatibility with the method we are developing. Even at diffraction limited resolution interference microscopy is a vast field, often preferable over conventional bright-field optical methods due to its enhanced contrast without the need to resort to dyes.

1.1 Classical interferometric microscopy

This refers to the set of instruments where the beam is split into two “arms”; one for reference and one for interacting with a sample in some way, accumulating a change of phase across its lateral axes. Once recombined, the transverse cross section of the beam exhibits variations in brightness due to the phase differences, themselves a result of the differing optical path lengths (OPL).

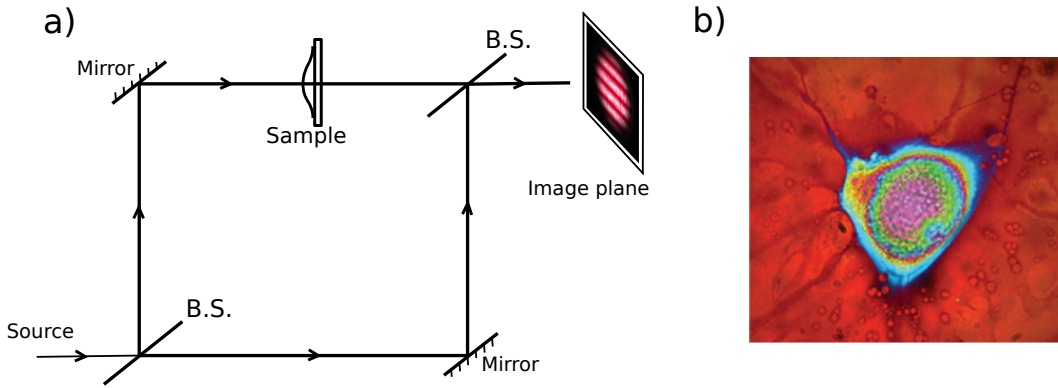


FIGURE 1.1: a) A Mach-Zehnder configuration. b) Mach-Zehnder interferogram of a snail neurone. Colour is indicative of changes to longitudinal depth or refractive index, though the correspondence is not one-to-one. Thus interpretation of the image necessitates use of a phase-stepping algorithm. b) taken from [Kaul et al. \[2010\]](#)

1.1.1 The Mach-Zehnder interferometer

Possibly the simplest of interferometer configurations, the Mach-Zehnder device passes the measurement arm through a semi-transparent sample with refractive index n_s (see fig. 1.1.a). The accumulated change in path length is

$$\Delta OPL = n_s d$$

where d is the thickness of the sample. The setup was developed in the 19th Century ([Zehnder \[1891\]](#), [Mach \[1892\]](#)) and is still used in research today, especially in cellular biology where the structures of interest are usually transparent. Recently for example, in [Mahlmann et al. \[2008\]](#) the dry weight of single living cells was inferred by taking a Mach-Zehnder reading and formulating the OPD distribution in terms of its component refractive indices. In [Kaul et al. \[2010\]](#) the configuration revealed that the *Lymnaea stagnalis* neurone exhibited a change of refractive index when stimulated electronically.

As with all classical interferometric instruments, Mach-Zehnder devices are extremely sensitive to mechanical vibrations, though such effects are minimised when the system is self contained on a chip ([Nolte \[2012\]](#)). Care must be taken to ensure the beams are recombined without any tilts or baseline displacements to ensure there are no background fringes, as seen in fig. 1.1.a.

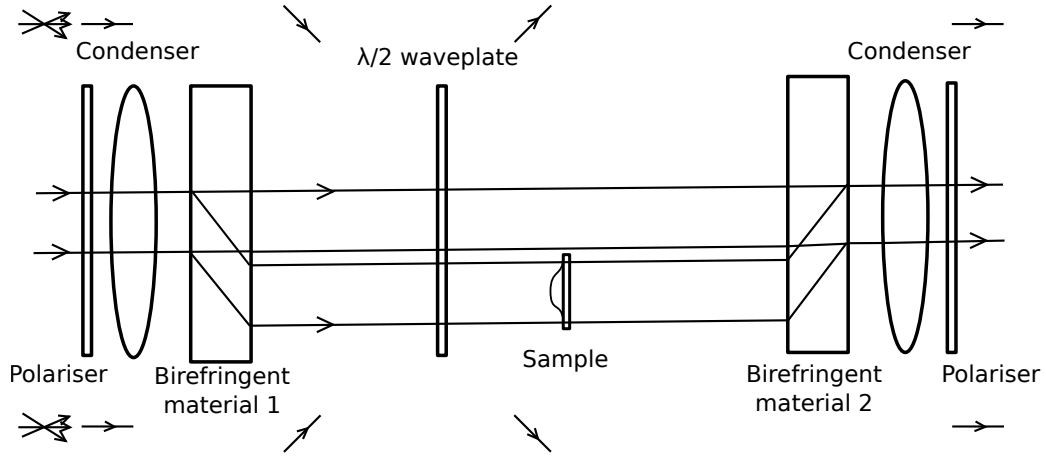


FIGURE 1.2: Jamin-Lebedeff interferometer. Polarisation is indicated above the reference beam and below the sample beam.

1.1.2 Jamin-Lebedeff interference microscopy

The Jamin-Lebedeff interferometer reduces the effect of mechanical vibrations through eliminating the need for mirrors. It is also often preferable due to the parallel optical paths. The beam is split via a birefringent material, where the refractive index depends on the polarisation of the incident light. The linearly polarised light upon entry is split into its orthogonally polarised *ordinary* and *extraordinary* components, the second of which is refracted to form the measurement arm. The $\lambda/2$ wave-plate induces a phase shift of π but also swaps the polarisations such that “birefringent material 2” (see fig. 1.2) inverts the splitting of “birefringent material 1”.

Recently in [Leertouwer et al. \[2011\]](#) this instrument was used to determine the Cauchy coefficients of unpigmented butterfly chitin and bird-feather keratin. Refractive index is wavelength dependent, and the relationship $n(\lambda)$ is parameterised using the substance’s Cauchy coefficients.

1.1.3 Phase stepping algorithms

In [Ransley \[2013\]](#) we discussed what was termed the 2π ambiguity, where OPL changes separated by integer multiples of λ are indistinguishable in the interferogram. This is pronounced in fig. 1.1.b, where the phase reading is repeated with increasing depth. Previously we suggested using multiple wavelengths to remedy this, and have since encountered this idea being put into practice ([Manhart et al. \[1990\]](#)).

However a more sophisticated method of extracting ΔOPL from interferometer output is that of phase stepping, where additional phase modulations are introduced between the reference and measurement beams, for which algorithms exist to determine the distance (Stoilov and Dragostinov [1997], Kinnstaetter et al. [1988], Creath [1988]). This can be accomplished through diversion of the reference arm, introduction of refractive materials, axial translation of the sample or through the use of an electro-optic modulator.

1.2 Single beam interference microscopy

Single beam methods are used to provide enhanced contrast over traditional optical microscopy, and create the appearance of a 3D image. However, this is illusory due to the nonlinear relationship between OPL and phase.

1.2.1 Phase contrast

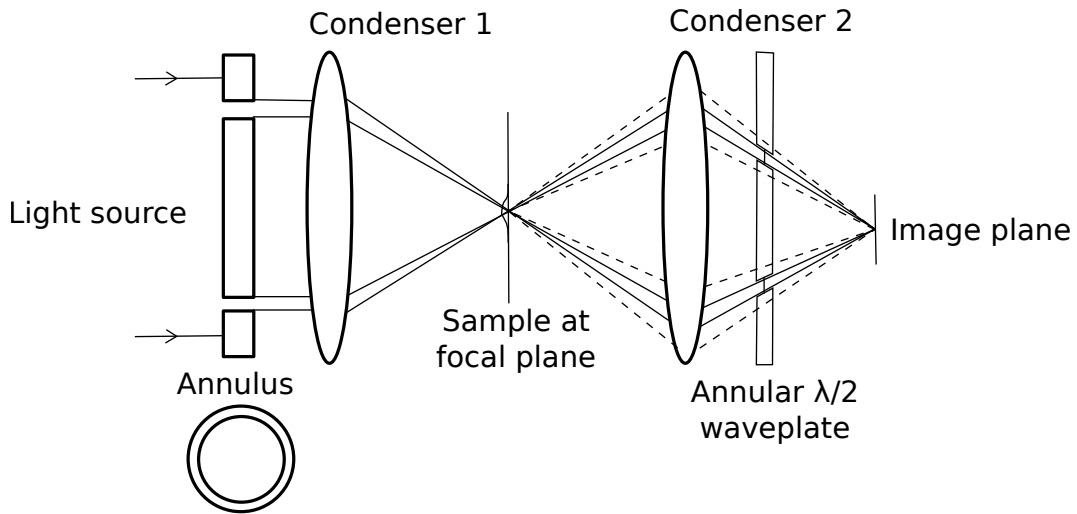


FIGURE 1.3: The phase contrast microscope. Dashed lines represent cones of light scattered or diffracted by the sample.

Phase contrast microscopy (Zernike [1955]) passes collimated light through an annulus, such that the light becomes a hollow cylinder that is then focused by a condenser to a point in the sample plane. The emerging light consists of the original source that passed through unhindered and an additional field of light diffracted or scattered by objects in the sample. A second condenser focuses this combination onto an image plane, whilst an annular $\lambda/2$ wave plate induces a phase shift *only* in the unscattered field, to increase

the contrast. Subsequent interference between the scattered and unscattered beams at the image plane produces the resulting image.

Phase contrast is equally effective in x-ray microscopy, where the lower wavelength allows features as small as $0.16\mu\text{m}$ to be resolved, though this technique is known to be harmful to living specimens. See [Davis et al. \[1995\]](#) and [Momose et al. \[1996\]](#) for further details.

