All You Need Is Light - Characterizing Beetles With Stokes Parameters

Scheinowitz, Naor

All You Need Is Light
Characterizing Beetles With Stokes Parameters

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
in
PHYSICS

Author :	N.M.A. Scheinowitz
Student ID :	1531921
Supervisor :	Dr. M.J.A. de Dood
Second corrector :	Prof.dr. T. Oosterkamp

Leiden, The Netherlands, April 19, 2022

All You Need Is Light
Characterizing Beetles With Stokes Parameters

N.M.A. Scheinowitz

Huygens-Kamerlingh Onnes Laboratorium, Universiteit Leiden
P.O. Box 9500, 2300 RA Leiden, The Netherlands

April 19, 2022

Abstract

Within the diverse order of beetle species that produce color through structural coloration, some beetles produce light with a strong left-handed circular polarization caused by a chiral structure in their outer shells. From an evolutionary perspective, there should be no substantial benefit to left-handed over rather-handed polarization. This raises the question why no beetles have been found showing right-handed polarization. Large-scale beetle surveys are required to investigate this question further. A systematic beetle classification scheme based on polarization would aid in such a survey and highlight species with interesting optical properties for further investigation. To this end we constructed a setup capable of measuring the Stokes parameters of a beetle specimen for angles of incidence and observation. We tested this setup on the beetle species *protaetia speciosa jousselini* and verified that the setup produces accurate results. We are however sceptical this method will result in identifying beetle-specific Stokes parameters, as their values vary with respect to both location on the beetle cuticle as well as observational angles. Furthermore we provide quantitative evidence for earlier reported inversion of the polarization handedness in beetle species *jousselini* at large angles of observation. We detect such inversion at angles of observation of at least 70°.

Cover picture: collage of different beetles taken from Seago et al. (2009).[1] The heart in the center shows images of beetle species *protaetia speciosa jousselini* taken with a mirroless camera through polarizing filters, with on the left a left-handed circular polarization filter and on the right a right-handed circular polarization filter.

Contents

Chapter 1

Introduction

Through millions of years of biological evolution, the animal kingdom contains many species with utterly astounding abilities. Take for example the *alpheidae*, a group of shrimp species capable of snapping their claws with devastating effect. Their claws are specialized to create cavitation bubbles, which, upon implosion, create acoustic pressures large enough to kill small fish. Unsurprisingly, these types of shrimp are commonly called *pistol shrimp*.[2]

Another amazing phenomenon involves color. Most objects we encounter in our daily lives produce color by absorption and partial reflection by chemical compounds. A large number of animals have devised a completely different mechanism: *structural coloration*. Instead of using chemical properties, these animals exhibit wavelength-scale microstructures on their surface which produce color by interference effects. The resulting iridescent hues can be incredibly saturated and otherwise rare in nature. Structural coloration has been found in birds, fish, and insects.[3]

In this thesis we focus on one particular type of animal: beetles. In terms of number of different species, beetles outnumber all other animals: approximately 25% of all animal species are thought to be beetle species.[4, 5] Beetles live in every habitat and their appearances range from the camouflaged to the extravagant. With such broad diversity of species, it is no surprise that different mechanisms for producing color have been found in beetles. Despite this diversity, most beetles that show structural coloration have some form of multilayer reflector mechanism, where alternating layers of different refractive indices cause constructive interference at a particular frequency.[1]

Some beetles, however, have a curious modification to the multilayer reflector mechanism: they produce light that is strongly left-handed circularly polarized.[1, 6–9] Since circular polarized light is rare in nature, one might wander if these beetles obtain some evolutionary benefit thanks to this mechanism. Some biologists suggest a link to intra-species communication [10–12], but there is still the possibility it has no evolutionary benefit at all.

Whatever the case may be, a more fundamental question about this ability remains. Why is the left-handed orientation prevalent when evolving right-handed polarizing iridescence should be equivalent? This phenomenon of seemingly arbitrary asymmetries can be found throughout nature, from the chirality of snail shells [13] to the internal layout of human organs. Nature seems to 'choose' specific orientations and reproduce them with a high degree of accuracy. For example, reversal of the organ layout (*situs inversus totalis*) occurs in only about 0.015% of

human newborns.[14] The cause of these asymmetries is a yet unanswered question in biology.

Since beetle species are so numerous, large scale surveys are required to better understand the origin of this circular polarization. To date, no beetle species has been found with a right-handed polarization, nor do we know if a genetic mutation exists that can reverse the polarization orientation within a species. To perform such a survey, one needs some systematic approach to measuring and perhaps even classifying beetle specimen.

The need for a systematic beetle classifier formed the synthesis for this research project. We wanted to investigate whether the Stokes parameter formalism can be used as a classification scheme for beetles with polarizing iridescence mechanisms. In this thesis we describe our attempt at constructing a setup that can produce images of a beetle with pixel-specific values of the Stokes parameters. We tested our setup with a beetle of the species *protaetia speciosa jousselini*. This beetle produces left-handed polarized light, but perhaps more interestingly, has been reported to produce right-handed polarized light as well under certain conditions.[9, 15]

Chapter 2

Background

Through millions of years of evolution, numerous species have devised a remarkable way of producing color. Most objects we encounter in our daily lives produce color through absorption by chemical compounds, reflecting only certain wavelength (ranges) of light due to their molecular energy spectra. Many animals however, from birds to fish to insects, produce color through wavelength-scale structure. This *structural coloration* has its origin in interference effects caused rather than pigments. This wavelength-sized structure could be the result of the underlying microscopic structure of molecules through self-assembly processes, but the nature of this self-assembly is unknown and is part of active research on structural color.

Unlike colors that are produced by ordinary absorption and scattering, hues produced by structural coloration tend to shift as the angle of illumination changes, a phenomenon defined as *iridescence*. Different animals produce iridescent colors differently. Even within the species of insects of order coleoptera* there is a great diversity of iridescence mechanisms.[1] In section 2.1 we give a brief overview of the known iridescence mechanisms used by beetles as context for the later chapters.