1.2.2 Differential interference contrast

DIC is effectively a hybrid of phase contrast and Jamin-Lebedeff imaging. The two arms of the latter instrument are separated by a small amount Δx such that there is a significant overlap between them, and both pass through the sample. The resulting image upon recombination is a map of the *phase gradient* with respect to Δx , and tilting or rotation of the birefringent (for DIC a Wollaston prism is used) allows for the degree and orientation of shear to be varied. The depth of focus for the two beams can also be controlled through objective lenses. These variables must be carefully selected in order to discriminate features of interest.

DIC has been applied to confocal ([Cogswell and Sheppard \[1992\]](#)) and x-ray ([Wilhein et al. \[2001\]](#)) microscopy systems to produce images of enhanced contrast and resolution.

1.2.3 Synthetic aperture 3D DIC

Synthetic aperture is a technique previously used in radar and astronomy, whereby readings taken from multiple locations can be combined digitally to effectively increase the Numerical Aperture. Since microscopy is hindered by Abbe's diffraction limit, meaning features smaller than $\lambda/2NA$ cannot be resolved, synthetic aperture imaging enables an increase in microscopic resolution. In [Dasari et al. \[2013\]](#) a Mach-Zehnder type instrument was used with phase stepping to calculate $\phi(x, y)$, and a reading without reference beam taken as the amplitude $A(x, y)$, thus providing the complex field image

$$E(x, y) = A(x, y)e^{i\phi(x, y)}$$

Fourier transforming the complex field image under oblique illumination, a peak was observed in the centre of the Fourier plane, representing unscattered light (denoted by

the yellow spot in fig. 1.4.b). The diffraction limit in Fourier space is manifest as a disk of radius NA/λ , with frequency components lying only within this disk (shown as a red dashed ring in fig. 1.4.b).

In Dasari et al. [2013] the complex field range was computed for 360 hemispherical polar angles of illumination θ (fig. 1.4.a), which took around 5 minutes to capture. For each value of θ the Fourier peak was observed to be offset accordingly (fig. 1.4.b.ii), which was explained by modelling the transmitted field at the back focal plane. Translating each Fourier representation across k -space to remove this offset (fig. 1.4b.iii) the set of Fourier components was summed across θ to form the field with synthetic aperture, denoted by the black ring in fig. 1.4.b.iv).

This was found to be 1.8 times larger than the natural aperture, and a corresponding increase in sample frequency was observed in images synthesised from the synthetic

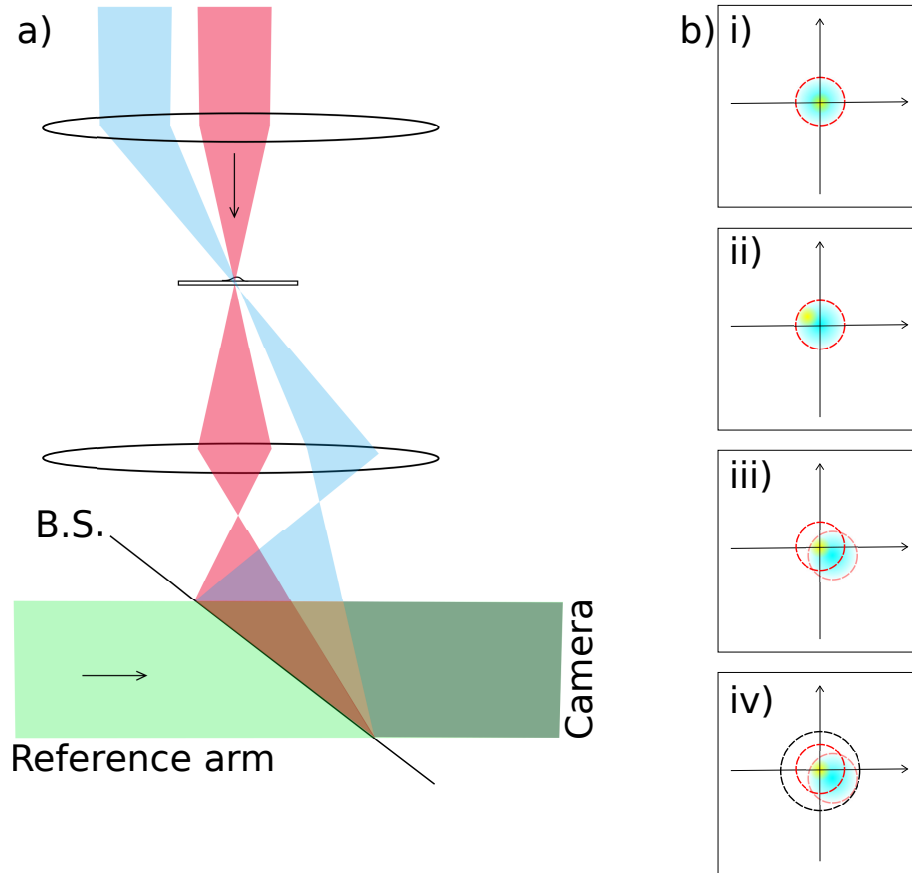


FIGURE 1.4: a) measurement section of 3D synthetic aperture interferometric microscope. Red represents oblique illumination of the sample whilst blue represents illumination at angle θ . Note that the lenses are positioned such that the reference and measurement arms recombine regardless of the value of θ . b) Fourier transforms under oblique and angular illumination.

aperture field. The group applied this technique to imaging live He-La cels, and achieved not only higher clarity images but an 88% reduction in background phase noise.

Using this method DIC images were synthesised *a posteriori*, so that selective shearing was able to reveal previously unseen cellular features.

Granted the 5 minute capture time for this technique renders it currently unsuitable for studying the dynamical processes of interest to biologists, however the 4-step phase shifting process may be able to be eliminated through use of a dual-wavelength sample beam, as described in [Ransley \[2013\]](#). Perhaps a smaller angular measurement domain would increase capture speed without putting too great a compromise on synthetic field sample frequency. It is possible that the increased axial resolution of synthetic aperture interferometry combined with the 50pm axial sensitivity of the synthetic wavelength method described in [Chen et al. \[2002\]](#) could produce a super-resolution instrument for the detailed and non-invasive study of living biological structures.

Chapter 2

The parallel problem

For both the classical interference microscopy examples considered in Chapter 1, it is assumed that, neglecting phase stepping and identifiability issues surrounding $\Delta OPL = dn_s$, there is a one to one mapping between the sample and its interferogram. This is not the case in situations such as the one shown in fig. 2.1, where a reflective object manipulates some of the incoming light such that it contributes to the image plane phase at a point other than the one directly above where it started. The interferogram would imply that below point * there is an elevation or object of higher refractive index than its host medium. Similar scenarios could arise when using interferometry for surface profiling, particularly when the subject is illuminated with a well directed laser beam.

As discussed in [Ransley \[2013\]](#), extracting the sample structure from the interferogram in such cases could potentially be achieved through solving an inverse problem, where the optics system and sample are modelled and the sample is optimised to match the

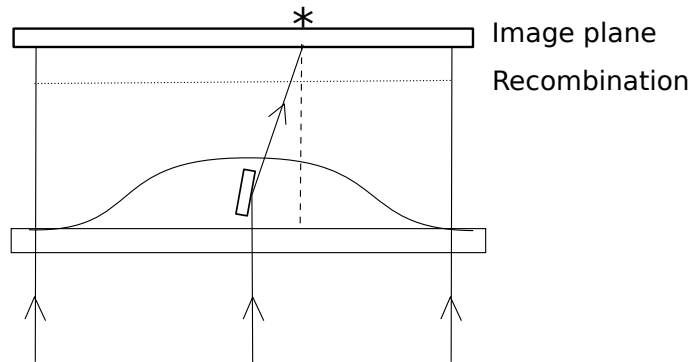


FIGURE 2.1: Objects within a sample can lead to issues when interpreting the interferogram.

simulated and observed instrument outputs. Issues of identifiability are raised by the OPL and phase ambiguities, but could be remedied through phase stepping and the application of reasonable constraints. Constraints could include restricting the refractive indices and distances to known values, and capturing/simulating interferograms from several angles. In the latter case the modification of one parameter during the optimisation could cause multiple changes to the output, informing the optimiser of the best direction in which to proceed.

In the previous report we simulated interference using Gaussian beam models, and found the method to be adequate for simple reflections from flat surfaces under displacements and tilts. However, for modelling complex surfaces or light propagation through biological matter, solutions to Maxwell's equations prove very computationally intensive. Here we offer a simpler and faster approach.

Through the remainder of this chapter we apply ray-theoretic techniques to interferometric simulations, with the aim of solving the described inverse problem. Since interference is purely a wave phenomenon, novel methods are proposed.

2.1 CGI rendering techniques

In the field of computer graphics, rendering algorithms are used to form a 2D representation of a 3D environment, hereon referred to as a scene. Fermat's principle states that a ray of light travels between two points in the least time possible, and so the majority of rendering calculations are basic linear algebra.

2.1.1 Ray tracing

Forward ray tracing - the process of simulating a light source and its propagation through the entire scene - is often computationally wasteful, since generally only a small fraction of that light is received by the camera. More commonplace is backward ray-tracing, where rays are instead fired from the camera and traced through the scene back to their origins, after which it is possible to compute the brightness and colour across the camera's pixels.

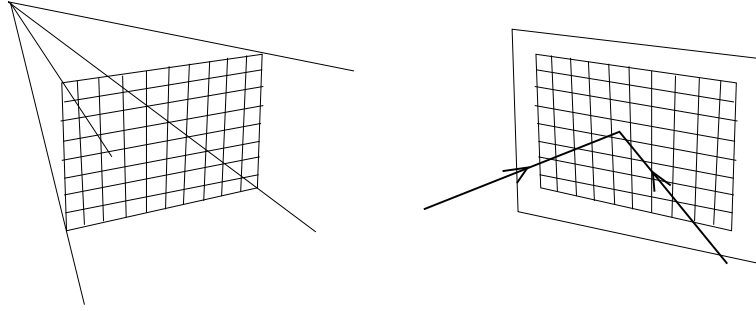


FIGURE 2.2: Perspective based rendering (left) compared with an optical CCD (right).

In the case of laser optics however, it can be assumed that the scene is set up in such a way as to direct a substantial proportion of the light to the camera, or charge coupled device (CCD). Additionally, conventional rendering pipelines aim to emulate perspective, requiring that the rays shot from each camera pixel have a predetermined starting direction. This differs from interferometry, where light incident on the CCD could come from any direction on any pixel (see fig. 2.2). For these reasons we adopt the forward ray tracing approach, tracing the light paths from the source.

2.2 Interferometric ray tracing

The interferometric ray tracing engine takes as input a set of objects that comprise the scene. Before objects are added, the measurement and reference beam sources are set up symmetrically to emulate a beam splitter. Reference mirrors and the CCD are placed such that in the absence of objects, the reference and measurement beams converge across the CCD. In the case where refraction is not permitted, and objects reflect all incoming light, the ray paths are computed as follows:

- The starting coordinates for the rays are distributed over the source. The CCD is similarly divided up into the given number of pixels.
- Each ray is checked for intersection with each object and each pixel of the CCD. If an object is hit, that section of the ray terminates and a new one is formed at the point of intersection, with direction vector given by the reflection formula.
- If a CCD pixel is hit, the ray terminates there without forming a reflection.
- If a ray leaves the boundaries of the scene, it is terminated.

- The ray tracing ends when all rays have reached a CCD pixel, left the scene or exceeded their maximum number of reflections.
- Each pixel of the CCD contains a record of the rays it terminated, their OPLs, and the number of reflections accrued on the way.
- Each ray's phase is calculated as a function of its wavelength, OPL and the number of reflections it underwent.
- For each pixel, the field equations for its incident rays are calculated and summed, and the absolute value taken to give the pixel's intensity.

The CCD image is given as output.

2.2.1 Implementation

As a proof of concept, the above algorithm was implemented in Matlab for converting 2D scenes into 1D CCD output. Objects in the scene are given as a set of lines, described by their start and end coordinates. The set of rays is described by the *ray-matrix*, a 3D array. Each 2D cross section of the ray-matrix describes the set under successive reflections, and is structured as:

Ray#	Normal-vector	Gradient	Start	End	Hits object	y-axis intercept
⋮	⋮	⋮	⋮	⋮	⋮	⋮

For much of the algebra used it is simpler to parameterise the rays by their gradient and y-axis intercept, however at times the normalised vector is necessary since the former representation does not account for the rays' directions.

The point where ray i intersects object j is given by

$$x_{int} = \frac{c_j - c_i}{m_i - m_j}$$

$$y_{int} = m_i x_{int} + c_i$$

where m are the gradients and c are the y-axis intercepts.

It is then checked whether the intersection occurs within the start and end points of ray i and object j . If the ray hits multiple objects, the intersection closest to the ray's starting point is taken. The initial end coordinates of a ray are given as the point where it crosses the scene boundaries, though these are overwritten if it successfully hits an object.

The normalised vector of the reflected ray is given by the vectorised form of Snell's law:

$$\mathbf{v}_{i,k+1} = -2(\mathbf{n}_j \cdot \mathbf{v}_{i,k})\mathbf{n}_j + \mathbf{v}_{i,k}$$

where \mathbf{n} is the object's unit normal vector. It should be noted that subscript k denotes position in the third dimension of the ray-matrix. From the normal vector, the gradient, y-intercept and initial end coordinates of the reflected ray are calculated.

If the object intersected is the CCD, the pixel hit is determined and the ray's subscripts are added to that pixel's row of *CCDrays*, a structure used in calculating the final intensities.

The above process is iterated across the rays through to the maximum number of reflections. If a ray starts outside the scene boundaries or on the CCD, intersection checking is skipped and its end coordinates are taken as equal to its starting ones.

For CCD pixel m , the OPLs of its incident rays are found by summing the rays' lengths over their reflected components. The phase of ray i at the CCD is

$$\phi_i = \frac{2\pi(OPL_i \bmod \lambda)}{\lambda}$$

The intensity of CCD pixel m is then

$$I(m) = \left| \sum_{i=1:\#rays} A(i) \exp(i(\phi_i + k\pi)) \right|$$

where $A(i)$ is the amplitude of the ray. This can be variable, for instance, when the source is brightest at the centre, but is assumed constant in our simulations. It is also assumed that the amplitude does not decrease with distance. k is the number of reflections the ray has undergone, and this term is included since reflection from an optically denser medium results in a half phase shift (Benenson and Stöcker [2002]).

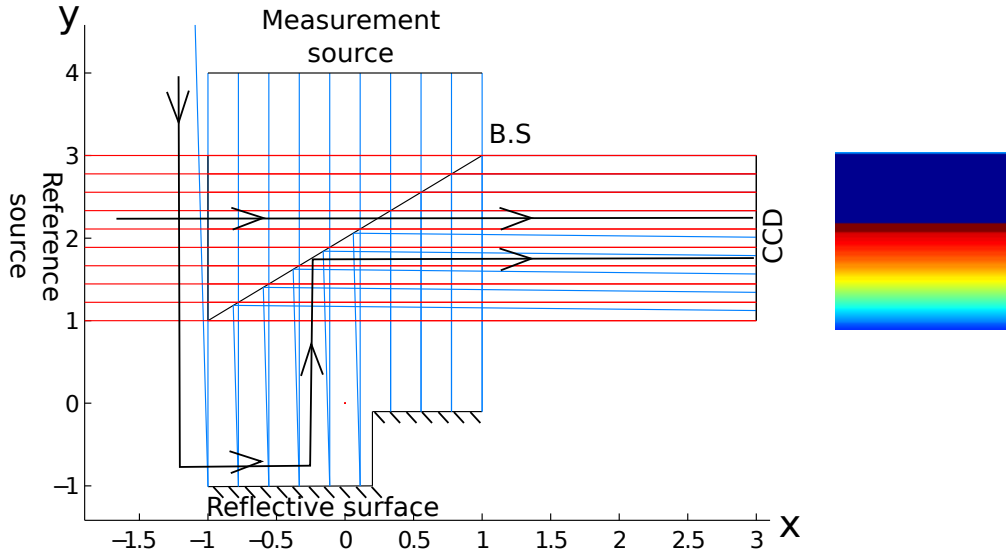


FIGURE 2.3: Simulation of a step and slant in the reflecting surface of a Michelson-type device. Left: the scene drawn by Matlab using 10 rays per beam, with the step occurring at $x = 0.2$. Generalised optical paths are shown in black for clarification. The upward gradient from $x = -1 \rightarrow 0.2$ is barely perceptible, but becomes apparent in the ray paths and resulting interferogram. From $x = 1$ onwards, measurement (blue) rays are present, but are obscured by the reference rays (red). Right: interferogram captured by the CCD, here using 100 rays per beam and 100 pixels.

In fig. 2.3 an example of the system's output is shown. To implement the beam splitter the system was programmed to ignore the reference and measurement beams' initial intersections with this object - an effective trick to get around simulating partial reflections. On the interferogram (right) the constant blue section stems from the raised flat surface over $x = 0.2 \rightarrow 1$. The gradual change in intensity through the lower half of the interferogram results from the offset measurement rays no longer having wave fronts incident on the CCD. The dark red band in the middle is a section comprised purely of reference beam, since the measurement rays have been split around it.

2.3 Optimisation

To investigate whether sample geometries could be determined through solving the ray-tracing inverse problem, optimisation tasks were developed with increasing difficulty. The simplest task was to determine the location of a step along the x -axis, where the height was known. An interferogram was generated for the given value of x , against which guesses were compared by the optimisation algorithm. Matlab's `fminsearch`

solver found x from a number of starting locations in a matter of seconds, even with 100 rays and pixels. Solving for the additional unknown of height y was also achievable provided the heights were kept low enough to prevent multiple solutions, and even several steps were possible.

The sloping geometry shown in fig. 2.3 was also solvable, demonstrating a significant advancement over the previous Gaussian beam models. The optimiser struggled with multiple gradients however, due to the high level of nonlinearity associated with the output, and the large number of measurement rays being deflected away from the CCD. The output could be linearised though incorporation of a phase-stepping algorithm (Chapter 1.1.3), and the issue of deflected rays could be remedied through lens modelling and rotation of the sample. These topics are recommended for future work.

2.4 Diffraction

To test the algorithm against 2D sections of real interferograms we constructed a Michelson Interferometer as shown in fig. 2.4.a. The measurement arm was directed onto a *spatial light modulator* capable of inducing programmable phase shifts across a 2D grid of pixels each sized $8\mu m^2$. The image shown in fig. 2.4.b was sent to the SLM to induce a phase shift, creating the impression of a rectangular displacement in the reflecting surface.

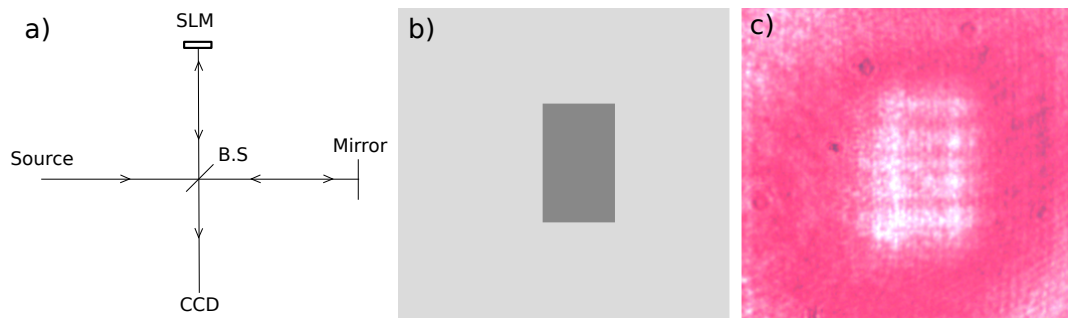


FIGURE 2.4: a) Michelson interferometer with SLM configuration, b) the image sent to the SLM, c) the interferogram captured.