Some, but not all, iridescence mechanisms produce light that is left-handed circularly polarized. This begs the question: are all beetles left-circularly polarized or do species exist that produce right-handed circularly polarized light? If not, what is the cause for this asymmetry? To answer these questions, large surveys of beetle species are required. While we do not propose or perform such a survey in this project, we investigate whether beetles can be classified based on their polarizing properties. Specifically, we attempt to find a unique signature of a beetle using the Stokes parameter formalism. Such a signature could, for example, consist of a numerical value for Stokes parameter S_3, which denotes the amount of circularly polarized light. As a case study, we investigated the beetle *protaetia speciosa jousselini* as it not only produces left-handed polarized light, but also right-handed circular polarization under the right circumstances.[9, 15] In section 2.2 we expand on the compelling characteristics of this beetle species.

Our method for characterizing the polarization of the beetle involved measuring its Stokes parameters. Using this quantitative approach we hoped to find a specific signature for the beetle involved, or at the very least learn something about its iridescence mechanism. In section 2.3 we introduce the theoretical basis for the Stokes parameter calculation and in section 3.1 we

*Entomological term for beetles.

introduce the physical setup used for performing the measurements.

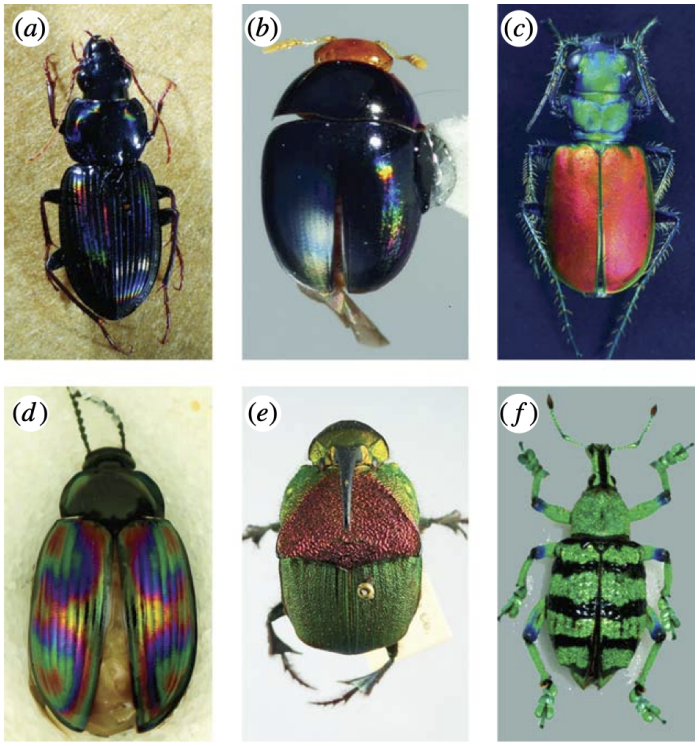

Figure 2.1: *Examples of beetle iridescence. (a-b) diffraction grating, (c-e) multilayer reflector, (f) photonic crystal. Image taken from Seago et al. (2008).[1]*

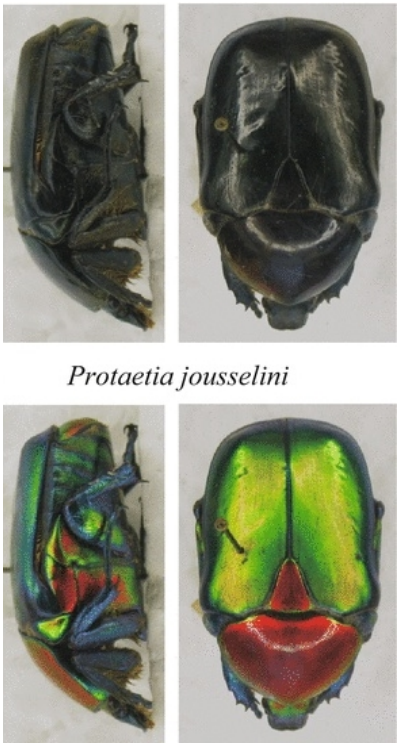

Figure 2.2: *Bottom: image of protaetia speciosa jousselini through a left-handed polarization filter. Top: image through a right-handed polarization filter. Image taken from Horváth et al. (2014).[6]*

2.1 Overview of Iridescence Mechanisms in Coleoptera

While no two species of coleoptera have identical structural color, most can be grouped into three main mechanisms: diffraction gratings, multilayer reflectors, and three-dimensional photonic crystals. We describe these groups in general, but we highly recommend reading the review by Seago et al. (2008) for an in-depth description.[1]

The most common iridescence mechanism in beetles is the multilayer reflector, three examples of which are shown in figure 2.1c-e. This structure consists of multiple layers of differing refractive index, commonly built up by layers of chitin. These layers cause light at particular wavelengths to interfere constructively producing dominant hues. Since the path length of light through this structure is angle-dependent, the wavelength at which constructive interference occurs is angle-dependent as well. This results in a blue-shift in the visible hues when the observation angle increases.

Other beetles show a surface structure that consists of microscopic ridges, edges, or slits with a broad-scale preferred direction. Such a structure acts as a diffraction grating and produces a rainbow-like iridescence pattern, as seen in figure 2.1a-b.

The final common structure has some similarities to the multilayer reflector. Where the multilayer is essentially a one-dimensional structure or crystal, some beetles show a structure that is three-dimensional in nature. In other words, the structure consists of a highly ordered lattice with a periodicity comparable to the wavelength of visible light. Typically, these structures are

not perfect as they are not ordered over the entire beetle surface, but rather form patches of ordered structure. This results in a more diffuse reflection and thus reduced angle dependence of the visible color, see figure 2.1f.

2.2 Motivation for the Case Study on *Protaetia Speciosa Jousselini*

What sets *protaetia speciosa jousselini*, and some other beetle species, apart from the overwhelming majority of coleoptera is the fact that it reflects light that is left-handed circularly polarized, as seen optically in figure 2.2.[6–8] Circular polarization is rare in nature, and even in human experience. Most people, without even realizing it, will only encounter circular polarization while watching a 3D movie.[16, 17]

The mechanism for producing circularly polarized light in this beetle is a modification to the multilayer reflector. The main difference is that the chitin molecules order into long fibers that are themselves aligned into two-dimensional layers. These rods of chitin molecules are strongly anisotropic and thus optically birefringent. Each consecutive layer of chitin fibers is rotated with respect to the underlying layer which creates a material with a helical structure that shows circular birefringence. Light with a wavelength that matches the periodicity of this helix is then constructively interfered and circularly polarized.

Biologists argue about potential evolutionary benefits of circular polarization reflectance. A common hypothesis is that it aids in communication through vision.[10–12] We will not attempt to answer this question here. Instead, in this work we will try to characterize this polarization signal quantitatively through Stokes parameters, in the hope of defining a unique polarization signature for *protaetia speciosa jousselini*.

2.2.1 Handedness Inversion and Comparison to Bragg Mirror

The multilayer reflector that is so common among beetles is essentially a form of dielectric mirror or Bragg mirror, a material built up of layers of alternating refractive index such that constructive interference occurs at reflections of a specific wavelength (range). The properties of Bragg mirrors can be modeled numerically using Berreman calculus, a transfer matrix method for Mueller calculus.[8]

Using this approach, Leiden University student Tim Reisinger found that beetles of species *jousellini* are in fact capable of producing right-handed polarized light despite their surface structure being left-handed. This effect becomes apparent at angles of incident larger than 40° and will be accompanied by a blue-shift of the reflection peak at increasing angles. This suggests that this transformation from left to right-handed polarization indeed originates from the helical structure itself. The presence of right-handed circularly polarized light has also been observed experimentally by Hagedüs et al. (2006) in species *jousellini*.[9, 15]

We will attempt to add to the existing experimental and numerical evidence on this handedness inversion by observing the Stokes parameters of the beetle at extreme angles of incidence.

2.3 Stokes Formalism

Characterizing the polarization of iridescent light from coleoptera (or any light source for that matter) can be done using the Stokes parameter framework developed in the 19th century. It is a set of four parameters that completely decompose and characterize the polarization state

of light. The four parameters are defined as combinations of the electric field in different bases for polarized light:

$$S_0 = \langle E_x^2 \rangle + \langle E_y^2 \rangle, \tag{2.1a}$$
$$S_1 = \langle E_x^2 \rangle - \langle E_y^2 \rangle, \tag{2.1b}$$
$$S_2 = \langle E_a^2 \rangle - \langle E_b^2 \rangle, \tag{2.1c}$$
$$S_3 = \langle E_r^2 \rangle - \langle E_l^2 \rangle. \tag{2.1d}$$

Here the brackets $\langle \cdot \rangle$ denote time averages over the electric field. The basis (x, y) is the standard Cartesian basis of linearly polarized light in horizontal and vertical directions, (a, b) is the cartesian basis rotated by 45 degrees, and (r, l) is the circular basis defined such that $\hat{l} = (\hat{x} + i\hat{y})/\sqrt{2}$, $\hat{r} = (\hat{x} - i\hat{y})/\sqrt{2}$. The four parameters can equivalently be defined purely in the Cartesian basis:

$$S_0 = \langle E_x^2 \rangle + \langle E_y^2 \rangle, \tag{2.2a}$$
$$S_1 = \langle E_x^2 \rangle - \langle E_y^2 \rangle, \tag{2.2b}$$
$$S_2 = 2\langle E_x \rangle \langle E_y \rangle \cos(\delta), \tag{2.2c}$$
$$S_3 = 2\langle E_x \rangle \langle E_y \rangle \sin(\delta). \tag{2.2d}$$

Where δ is the phase difference between the x and y components of the beam. The four Stokes parameters are often combined into a *Stokes Vector* $\vec{S} \equiv (S_0, S_1, S_2, S_3)$.

The first Stokes parameter gives the total intensity of a beam of light, which, in experimental contexts where this quantity is not relevant in absolute terms, is often used to normalize the Stokes vector such that $S_0 = 1$. The parameters $S_1 \dots S_3$, and in particular their sign, allow for clear decomposition of the polarization of a beam of light into horizontal (+) or vertical (-), plus or minus 45°, and right- (+) or left-handed (-) polarization, respectively.

2.3.1 Motivation for Preferring Stokes over Jones Formalism

The experienced reader might wonder why we are using the Stokes/Mueller calculus with 4-vectors and 4x4 matrices instead of the numerically simpler Jones calculus, which describes polarized states with 2-vectors and optical elements with 2x2 matrices. The reason is straightforward: Jones vectors can only describe purely polarized states. Experiments involving coherent or fully polarized light can use this fewer-element calculus. But in cases where one wants to account for light with arbitrary (partial) polarization and the effects of depolarization, one cannot avoid using Stokes/Mueller.

In this particular thesis, we have set our goal at measuring a polarization profile of the reflection and scattering of light on surfaces of coleoptera. We have no reason to expect that the response to unpolarized light of these species is purely polarized, hence we have chosen to design our measurement apparatus in accordance with the Stokes formalism.