As the captured interferogram in fig. 2.4.c shows, the phase shift causes constructive interference and hence a greater intensity in the captured image. However the rectangle is very blurry and the interference not constant across it. The details could not be

focussed and an image without the reference beam contribution proved to exhibit the same features, implying the distortion was caused by diffraction at the rectangle edges.

This is due to the Huygens-Fresnel principle, which states (from [Agu and Hill Jr \[2003\]](#)) that:

“Every unobstructed point of a wavefront, at a given instant in time, serves as a source of spherical secondary wavelets (with the same frequency as that of the primary wave). The amplitude of the optical field at any point beyond is the superposition of all these wavelets (considering their amplitudes and relative phases).”

In our experiment, the phase offset about the rectangle puts the emanating secondary wavelets out of phase with the surrounding ones, causing interference in the measurement beam and rendering such secondary waves visible in the image.

2.4.1 Raytracing diffraction

Previous work has been done on incorporating diffractive effects into raytracing algorithms, albeit with limited scope. In [Agu and Hill Jr \[2003\]](#) an expression for viewing angle dependent intensity over a diffraction grating was derived from the Huygens Fresnel principle, providing an extension to the specular component of a ray tracing engine. However diffraction around arbitrary objects is still a challenge to raytracing. In Chapter 3 we discuss some of the techniques used for rendering structural colour, where diffraction is often involved.

2.5 Conclusions

The raytracing algorithm is capable of producing interferogram-like output from 2D environments, and our optimisation method can determine causal structure for interferograms generated by the engine itself. However, the data we captured appears heavily distorted by diffraction, hindering us from currently working with it. There are many examples in the literature of Mach-Zehnder and phase contrast type interferograms where diffraction is not noticeable (see fig. 1.1.b), so we suggest future work considers adding a refractive component to the algorithm.

Chapter 3

Rendering structural colour

Structural colour, most commonly manifested as iridescence, is observed across the animal kingdom, with well studied specimens including the *blue morpho* butterfly family (Vukusic et al. [1999], Okada et al. [2013]), the weevil beetle (Seago et al. [2009]), and the Indian peacock (Zi et al. [2003], Yoshioka and Kinoshita [2002]). It has also been studied in plants (Glover and Whitney [2010]) and is a prominent example of interference serving a function in nature. Structural colour is the term used when an object's hue arises from features other than the colour of its pigments, though as we shall see the arrangement of pigments can play a role. Features capable of producing interference effects are called *photonic structures*, and serve different functions across the biological world, being used dynamically for camouflage in cephalopods (Mäthger et al. [2009]) and more commonly in birds, insects and fish to attract mates (Kodric-Brown [1985]).

In this chapter we examine the causes of structural colour, together with the methods for measuring it, current research into rendering it, and potential applications to technology. We present an analysis of the far field reflectance properties of photonic structures in the peacock tail feather, and investigate whether our interference raytracing algorithm can replicate these findings.

3.1 Types of structural colour

Multiple mechanisms have been evolved to produce natural iridescence. The simplest and arguably most common is thin-film interference, where the organism is coated in

a transparent layer with thickness close to wavelengths in the visible light spectrum. This is responsible for the structural coloration of fish (Denton and Land [1971]). In butterflies such as the blue morpho, exterior periodic nano structures effectively form a diffraction grating, whereby the incoming light is split into its colour components according to Huygens-Fresnel principle. In peacocks, regular lattices of the pigment melanin give rise to iridescence, possibly through Bragg reflection.

3.2 Quantifying structural colour

Microscopy techniques reviewed in Ransley [2013], specifically SEM, TEM and AFM, are commonly used to study the structures involved in iridescence at the nanometric level, though these methods carry the disadvantage of being colourless. Several techniques exist for measuring far field angular colour distribution and reflectivity, and are reviewed in detail in Vukusic and Stavenga [2009]. These include imaging scatterometry, where an ellipsoidal mirror focuses the hemisphere of scattered light into a beam, the integrating sphere method, where the sample is illuminated inside a white diffusely scattering sphere such that all reflections are received by the detector, and the practice of producing 20,000x scale replicas of the structures for interrogation with microwaves.

However the most common method for quantifying structural colour is through obtaining the *bidirectional reflectance distribution function* (BRDF), whereby the structure's reflective intensity is recorded as a function of both the angles of illumination and measurement in a hemisphere around the sample.

Obtaining the BRDF is a time consuming process, especially when working in 3D space, since the number of measurements is n^2 when n angles are considered, though modern *gonioreflectometers* are able to automate this procedure.

3.3 Rendering structural colour

Simulating iridescence in CGI is currently a very active research field. Usually, a BRDF is measured or created, and the rendered colour is subsequently taken from a lookup table based on the angles of incidence and observation. This technique has been used by Pixar and other high-end CGI studios since 2009 (Dunlop [2009]).

In [Gonzato and Pont \[2004\]](#) the BRDF was found for the *moprho menelaus* butterfly by analytically solving a thin-film model. This method had the advantage of being faster than manually determining the BRDF, though there were some inaccuracies compared to measured data due to the simplified single layer model.

In [Okada et al. \[2013\]](#) the nanoscopic lamellae of the *didius blue morpho* butterfly were modelled geometrically. A non-standard finite difference time domain (NS-FDTD) algorithm was then used to solve Maxwell's equations in full form in the presence of a single modelled lamella for a range of angles of illumination and observation, obtaining the BRDF. Agreement with experimental measurements and highly realistic renderings were obtained, demonstrating the lamellae's structure as being the sole source of colour reflectance in this case, though at high computational cost - the simulation of the BRDF took around 10 hours on an Intel Core 2 Duo 2.66 GHz parallel CPU.

3.4 Structural colour in technology

The current ubiquity of portable display technology is hindered somewhat by the liquid crystal and LED based screens available, placing restrictions on viewing angle, visibility in bright ambient lighting and miniaturisation, specifically thickness. Photonic crystal technology looks to be a promising candidate in future generations, and is discussed in [Arsenault et al. \[2007\]](#). Recent work has considered the use of colour structures to create displays with greater than diffraction limit resolution ([Wu et al. \[2013\]](#)), a concept with huge implications for the biomarking industry.

Colour structures have long been exploited decoratively through iridescent clothing, cars and appliances. Interviews with military consultants indicate a strong interest in using them for camouflage ([Birch \[2010\]](#)), though if such research is being conducted it is most likely classified.

3.5 Interferometric raytracing of structural colour

For our investigation we focus on the blue eyespot on the peacock tail feather. Its iridescence can be seen in fig. 3.1.a-b. The structural colour was initially believed to originate from thin-film interference ([Land \[1972\]](#)), but electron microscopy images show

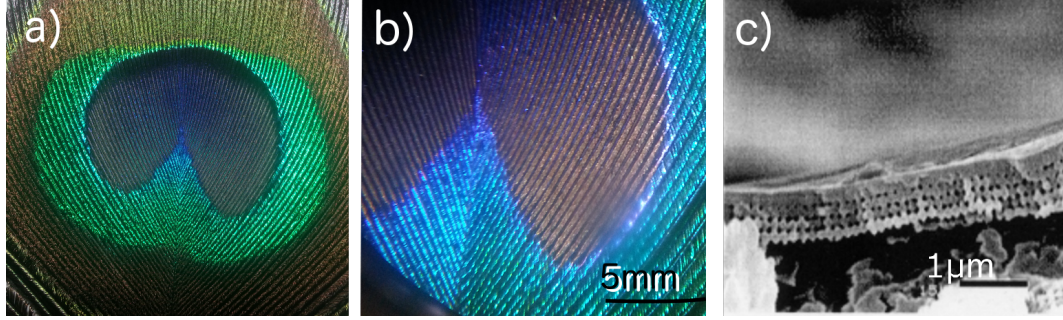


FIGURE 3.1: Eyespot of the peacock tail feather under a) 1x magnification, b) 4x magnification through an optical microscope, revealing coloration present on individual barbules. Note the colour change from green to blue resulting from varying angles of illumination. c) SEM cross section of an individual barbule (c. taken from [Yoshioka and Kinoshita \[2002\]](#)).

the barbules to contain a periodic lattice of melanin rods suspended in a transparent keratin matrix [Zi et al. \[2003\]](#).

3.5.1 Methods

To analyse the feather, we obtained the BRDF using the setup depicted in fig. 3.2. The instrument built used an LED torch as a white light source, chosen for its consistency and fairly well directed beam. A spectroradiometer was used to capture colour spectra. This instrument is frequently used in directional spectral analysis due to its sensitivity and very well directed recordings - a laser spot is emitted prior to capturing and only emissions from within it are analysed. The sample, a small blue patch of the tail feather, was placed on a black surface beneath the axes of measurement. The feather was trimmed to a symmetric alignment, reducing the domain to only one polar dimension for illumination and observation. For illumination angle θ_i and observation angle θ_o measurements were taken at (from vertical axis) 0° , $\pm 20^\circ$, $\pm 40^\circ$ and $\pm 60^\circ$, resulting in a total of 49 spectra.

As fig. 3.3 shows, there are two dominant wavelengths across the angular domain at approximately 445nm and 525nm. In fig. 3.4 the surface of intensity spectra for varying θ_i is plotted, illustrating the iridescent shift in spectral intensity. Similar results are seen for varying θ_o . Whilst we were anticipating symmetry about the polar origin for θ_i , this is not observed leading us to conclude either the structure does not exhibit the symmetric properties we believed, or else the wrong axis of measurement was chosen.