2.3.2 Measuring the Stokes Vector

Although the Stokes parameters cannot be measured directly, they can be inferred by the modulation of the intensity of a light source transmitted through two optical elements in series: a rotating quarter-wave plate followed by a fixed linear polarizer.[8, 18]

We can derive this relation analytically using Mueller calculus. This technique allows for the manipulation of Stokes vectors through (4x4) matrices that represent specific optical elements. For an incident beam with Stokes vector $\vec{S}_{\text{in}}$ passing through an element represented by Mueller matrix M, the outgoing Stokes vector $\vec{S}_{\text{out}}$ is simply

$$\vec{S}_{\text{out}} = M\vec{S}_{\text{in}}\,. \tag{2.3}$$

One of the benefits of Mueller calculus is that, through the associativity of matrix multiplication, several optical elements in series can be combined into a single matrix $\tilde{M}$ that describes the entire measurement setup:

$$\vec{S}_{\text{out}} = M_N M_{N-1} \ldots M_2 M_1 \vec{S}_{\text{in}} = \tilde{M}\vec{S}_{\text{in}}\,. \tag{2.4}$$

The order of the matrices is key. Matrix multiplication is not commutative, i.e. $M_i M_j \neq M_j M_i$, and operates from right to left. Therefore, the first optical element in the beam path, is represented by the right-most matrix and the last optical element by the left-most. For our purposes, we need only two Mueller matrices: the matrix for a horizontal linear polarizer,

$$M_{\text{HL}} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{2.5}$$

and the matrix for a general linear retarder,

$$M_{\text{retarder}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos^2(2\theta) + \sin^2(2\theta)\cos(\delta) & \cos(2\theta)\sin(2\theta)(1-\cos(\delta)) & \sin(2\theta)\sin(\delta) \\ 0 & \cos(2\theta)\sin(2\theta)(1-\cos(\delta)) & \cos^2(2\theta)\cos(\delta) + \sin^2(2\theta) & -\cos(2\theta)\sin(\delta) \\ 0 & -\sin(2\theta)\sin(\delta) & \cos(2\theta)\sin(\delta) & \cos(\delta) \end{pmatrix}, \tag{2.6}$$

where θ is the rotation angle of the linear retarder and δ is the phase difference that the retarder introduces.[19] For a rotating quarter-wave plate ($\delta = \frac{\pi}{2}$), matrix 2.6 simplifies to:

$$M_{\lambda/4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos^2(2\theta) & \cos(2\theta)\sin(2\theta) & \sin(2\theta) \\ 0 & \cos(2\theta)\sin(2\theta) & \sin^2(2\theta) & -\cos(2\theta) \\ 0 & -\sin(2\theta) & \cos(2\theta) & 0 \end{pmatrix}. \tag{2.7}$$

The combined Mueller matrix for the setup with a linear polarizer and a quarter-wave plate is then

$$\tilde{M} = M_{\text{HL}} M_{\lambda/4} = \frac{1}{2}\begin{pmatrix} 1 & \cos^2(2\theta) & \cos(2\theta)\sin(2\theta) & \sin(2\theta) \\ 1 & \cos^2(2\theta) & \cos(2\theta)\sin(2\theta) & \sin(2\theta) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{2.8}$$

A beam of light of arbitrary polarization passing through this system will have its polarization manipulated such that it exits the system with a Stokes vector of the form:

$$\vec{S}_{\text{out}} = \tilde{M}\,\vec{S}_{\text{in}} \tag{2.9}$$

$$= \frac{1}{2}\begin{pmatrix} S_0 + S_1\cos^2(2\theta) + S_2\cos(2\theta)\sin(2\theta) + S_3\sin(2\theta) \\ S_0 + S_1\cos^2(2\theta) + S_2\cos(2\theta)\sin(2\theta) + S_3\sin(2\theta) \\ 0 \\ 0 \end{pmatrix}. \tag{2.10}$$

The beam exits the system fully linearly polarized, but with the information of its original polarization encoded in the first two Stokes parameters. Since S_0 is simply the total intensity of the light source (see equation 2.1a), we have an exact relation between a measurable intensity and the degree of rotation of the quarter-wave plate:

$$I(\theta) = \frac{1}{2}\left(S_0 + S_1\cos^2(2\theta) + S_2\cos(2\theta)\sin(2\theta) + S_3\sin(2\theta)\right) . \tag{2.11}$$

Using the half-angle formula $\cos^2(2\theta) = 1/2 + 1/2\cos(4\theta)$ and the equality $\cos(2\theta)\sin(2\theta) = 1/2\sin(4\theta)$, we can rewrite the intensity equation to:

$$I(\theta) = \left(\frac{1}{2}S_0 + \frac{1}{4}S_1\right) + \left(\frac{1}{2}S_3\right)\sin(2\theta) + \left(\frac{1}{4}S_1\right)\cos(4\theta) + \left(\frac{1}{4}S_2\right)\sin(4\theta) \tag{2.12a}$$

$$= A + B\sin(2\theta) + C\cos(4\theta) + D\sin(4\theta) . \tag{2.12b}$$

Notice how the intensity relation has turned into a Fourier series with amplitudes $A \ldots D$. In experimental context one will always measure the intensity over discrete and finite samples of θ. Therefore, the values of amplitude $A \ldots D$ can be found by taking sample averages of products of I with (co)sines of the corresponding frequencies. Assuming we measure N different angles θ, we Fourier components are calculated by:

$$A = \frac{2}{N}\sum_n^N I(\theta_n) \tag{2.13a}$$

$$B = \frac{4}{N}\sum_n^N I(\theta_n)\sin(2\theta_n) \tag{2.13b}$$

$$C = \frac{4}{N}\sum_n^N I(\theta_n)\cos(4\theta_n) \tag{2.13c}$$

$$D = \frac{4}{N}\sum_n^N I(\theta_n)\sin(4\theta_n) . \tag{2.13d}$$

Finally, the four Stokes parameters can be extracted from the Fourier amplitudes:

$$S_0 = A - C \tag{2.14a}$$

$$S_1 = 2C \tag{2.14b}$$

$$S_2 = 2D \tag{2.14c}$$

$$S_3 = B \tag{2.14d}$$

As an example, figure 2.3 shows the expected intensity modulation as a function of wave-plate rotation angle for different Stokes vectors. The right side of the figure shows how the various Stokes vectors are represented in Fourier space. Clearly, the present angular frequencies correspond to the ones in equation 2.13.

The question remains how many different angles θ should be sampled to achieve an accurate result for the values of the Fourier components $A \ldots D$. The Nyquist criterion states that a continuous signal can be reconstructed from discrete samples if the sampling rate is at least twice as large as its largest frequency. The largest angular frequency in equation 2.12b is 4, so we conclude we must sample at least 8 different angles, where the sampling rate is $\Delta\theta = 180^\circ/8$.

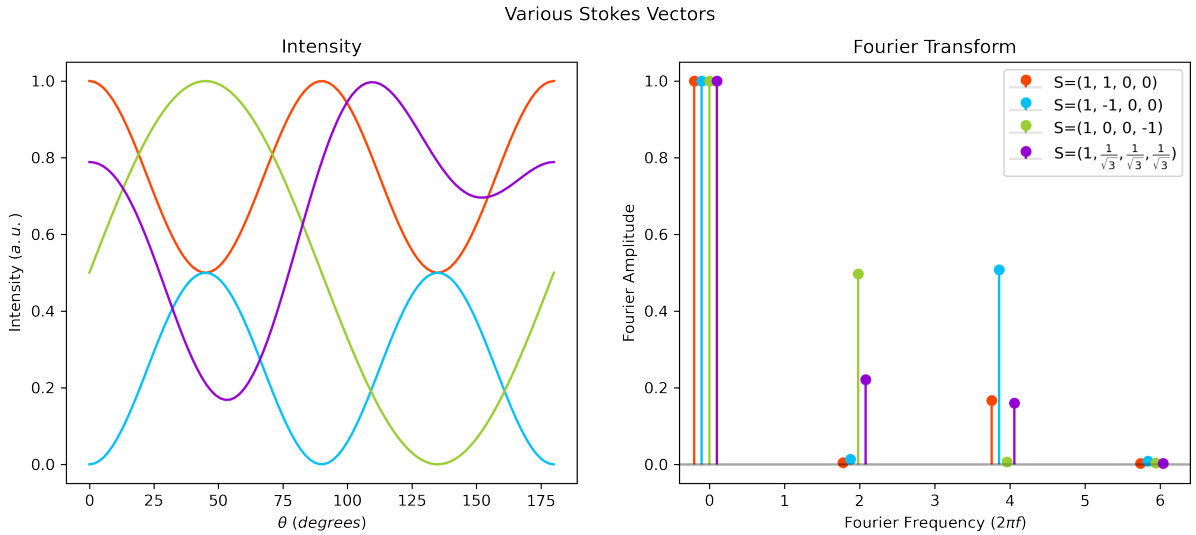

Figure 2.3: *Left: angle-dependent modulation of intensity of light with four different polarizations, according to equation 2.11. Right: Fourier transform of the left graph using the Fast Fourier Transform algorithm. The Fourier amplitudes shown here are the absolute values of the complex Fourier amplitudes. The four circles at each frequency are slightly displaced horizontally to increase readability but should be interpreted as being centered at the corresponding x-axis label. Both graphs were created numerically using 100 equidistant values of θ between 0 and 180 degrees.*

2.3.3 Note on Stokes Parameter Ambiguity

The interpretation of the Stokes parameters is inherently tied to the defined frame of reference of the experimental setup, as can clearly be seen in equation 2.1. As is often the case, we define our basis such that the x-axis is parallel to the horizontal direction for a lab observer. This implies for $S_1 = 1$ a fully horizontally polarized beam, $S_1 = -1$ fully vertically polarized etc.

However, an ambiguity to the interpretation for S_3 remains. The direction of rotation for a circularly polarized beam depends on perspective. Left-handed circular polarization from the *source perspective* is measured as right-handed from the *detector perspective*. In practice, changing between the two perspectives requires little more than switching the sign of S_3. But the experimenter must be weary to keep in mind from which perspective they are describing their experiment.

The procedure for measuring the Stokes vector in section 2.3.2 follows the source perspective of a beam of light traveling through optical elements. However, literature on the polarization of coleoptera, logically, describe the polarization of specimen from the perspective of the outside observer. Hence, to avoid confusion or apparent contradiction, we will present our findings in chapter 4 defined from the detector perspective.

2.3.4 Error Estimation

Through error propagation we can get a quantitative estimate of the error on all four Stokes parameters. We start with the error on a intensity measurement. If we measure the intensity at every wave plate rotation angle θ_n M times, we define the error on $I(\theta_n)$ as

$$s_{I(\theta_n)} = \frac{\sigma(\{I(\theta_n)\})}{\sqrt{M}}, \tag{2.15}$$

where σ is the sample standard deviation on the collection of measurements $\{I(\theta_n)\}$. Next, we calculate the error propagation by considering the variance formula

$$s_f = \sqrt{\sum_i \left(\frac{\partial f}{\partial x_i} s_{x_i}\right)^2} \tag{2.16}$$

where s_f is the error on function f, x_i are the variables that f depends on and s_{x_i} are their errors.[20] When applied to the equation for the first Fourier component, equation 2.13a, we get the following error equation:

$$\begin{aligned} s_A &= \sqrt{\left(\frac{\partial}{\partial I(\theta_n)} \frac{2}{N} \sum_n^N I(\theta_n) s_{I(\theta_n)}\right)^2} \\ &= \frac{2}{N} \sqrt{\sum_n^N s_{I(\theta_n)}^2}\,. \end{aligned} \tag{2.17}$$

Note here that the error is inversely proportional both to the number of angles measured N and to $\sqrt{M}$, where M is the total number of intensity measurements. Similarly, we get for the other Fourier components:

$$s_B = \frac{4}{N} \sqrt{\sum_n^N \sin^2(2\theta_n) s_{I(\theta_n)}^2} \tag{2.18}$$

$$s_C = \frac{4}{N} \sqrt{\sum_n^N \cos^2(4\theta_n) s_{I(\theta_n)}^2} \tag{2.19}$$

$$s_D = \frac{4}{N} \sqrt{\sum_n^N \sin^2(4\theta_n) s_{I(\theta_n)}^2}\,. \tag{2.20}$$