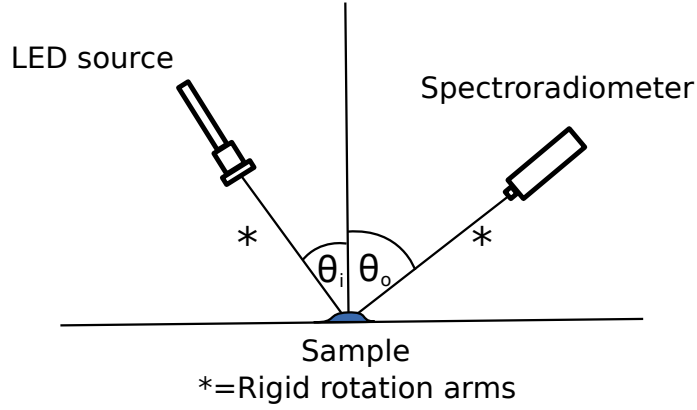


FIGURE 3.2: Schematic of the instrument used for measuring the BRDF. The instrument design did not permit axial movement beyond $\pm 60^\circ$.

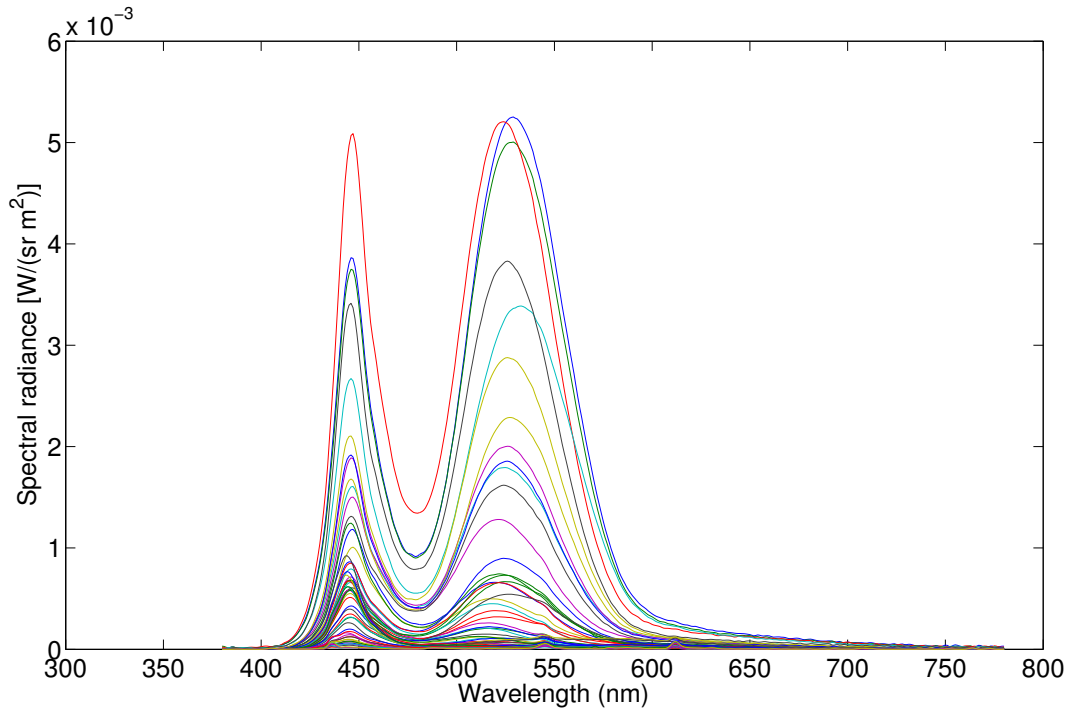


FIGURE 3.3: Spectral radiance for all 49 BRDF measurements.

3.6 Modelling the colour structure

The interferometric raytracing algorithm can currently only handle reflections, so a *Bragg reflection* model was chosen. The barbule structure was modelled as a regular lattice of 18 sided polygons, due to the measurements having been taken in increments of $\pi/9$ rad. As such, measurements from the simulated system could be taken at the same intervals, with the guarantee that light was reflected there. Due to the lack of refractive calculations, the simplifying assumption was made that the particles were suspended in air instead of a chitin matrix. Dimensions for the lattice were taken from [Yoshioka and](#)

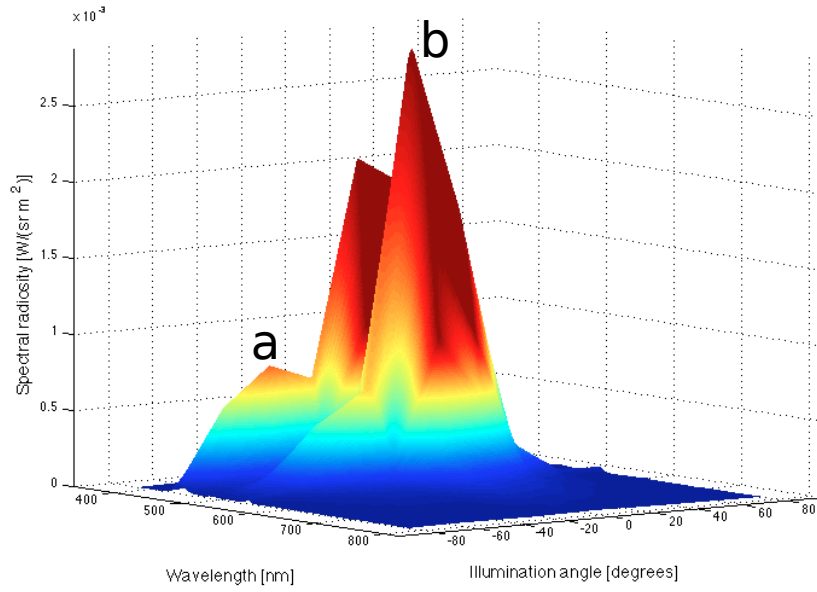


FIGURE 3.4: Spectral radiance distribution for $\theta_i \in [-60^\circ, 60^\circ]$, with $\theta_o = 0$. The shift from green to blue dominance is observable from a to b as $\theta_i \rightarrow 0$.

Kinoshita [2002]. The radius of a blue melanin rod in a peacock feather was given as $r = 65nm$ whilst the spacing between layers was $h = 150nm$. This was taken to be the spacing between extremities of each rod, since $150nm$ spacing from the centres of each particle would not allow enough room for Bragg reflection.

Bragg reflection, depicted in fig. 3.5.a, occurs on periodic lattice structures. The additional OPL travelled by the ray reflecting on the lower layer is $2h \cos \theta_i$. Hence when this distance is a multiple of the wavelength, constructive interference occurs. Our initial Matlab simulation shown in fig. 3.5.b demonstrates constructive interference at $\theta_o = \pi/9$ for secondary and even tertiary reflections.

The reference beam was turned off for this simulation, and the CCD was set to having one pixel of width equal to the source. Additional pixels were not necessary since the BRDF is an angular measurement as opposed to transverse. The simulation was run for all the measured values of θ_i and θ_o across the visible light spectrum using 2000 rays and a 20×4 lattice. This took only a few seconds. After calculating the OPLs at the CCD, the wavelengths were inserted afterwards to obtain the spectral radiance distribution, shown in fig. 3.6.

The results clearly do not fit with those measured in fig. 3.3, although the light blue line for $\theta_i = -\pi/9$, $\theta_o = \pi/9$ appears qualitatively similar. The remainder of the spectral

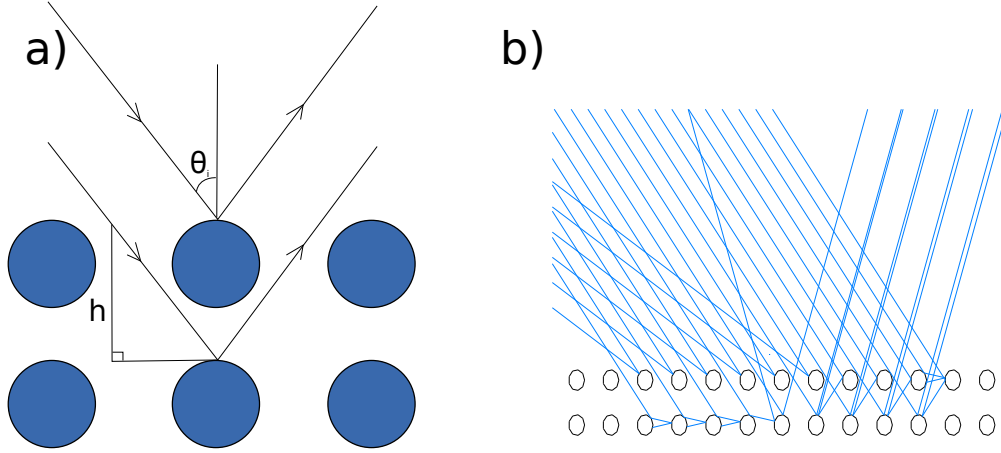


FIGURE 3.5: Bragg reflection depicted a) schematically, b) from the Matlab simulations. Maximum number of reflections is set to 2.

distributions indicate constructive interference for multiple modes of the spectrum, and the reason being that Bragg's equation

$$k\lambda = 2 \cos \theta_i$$

holds solutions for all integers k . It seems wise to conclude then that iridescence in peacock feathers is not purely the result of Bragg reflection. Perhaps including the refractive keratin matrix or allowing partial refraction through the melanin rods would effect a more realistic result, though this shall not be investigated here.

3.7 Conclusions

Importantly, this investigation has demonstrated that raytracing structural colour down at the structural level is possible, only in this case the structure was not correctly modelled. The sheer speed of the computations indicate that with some further development these techniques could be incorporated into standard ray tracers and, we hope, be of some relevance as structural colour becomes an increasingly “hot” field.