Here we assume that the error on the wave-plate angles θ are negligible. Finally, we can propagate these equations further to get errors for the individual Stokes parameters:

$$s_{S_0} = \sqrt{s_a^2 + s_C^2} = \frac{2}{N} \sqrt{\sum_n^N (1 + 4\cos^2(4\theta_n)) s_{I(\theta_n)}^2} \tag{2.21a}$$

$$s_{S_1} = 2s_C = \frac{8}{N} \sqrt{\sum_n^N \cos^2(4\theta_n) s_{I(\theta_n)}^2} \tag{2.21b}$$

$$s_{S_2} = 2s_D = \frac{8}{N} \sqrt{\sum_n^N \sin^2(4\theta_n) s_{I(\theta_n)}^2} \tag{2.21c}$$

$$s_{S_3} = s_B = \frac{4}{N} \sqrt{\sum_n^N \sin^2(2\theta_n) s_{I(\theta_n)}^2} \tag{2.21d}$$

In subsequent chapters the above equations will define the error values on Stokes parameter measurements.

Chapter 3

Method

The goal of this project is to gain understanding of the polarizing reflection characteristics of the beetle species *speciosa jousellini* by studying its Stokes parameters. To do this systematically, we need a device that can measure the Stokes parameters, a scheme to systematically define incoming and outgoing reflection angles, and some automation scheme to collect the data. In this chapter we describe how we built a setup capable of fulfilling these requirements.

3.1 Setup

The measurement setup can be split into two parts: source and imaging, as seen in figure 3.1. The source consists of a Xenon arc lamp connected to an optical fiber with a 50 μm core-size. The diverging beam from the fiber is collimated using a 50 mm lens and is directed to the beetle through a variable aperture.

While the light-producing setup is relatively straightforward, one aspect is key: the use of a high brightness source coupled to the optical fiber to create a well-collimated beam. We want to investigate the reflection of the cuticle of a beetle as a function of both incoming and outgoing angle. If we use a focused beam, or a standard household lamp for that matter, we introduce a wide collection of incoming beams angles unto our sample, muddying the resulting measurements. By using a collimated beam, we instead maximally constrict this factor in our setup. The limiting factor in this accuracy is the cuticle of the beetle itself. Its curvature is so significant that any practical collimated light source will have some uncertainty with regards to the exact angle of incidence on the cuticle surface.

The downside of the use of a collimated spot on the beetle is brightness. The amount of light that actually reaches the beetle is so sparse that multiple-second exposures are required to detect a signal with a reasonable signal-to-noise ratio.

A necessary practical corollary to the use of a collimated beam is the introduction of an aperture. Without one, there would be no means to selecting the area on the beetle to be illuminated.

The imaging part of the setup measures light reflected by the beetle, transmitted by a series of three optical elements: an achromatic quarter-wave plate, a linear polarizer and a 75 mm lens. By rotating the wave plate around an angle $\theta \in [0, \pi]$ and fixing the linear polarizer at a

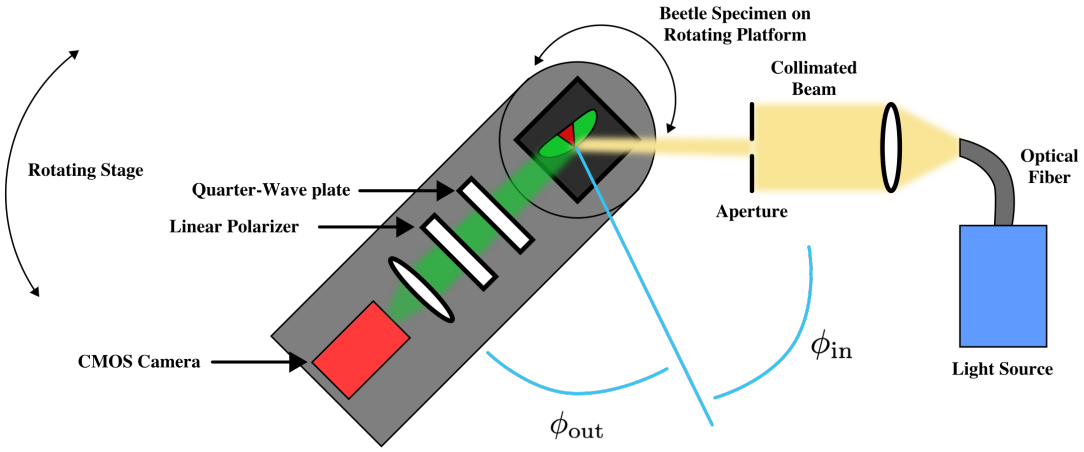

Figure 3.1: *The measurement setup used the measure the polarization of a species of coleoptera. The setup can be split into two parts: source and imaging. The source produces a collimated beam of white light (shown here in yellow), while the imaging part measures reflected light (shown here in green) modulated by a rotating quarter-wave plate and a linear polarizer. Both the platform holding the specimen and the arm holding the imaging device can rotate freely in the same plane. The reflection angles ϕ_{in} and ϕ_{out} are defined in a co-rotating frame with the specimen.*

horizontal* position, the intensity of reflected light from the beetle is modulated according to equation 2.11. The collected light is focused into a CMOS camera and read-out by a connected PC. Collectively, this setup allows for the measurement of the Stokes vector of light reflected or scattered by a specimen on a per-pixel basis.

Both the stage holding the imaging setup as well as the platform with the specimen-holder can rotate freely around a central axis, allowing investigation of reflection characteristics for many combinations of angles. The reflection angles ϕ_{in} and ϕ_{out} are defined such that the normal axis rotates along with the beetle, as shown in figure 3.1.