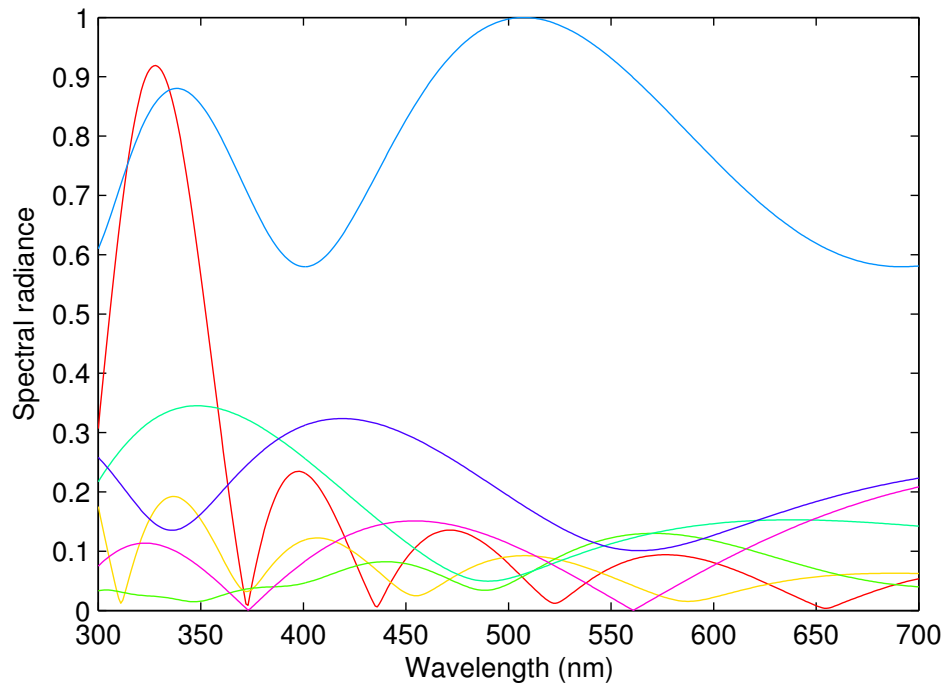


FIGURE 3.6: The results from our Matlab simulation of the Bragg model for $\theta_i = -60^\circ$, $\theta_o = \pm 60^\circ$, $\pm 40^\circ$, $\pm 20^\circ$ and 0° .

Bibliography

- Emmanuel Agu and Francis S Hill Jr. A simple method for ray tracing diffraction. In *Computational Science and Its Applications ICCSA 2003*, pages 336–345. Springer, 2003.
- André C Arsenault, Daniel P Puzzo, Ian Manners, and Geoffrey A Ozin. Photonic-crystal full-colour displays. *Nature Photonics*, 1(8):468–472, 2007.
- Walter Benenson and Horst Stöcker. *Handbook of physics*. Springer, 2002.
- Hayley Birch. How to disappear completely. 2010. URL <http://www.rsc.org/chemistryworld/Issues/2010/June/HowToDisappearCompletely.asp>.
- Benyong Chen, Xiaohui Cheng, and Dacheng Li. Dual-wavelength interferometric technique with subnanometric resolution. *Applied optics*, 41(28):5933–5937, 2002.
- Carol J Cogswell and CJR Sheppard. Confocal differential interference contrast (dic) microscopy: including a theoretical analysis of conventional and confocal dic imaging. *Journal of microscopy*, 165(1):81–101, 1992.
- Katherine Creath. Phase-measurement interferometry techniques. *Progress in optics*, 26(26):349–393, 1988.
- Ramachandra Rao Dasari, Michael S Feld, Yongjin Sung, Moonseok Kim, Youngwoon Choi, Christopher Fang-Yen, Kwanhyung Kim, and Wonshik Choi. Three-dimensional differential interference contrast microscopy using synthetic aperture imaging. SPIE, 2013.
- TJ Davis, D Gao, TE Gureyev, AW Stevenson, and SW Wilkins. Phase-contrast imaging of weakly absorbing materials using hard x-rays. *Nature*, 373(6515):595–598, 1995.

- EJ Denton and MF Land. Mechanism of reflexion in silvery layers of fish and cephalopods. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 178(1050):43–61, 1971.
- R Dunlop. The making of pixar’s up. 2009. URL <http://www.techradar.com/news/video/the-making-of-pixar-s-up-603600>.
- Beverley J Glover and Heather M Whitney. Structural colour and iridescence in plants: the poorly studied relations of pigment colour. *Annals of botany*, 105(4):505–511, 2010.
- Jean-Christophe Gonzato and Bernard Pont. A phenomenological representation of iridescent colors in butterfly wings. In *WSCG (Short Papers)*, pages 79–86, 2004.
- RA Kaul, DM Mahlmann, and P Loosen. Mach–zehnder interference microscopy optically records electrically stimulated cellular activity in unstained nerve cells. *Journal of Microscopy*, 240(1):60–74, 2010.
- K Kinnstaetter, Adolf W Lohmann, Johannes Schwider, and Norbert Streibl. Accuracy of phase shifting interferometry. *Applied Optics*, 27(24):5082–5089, 1988.
- Astrid Kodric-Brown. Female preference and sexual selection for male coloration in the guppy (*poecilia reticulata*). *Behavioral Ecology and Sociobiology*, 17(3):199–205, 1985.
- MF Land. The physics and biology of animal reflectors. *Progress in biophysics and molecular biology*, 24:75–106, 1972.
- Hein L Leertouwer, Bodo D Wilts, and Doekele G Stavenga. Refractive index and dispersion of butterfly chitin and bird keratin measured by polarizing interference microscopy. *Opt. Express*, 19(24):24061–24066, 2011.
- Ludwig Mach. Ueber einen interferenzrefraktor. *Instrumentenkunde*, 12:89–94, 1892.
- Daniel M Mahlmann, Joachim Jahnke, and Peter Loosen. Rapid determination of the dry weight of single, living cyanobacterial cells using the mach-zehnder double-beam interference microscope. *European Journal of Phycology*, 43(4):355–364, 2008.
- Sigmund Manhart, R Maurer, Hans J Tiziani, Zoran Sodnik, Edgar Fischer, A Mariani, R Bonsignori, Giancarlo Margheri, C Giunti, and Stefano Zatti. Dual-wavelength interferometer for surface profile. 1990.

- Lydia M Mäthger, Eric J Denton, N Justin Marshall, and Roger T Hanlon. Mechanisms and behavioural functions of structural coloration in cephalopods. *Journal of the Royal Society Interface*, 6(Suppl 2):S149–S163, 2009.
- Atsushi Momose, Tohoru Takeda, Yuji Itai, and Keiichi Hirano. Phase-contrast x-ray computed tomography for observing biological soft tissues. *Nature medicine*, 2(4):473–475, 1996.
- David D Nolte. *Bioanalysis: Optical Interferometry for Biology and Medicine*, volume 1. Springer, 2012.
- Naoki Okada, Dong Zhu, Dongsheng Cai, James B Cole, Makoto Kambe, and Shuichi Kinoshita. Rendering morpho butterflies based on high accuracy nano-optical simulation. *Journal of Optics*, pages 1–12, 2013.
- M Ransley. Considerations for a 3d super-resolution interferometric microscope in biology. *UCL CoMPLEX CP2*, 2013.
- Ainsley E Seago, Parrish Brady, Jean-Pol Vigneron, and Tom D Schultz. Gold bugs and beyond: a review of iridescence and structural colour mechanisms in beetles (coleoptera). *Journal of the Royal Society Interface*, 6(Suppl 2):S165–S184, 2009.
- G Stoilov and T Dragostinov. Phase-stepping interferometry: five-frame algorithm with an arbitrary step. *Optics and lasers in engineering*, 28(1):61–69, 1997.
- P Vukusic and DG Stavenga. Physical methods for investigating structural colours in biological systems. *Journal of the Royal Society Interface*, 6(Suppl 2):S133–S148, 2009.
- P Vukusic, JR Sambles, CR Lawrence, and RJ Wootton. Quantified interference and diffraction in single morpho butterfly scales. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 266(1427):1403–1411, 1999.
- Thomas Wilhein, Burkhard Kaulich, Enzo Di Fabrizio, Fillipo Romanato, Stefano Cabrini, and Jean Susini. Differential interference contrast x-ray microscopy with submicron resolution. *Applied Physics Letters*, 78(14):2082–2084, 2001.
- Yi-Kuei Ryan Wu, Andrew E Hollowell, Cheng Zhang, and L Jay Guo. Angle-insensitive structural colours based on metallic nanocavities and coloured pixels beyond the diffraction limit. *Scientific reports*, 3, 2013.

- T Wyatt. Investigation into the feasibility of a three axis interferometer for biological microscopy. *UCL CoMPLEX CP1*, 2012.
- Shinya Yoshioka and Shuichi Kinoshita. Effect of macroscopic structure in iridescent color of the peacock feathers. *FORMA-TOKYO-*, 17(2):169–181, 2002.
- Ludwig Zehnder. Ein neuer interferenzrefraktor. *Z Instrum*, 11:275–285, 1891.
- Frits Zernike. How i discovered phase contrast. *Science*, 121(3141):345–349, 1955.
- Jian Zi, Xindi Yu, Yizhou Li, Xinhua Hu, Chun Xu, Xingjun Wang, Xiaohan Liu, and Rongtang Fu. Coloration strategies in peacock feathers. *Proceedings of the National Academy of Sciences*, 100(22):12576–12578, 2003.

Appendix A

Matlab Code

The code used has been included here since it forms a large part of the original work conducted in this project, and may contain techniques that are of interest. Working versions of the code can be found at <http://www.ucl.ac.uk/~ucbpran/research.html>. Where the prime (') symbol denotes transposition, it has been replaced with (") to avoid L^AT_EX typesetting conflicts.

```

% RTfcn.m traces rays from a given source (sourcecoords) through a given
% scene (objset) to a given CCD (CCDrange). Outputs captured CCD
% intensities.
function intensity=RTfcn(sourcecoords,CCDrange,objset)
hold off
clc
bounds=[0 -1 2 1]; % [scene bounds - [xmin, ymin, xmax, ymax].
sceneon=1; % Plot scene (rays, objects, CCD).
reference=0; % Use reference beam?
maxrefs=12; % Maximum number of reflections + 1.
lamda=5.4e-2; % Wavelength of monochromatic light source.
nrays=800; % Number of rays used.
CCDres=100; % Number of pixels along CCD.

% // Calculate gradients & normal vectors for each surface object
objgrad=(objset(:,4)-objset(:,2))./(objset(:,3)-objset(:,1));
objyint=objset(:,2)-objset(:,1).*objgrad;
v2=[objset(:,3)-objset(:,1) objset(:,4)-objset(:,2)];
v2n=zeros(size(v2));
for i = 1:size(v2,1)
    v2(i,:) = v2(i,:)/norm(v2(i,:));
    if v2(i,2)==0
        if v2(i,1)==1
            v2n(i,:)=[0 1];
        else
            v2n(i,:)=[0 -1];
        end
    elseif v2(i,1)==0
        if v2(i,2)==-1

```

```

        v2n(i,:)=[1 0];
    else v2n(i,:)=[-1 0];
    end
else
    v2n(i,:)=[1,-v2(i,1)/v2(i,2)];
    v2n(i,:)=v2n(i,:)/norm(v2n(i,:));
end
end

%// Setup ray matrix:
% Column key: (1)normalvect[x],(2)normalvect[y],(3)gradient,(4)startcoords[x],(5)startcoords[y],
% (6)endcoords[x],(7)endcoords[y],(8)obj.int,(9)yint
raymat=zeros(nrays,9,maxrefs);
if sourcecoords(4)==sourcecoords(2) % Special case when beam points vertically down.
    raymat(:,1,1)=0;
    raymat(:,2,1)=-1;
    raymat(:,3,1)=-inf;
    raymat(:,4,1)=linspace(sourcecoords(1),sourcecoords(3),nrays);
    raymat(:,5,1)=sourcecoords(2);
    raymat(:,6,1)=raymat(:,4,1);
    raymat(:,7,1)=bounds(2);
else
    g1=-1/((sourcecoords(4)-sourcecoords(2))/(sourcecoords(3)-sourcecoords(1)));
    raymat(:,1:2,1)=repmat([1 g1]/norm([1 g1]),nrays,1);
    raymat(:,3,1)=g1;
    raymat(:,4,1)=linspace(sourcecoords(1),sourcecoords(3),nrays);
    raymat(:,5,1)=linspace(sourcecoords(2),sourcecoords(4),nrays);
    raymat(:,9,1)=raymat(:,5,1)-raymat(:,3,1).*raymat(:,4,1);
    kkk = raymat;

```

```

end
// Update raymatrix according to layout of the scene:
raymat=computerefs(raymat,objset,objgrad,objjint,v2n,bounds);

// REFERENCE BEAM
if reference==1
    ref.coords=[sourcecoords(3),sourcecoords(2),sourcecoords(1),sourcecoords(4)];
    ref.grad = -1/((ref.coords(4)-ref.coords(2))/(ref.coords(3)-ref.coords(1)));
    ref.nrays=nrays;
    refmat=zeros(ref.nrays,9,maxrefs);
    refmat(:,1:2,1)=repmat([1 ref.grad]/norm([1 ref.grad]),ref.nrays,1);
    refmat(:,3,1)=ref.grad;
    refmat(:,4,1)=linspace(ref.coords(1),ref.coords(3),ref.nrays);
    refmat(:,5,1)=linspace(ref.coords(2),ref.coords(4),ref.nrays);
    refmat(:,9,1)=refmat(:,5,1)-refmat(:,3,1).*refmat(:,4,1);
    refmat(:,6,1)=bounds(3);
    refmat(:,7,1)=refmat(:,3,1)*bounds(3)+refmat(:,9,1);
    CCDrefrays=cell(CCDres,4);
    refmat=computerefs(refmat,objset,objgrad,objjint,v2n,bounds);
end

// Draw objects, sources, rays & CCD (can be very time consuming).
if sceneon==1
    for i = 1:size(objset,1);
        line([objset(i,1),objset(i,3)], [objset(i,2),objset(i,4)], "color", "k");
        hold on;
    end
    for i=1:nrays

```

```

for k=1:maxrefs-1
    line([raymat(i,4,k),raymat(i,6,k)], [raymat(i,5,k),raymat(i,7,k)])
end
end
line([sourcecoords(1),sourcecoords(3)], [sourcecoords(2),sourcecoords(4)], "color", "k");
line([CCDrange(1) CCDrange(3)], [CCDrange(2), CCDrange(4)], "color", "k")
if reference==1
    line([ref.coords(1), ref.coords(3)], [ref.coords(2), ref.coords(4)], "color", "k");
    for i=1:ref.nrays
        for k = 1:maxrefs-1
            line([refmat(i,4,k),refmat(i,6,k)], [refmat(i,5,k),refmat(i,7,k)], "color", "r");
        end
    end
end
axis([bounds(1) bounds(3) bounds(2) bounds(4)])
end
end

% // CAMERA
CCDrays=cell(CCDres,4); % For each pixes, contains: [which rays hit it, which parts of each ray hit it, OPL
% of each ray hitting it, phase of each ray hitting it].

inc=(CCDrange(4)-CCDrange(2))/CCDres;
if CCDrange(3)~=CCDrange(1)
    inc = sqrt((CCDrange(3)-CCDrange(1))^2+(CCDrange(4)-CCDrange(2))^2)/CCDres;
    CCDgrad=(CCDrange(4)-CCDrange(2))/(CCDrange(3)-CCDrange(1));
    CCDyint=CCDrange(2)-CCDgrad*CCDrange(1);
end

for j=1:CCDres
    xmin=min(CCDrange(1)+(j-1)*(CCDrange(3)-CCDrange(1))/CCDres, CCDrange(1)+j*(CCDrange(3)-CCDrange(1))/CCDres);

```

```

xmax=max(CCDrange(1)+(j-1)*(CCDrange(3)-CCDrange(1))/CCDres,CCDrange(1)+j*(CCDrange(3)-CCDrange(1))/CCDres);
ymin=min(CCDrange(2)+(j-1)*(CCDrange(4)-CCDrange(2))/CCDres,CCDrange(4)+j*(CCDrange(4)-CCDrange(2))/CCDres);
ymax=max(CCDrange(2)+(j-1)*(CCDrange(4)-CCDrange(2))/CCDres,CCDrange(4)+j*(CCDrange(4)-CCDrange(2))/CCDres);

for k = 1:size(raymat,3)-1      % Compute hits on each pixel.
    for i = 1:size(raymat,1)
        v=raymat(i,3,k)*CCDrange(1)+raymat(i,9,k);
        if CCDrange(3)==CCDrange(1) && v>=CCDrange(2)+(j-1)*inc && v<= CCDrange(2)+j*inc && ...
            min(raymat(i,4,k),raymat(i,6,k))<=CCDrange(1) && max(raymat(i,4,k),raymat(i,6,k))>=CCDrange(3)
            if k==1 | (k>1 & raymat(i,6:7,k)~=raymat(i,6:7,k-1))
                CCDrays{j,1}=[CCDrays{j,1},i];
                CCDrays{j,2}=[CCDrays{j,2},k];
                CCDrays{j,3}=[CCDrays{j,3},OPL(raymat(i,:,:),k,CCDrange(1))];
                CCDrays{j,4}=[CCDrays{j,4},mod(OPL(raymat(i,:,:),k,CCDrange(1)),lamda)*2*pi/lamda];
            end
        elseif CCDrange(3)~=CCDrange(1)
            CCDint=(CCDyint-raymat(i,9,k))/(raymat(i,3,k)-CCDgrad);
            % Below: Does the ray hit the pixel? EDIT add a CCDyint to
            % simplify.
            if CCDint>=xmin && CCDint<=xmax && raymat(i,3,k)*CCDint+raymat(i,9,k)>=ymin && raymat(i,3,k)*CCDint...
                +raymat(i,9,k)<=ymax && CCDint>=min(raymat(i,4,k),raymat(i,6,k)) && CCDint <= max(raymat(i,4,k),...
                    raymat(i,6,k)) && CCDint*raymat(i,3,k)+raymat(i,9,k)>=min(raymat(i,5,k),raymat(i,7,k)) && ...
                    CCDint*raymat(i,3,k)+raymat(i,9,k)<=max(raymat(i,5,k),raymat(i,7,k))
                if k==1 | (k>1 & raymat(i,6:7,k)~=raymat(i,6:7,k-1)) % Prevents rays stopped at the boundary from solving...
                    CCDrays{j,1}=[CCDrays{j,1},i];          % repeatedly with remaining reflection iterates.
                    CCDrays{j,2}=[CCDrays{j,2},k];
                    CCDrays{j,3}=[CCDrays{j,3},OPL(raymat(i,:,:),k,CCDint)];
                    CCDrays{j,4}=[CCDrays{j,4},mod(OPL(raymat(i,:,:),k,CCDint),lamda)*2*pi/lamda];
                end
            end
        end
    end
end