3.1.1 Automation

The very nature of this project implies collecting a large amount of data. Varying both incoming and outgoing reflection angles, measuring the intensity at different values for rotation angle θ, and repeating the measurement multiple times for statistical validity, one quickly accumulates thousands of single-measurement events. Any experimenter faced with such a daunting amount of measurements, tedious if done manually, will be compelled to automate as many elements of their setup as possible.

To tackle this problem, all to-be moved parts of the setup were connected to a combination of motion controllers. These devices connect to at most three stepper-motors and allow for accurate, independent rotation of each. Four parts of the setup were connected to two controllers in such a manner: the platform holding the beetle specimen (ϕ_{in}), the arm holding the imaging setup (ϕ_{out}), the quarter-wave plate (θ), and the linear polarizer. As noted in the previous section, the measurements themselves were performed by a CMOS camera. Both it and the motion controllers were connected using a USB interface to a measurement PC.

These devices did not free the experimenter of all physical involvement in measurements however: selecting a spot on the specimen to be illuminated still required manual alignment and

*Using the convention stated in section 2.3.3

aperture adjustments. Once aligned though, long series of measurements could be performed automatically through a scripting program.

To this end an object-oriented approach to scripting was chosen, using Python to drive the involved hardware. Although Python is not the most performant programming language, its ease of use and broad availability of (third-party) libraries made it an obvious choice. In pure modular fashion, separate classes were written for each hardware device. Many custom functions were written with built-in error handling to make the experience of using the setup as user-friendly as possible. In the same vain, a master document was written using *Jupyter Notebook* with general instructions on how to operate the different subsystems, while simultaneously functioning as a bare-bones interface. Appendix A contains the essential details on how the code works and how hardware challenges were circumvented.

3.1.2 Data Management

The amount of single-measurement events involved in this project evidently produce large amounts of data. A simple example: simple a measurement sequence involving a fixed ϕ_{in}, ten values for ϕ_{out} and the bare minimum eight values for θ, measured on a sensor with 1000x1000 pixels, yields 240 megabytes of data (not counting file format overhead). If one repeats the measurement for statistical validity or adds more values of θ, the amount of raw data quickly grows to the order of gigabytes.

Clearly, this project required a robust data management system. The data format HDF5 quickly emerged as a suitable format. It has a number of compelling features specifically for research projects where data takes the form of a n-dimensional data cube. This makes the format popular in fields where observations are image-based, which for example is predominantly the case in astronomy.

HDF5 is a system that stores a folder-like hierarchy in a single data file. N-dimensional arrays are stored as *datasets*, which in turn can be grouped in to *groups*. What makes HDF5 particularly useful for our purposes is that each object in the hierarchy can be assigned an arbitrary amount of metadata. This allows, for example, each camera setting to be attached to every dataset, which is incredibly useful in the data analysis stage of any project.

Additionally, HDF5 supports saving arrays using lossless data compression, which can reduce the required storage by an order of magnitude. It does this while still allowing for on-the-fly array indexing using `NumPy` syntax. For computers with (relatively) modern processors, this is a no-regret strategy to data storage.

Unfortunately, HDF5 also comes with it's drawbacks. Opening and manipulating the files requires either out-dated software or the use of a non user-friendly command-line interface. Furthermore, while stored data is very robust and can be accessed quickly, *storing* data is at least inconvenient and at most volatile. Data cannot be inspected while a measurement is running, an error during an experiment can lead to total data corruption and even the act of transferring a large file over the network can result in data loss. Caution is clearly advised. Despite these qualms, the benefits of the format prevailed and it remained the chosen data format for this project. Appendix A.2.3 contains information on a custom Python script that was written to ease the use of files.

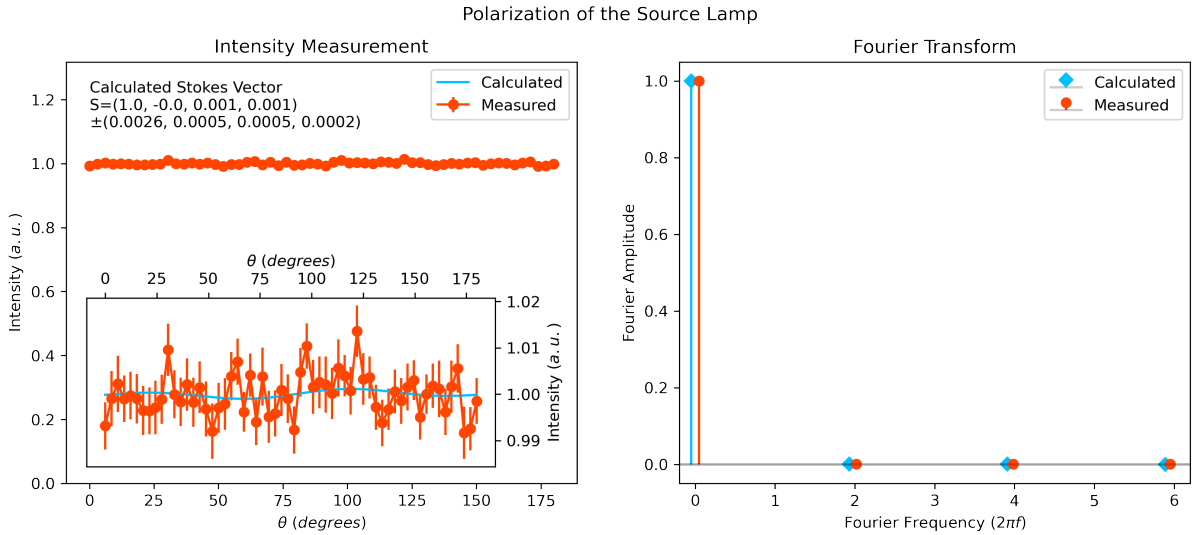

Figure 3.2: *Measurement of the polarization of the source lamp. The intensity of the lamp was measured for 60 different values of θ and is calculated by summing all pixel counts of the CMOS sensor. Both the fit on the left as the Fourier transform on the right imply a negligible net polarization.*

3.2 Calibration

To properly characterize the polarization of a specimen, we must investigate if the setup has an inherent polarization preference or characteristic. Upon first inspection, two parts of the setup require such investigation: the source lamp and the camera. However, by design the latter can be ignored. The final polarizing element in the setup is a fixed linear polarizer, eliminating the relevance of a preferred polarization of the camera. The lamp however warrants some attention.