```



```

end
end
end
end

if reference==1
    for k = 1:(size(refmat,3)-1)
        for i = 1:size(refmat,1)
            v=refmat(i,3,k)*CCDrange(1)+refmat(i,9,k);
            if CCDrange(3)==CCDrange(1) && v>=CCDrange(2)+(j-1)*inc && v< CCDrange(2)+j*inc && min(refmat(i,4,k),...
                refmat(i,6,k))<=CCDrange(1) && max(refmat(i,4,k),refmat(i,6,k))>=CCDrange(3)
                    if k==1 | (k>1 & refmat(i,6:7,k)~=refmat(i,6:7,k-1))
                        CCDrefrays{j,1}=[CCDrefrays{j,1},i];
                        CCDrefrays{j,2}=[CCDrefrays{j,2},k];
                        CCDrefrays{j,3}=[CCDrefrays{j,3},OPL(refmat(i,:,:),k,CCDrange(1))];
                        CCDrefrays{j,4}=[CCDrefrays{j,4},mod(OPL(refmat(i,:,:),k,CCDrange(1)),lamda)*2*pi/lamda];
                    end
                elseif CCDrange(3)~= CCDrange(1)
                    CCDint=(CCDyint-refmat(i,9,k))/(refmat(i,3,k)-CCDgrad);

                    if CCDint>=xmin && CCDint<=xmax && refmat(i,3,k)*CCDint+refmat(i,9,k)>=ymin && refmat(i,3,k)*CCDint+...
                        refmat(i,9,k)<=ymax && CCDint>=min(refmat(i,4,k),refmat(i,6,k)) && CCDint <= max(refmat(i,4,k),...
                            refmat(i,6,k)) && CCDint*refmat(i,3,k)+refmat(i,9,k)>=min(refmat(i,5,k),refmat(i,7,k)) && ...
                                CCDint*refmat(i,3,k)+refmat(i,9,k)<=max(refmat(i,5,k),refmat(i,7,k))
                            if k==1 | k>1 & refmat(i,6:7,k)~=refmat(i,6:7,k-1)
                                CCDrefrays{j,1}=[CCDrefrays{j,1},i];
                                CCDrefrays{j,2}=[CCDrefrays{j,2},k];
                                CCDrefrays{j,3}=[CCDrefrays{j,3},OPL(refmat(i,:,:),k,CCDint)];

```

```

        CCDrefrays{j,4}=[CCDrefrays{j,4},mod(OPL(refmat(i,:),:),k,CCDint),lamda)*2*pi/lamda];
    end
end
end
end
end
intensity=zeros(CCDres,1); % Compute intensity of each pixel.
if reference==1
    for i = 1:CCDres
        intensity(i,1)=abs(sum(exp(1i*opl2phase(CCDrays{i,3},lamda)))+sum(exp(1i*opl2phase(CCDrefrays{i,4},lamda))));
    end
end
else
    for i = 1:CCDres
        intensity(i,1)=abs(sum(exp(1i*opl2phase(CCDrays{i,3},lamda))));
    end
end
end
if CCDon==1
    figure
    imshow(repmat(flipud(intensity),1,CCDres))
    caxis([0 5])
end

```

```

% computerefs.m updates the raymatrix according to the layout of the scene.
% Additional data such as objgrad and v2n are passed through to save
% computation. bounds is required to prevent rays going to infinity when
% leaving the scene.
function raymat=computerefs(raymat,objset,objgrad,objjint,v2n,bounds)
for k = 1:size(raymat,3)-1
    hitmat=zeros(size(raymat,1),size(objset,1)); % Will contain x-coordinate of every possible intersection.
    %// Determine which rays hit which objects
    for n=1:size(raymat,1)
        for m=1:size(objset,1)
            if objset(m,3)==objset(m,1) % Vertical surfaces (infinite gradient) require special treatment.
                xint = objset(m,1);
                if raymat(n,3,k)*xint+raymat(n,9,k)>=min(objset(m,2),objset(m,4)) ...
                    && raymat(n,3,k)*xint+raymat(n,9,k)<max(objset(m,4),objset(m,2))
                    hitmat(n,m)=xint; % if statement requires intersection to occur within the line bounds.
                end
            else
                % Non-vertical case.
                xint = (-raymat(n,9,k)+(objset(m,2)-objgrad(m)*objset(m,1))./(raymat(n,3,k)-objgrad(m)));
                yint = raymat(n,3,k)*xint+raymat(n,9,k);
                if(min(objset(m,1),objset(m,3))<=xint && xint<=max(objset(m,1),objset(m,3))) ...
                    && yint>=min(objset(m,4),objset(m,2)) && yint...
                        <=max(objset(m,4),objset(m,2));
                    hitmat(n,m)=xint;
                end
            end
        end
    end
    if abs(raymat(n,3,k))==inf && raymat(n,4,k)>=min(objset(m,1),objset(m,3)) && raymat(n,4,k)<=max(objset(m,1),...
        objset(m,3))
        hitmat(n,m)=raymat(n,4,k); % Vertical rays require special treatment.
    end
end

```

```

if sign(hitmat(n,m)-raymat(n,4,k))~=sign(raymat(n,1,k)) && hitmat(n,m)-raymat(n,4,k)~=0
    hitmat(n,m)=0; % Clear entry if ray is travelling away from the surface.
end
end
end
%// Find which object, if any, is hit first for each ray & calculate appropriate reflectance gradients.
HM=zeros(size(hitmat));
for i=1:size(hitmat,1)
    for j=1:size(hitmat,2)
        if hitmat(i,j)~=0
            if abs(objgrad(j))==inf
                HM(i,j)=sqrt((hitmat(i,j)-raymat(i,4,k))^2+(raymat(i,3,k)*hitmat(i,j)+raymat(i,9,k)-raymat(i,5,k))^2);
            else
                HM(i,j)=sqrt((hitmat(i,j)-raymat(i,4,k))^2+(hitmat(i,j)*objgrad(j)+...
                    objjint(j)-raymat(i,5,k))^2); % Pythagorean distance of hit from start point.
            end
        end
    end
end
end
HM(abs(HM)<1e-12)=0; % Removes the starting point from possible hits, in case of numerical error.
HM(HM<=0)=inf; % This will allow the closest hit to be found.
%// Update end coordinates where object is hit, and start coordinates for reflected ray.
for i=1:size(raymat,1)
    [Y I]=min(HM(i,:));
    if Y~=inf
        raymat(i,8,k)=I;
        if k>1 && raymat(i,8,k)==raymat(i,8,k-1)

```

```

    raymat(i,8,k)=0;      % Needed to stop rays at bounds.
end
end
if raymat(i,8,k)~=0
    if raymat(i,3,k)~= -inf
        raymat(i,6,k)=hitmat(i,1); % Terminate ray at hitpoint.
        raymat(i,7,k)=raymat(i,3,k)*raymat(i,6,k)+raymat(i,9,k);
    else raymat(i,6,k)=raymat(i,4,k);
        raymat(i,7,k)=objjgrad(raymat(i,8,k))*raymat(i,4,k)+objjint(raymat(i,8,k));
    end
end

if abs(raymat(i,6,k))==inf      % Terminate ray at bounds if need be.
    if sign(raymat(i,1,k))==1
        raymat(i,6,k)=bounds(3);
        raymat(i,7,k)=bounds(3)*raymat(i,3,k)+raymat(i,9,k);
    else
        raymat(i,6,k)=bounds(1);
        raymat(i,7,k)=bounds(1)*raymat(i,3,k)+raymat(i,9,k);
    end
end

if abs(raymat(i,7,k))==inf
    if sign(raymat(i,2,k))==1
        raymat(i,7,k)=bounds(4);
    else
        raymat(i,7,k)=bounds(2);
    end
end

end
%// Compute normal vector, gradient, start points and
% y-intercept for reflected ray.

```

```

raymat(i,1:2,k+1)=-2*dot(raymat(i,[1 2],k),v2n(raymat(i,8,k,:)))*v2n(raymat(i,8,k,:))+raymat(i,[1 2],k);
raymat(i,3,k+1)=raymat(i,2,k+1)/raymat(i,1,k+1);
raymat(i,4:5,k+1)=raymat(i,6:7,k);
raymat(i,9,k+1)=raymat(i,5,k+1)-raymat(i,3,k+1)*raymat(i,4,k+1);
if sign(raymat(i,1,k+1))==1 % Ensure ray begins by terminating at bounds (hits computed on next run).
    raymat(i,6,k+1)=bounds(3);
    raymat(i,7,k+1)=raymat(i,3,k+1)*bounds(3)+raymat(i,9,k+1);
elseif sign(raymat(i,1,k+1))==-1
    raymat(i,6,k+1)=bounds(1);
    raymat(i,7,k+1)=raymat(i,3,k+1)*bounds(1)+raymat(i,9,k+1);
else % EDIT may need y-direction conditions.
    raymat(i,6,k+1)=raymat(i,4,k+1);
    raymat(i,7,k+1)=bounds(4);
end
end
if k>1 && (raymat(i,6,k-1)==bounds(1)||raymat(i,6,k-1)==bounds(3) || raymat(i,5,k-1)==bounds(2)...
    || raymat(i,5,k-1)==bounds(4))
    raymat(i,6:7,k)=raymat(i,4:5,k); % Ensures when a ray terminates at bounds it stays that way.
end
if k == 1 && raymat(i,8,k)==0
    raymat(i,7,k)=bounds(2);
    if raymat(i,3,k)==-inf
        raymat(i,6,k)=raymat(i,4,k);
    else
        raymat(i,6,k)=(raymat(i,7,k)-raymat(i,9,k))/raymat(i,3,k);
    end
end
end
end

```

end

% opl2phase.m computes the phase at the end of a beam's path.

function phase=opl2phase(OPL,lamda)

phase=mod(OPL,lamda)*2*pi/lamda;

end

function clattice=clattice(resolution,numx,numy,radius,spacing,offset) % Generates a lattice of circles to be used as a scene.

n=(1:resolution)*2*pi/resolution;

x = radius*cos(n)+offset(1);

y = radius*sin(n)+offset(2);

clattice=[];

for i = 1:numx

 for j = 1:numy

 if mod(j,2)==0

 clattice=[clattice;formatobject([x"+spacing*(i-1)+spacing/2,y"+spacing*(j-1)])];

 else

 clattice=[clattice;formatobject([x"+spacing*(i-1),y"+spacing*(j-1)])];

 end

 end

end