We measured the polarization of the lamp in a manner identical to how the polarization of a specimen is to be measured: by passing its beam through the rotating wave-plate and the linear polarizer. The imaging stage was rotated such that the camera was pointing straight at the collimated beam produced by the lamp. The results of the measurement are shown in figure 3.2. The result clearly implies a negligible inherent polarization originating from our source.

While we didn't expect any significant polarization in our light source, we did have concerns about its output consistency. Arc lamps inherently vary in brightness on sub-second timescales. This is obviously visible to the experimenter as 'flickering' of the beam. To quantify this variation in time, we measured the brightness of the lamp a hundred times at two exposure times many orders of magnitude apart: 0.1 ms and 10^4 ms. The histograms of these measurement are shown in figure 3.3.

At 0.1 ms exposures the distribution has an obvious mean but with a long tail of high brightness peaks. These peaks of brightness reach up to a 20% higher value than the mean. This distribution shows the 'spark-like' behavior that we expect from an arc lamp. The same cannot be said about the exposures at 10 s. In this regime we see that the high intensity sparks are averaged out in time, leading to a normally distributed set with a clear mean and a standard deviation corresponding to about 0.2%.

What do these results imply for the subsequent measurements with beetles? In practice, light

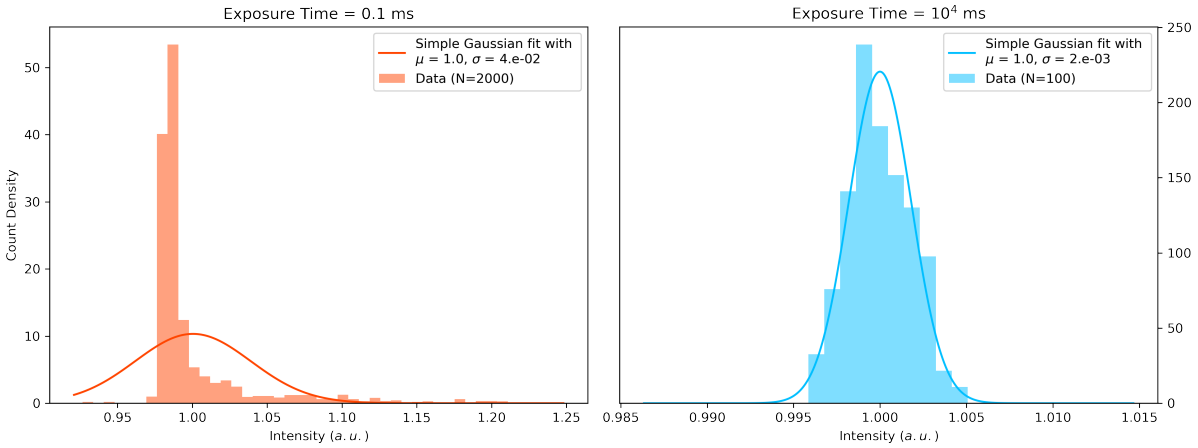

Figure 3.3: *Histograms of lamp brightness measurements at two different exposure times. The 0.1 ms exposures are spaced out in time by 36 ms due to processing limitation of the CMOS camera. Brightness values are calculated by summing all pixel counts. The gaussian fit is defined as* $G(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$ *where* μ *is the sample mean and* σ *is the sample standard deviation.*

is a scarce quantity in the lab setting and long exposures are required to obtain a usable beetle image. In fact, all measurements were performed with exposures larger than 1 second, giving us confidence that we are in the normally distributed regime of the lamp.

Chapter 4

Results

Using the measurement setup described in the previous chapter we can create a four dimensional data cube with axes [ϕ_{out}, x, y, stokes vector] for each angle of incoming light ϕ_{in}. Figure 4.1 shows a representation of this data for a specific combination of (ϕ_{in}, ϕ_{out}).

4.1 Stokes Vector as Unique Signature

Figure 4.1 already showcases a large challenge that we faced during measurement. The beetle is highly reflective. When illuminated with a non-diffuse light source, this creates large peaks of specular reflection in our data. In practice, using our 8-bit CMOS sensor, this means we cannot properly expose the entire beetle surface illuminated by the collimated beam. Either we properly expose the specular reflections or as much as possible of the other parts. Normally one would attempt to minimize influence from specular reflections with linear polarization filters. But this is inherently impossible in our setup.

In chapter 2.2 we expressed the hope to use these Stokes measurements to find a unique signature of the beetle *jousellini*. However we see that all Stokes parameters are non-uniform on the surface of the beetle, and that, as we will see later, they vary in magnitude depending on the angle of incidence and angle of observation. This is somewhat surprising for S_3, as the magnitude of circular polarization is orientation independent, unlike S_1 and S_2. An explanation for this observation is that varying amounts of linearly polarized surface reflections suppress the proportion of detected circular polarized light. In short, we are skeptical that this type of measurement can be used as a reliable classification scheme for beetles.

4.2 Handedness Inversion

To get a more general value of the angle dependent magnitude of the Stokes vector, we performed a measurement with decreased beam size such that only the red triangle on the beetle surface is illuminated. We measured the intensity of reflected light at various angles and treated the entire camera sensor as a single pixel. Essentially we measure the average value of the Stokes vector on this part of the beetle surface. The exposure time was lowered to prevent overexposing the specular reflections.

Figure 4.2 shows the result we were looking for that and was earlier reported by Reisinger [15]

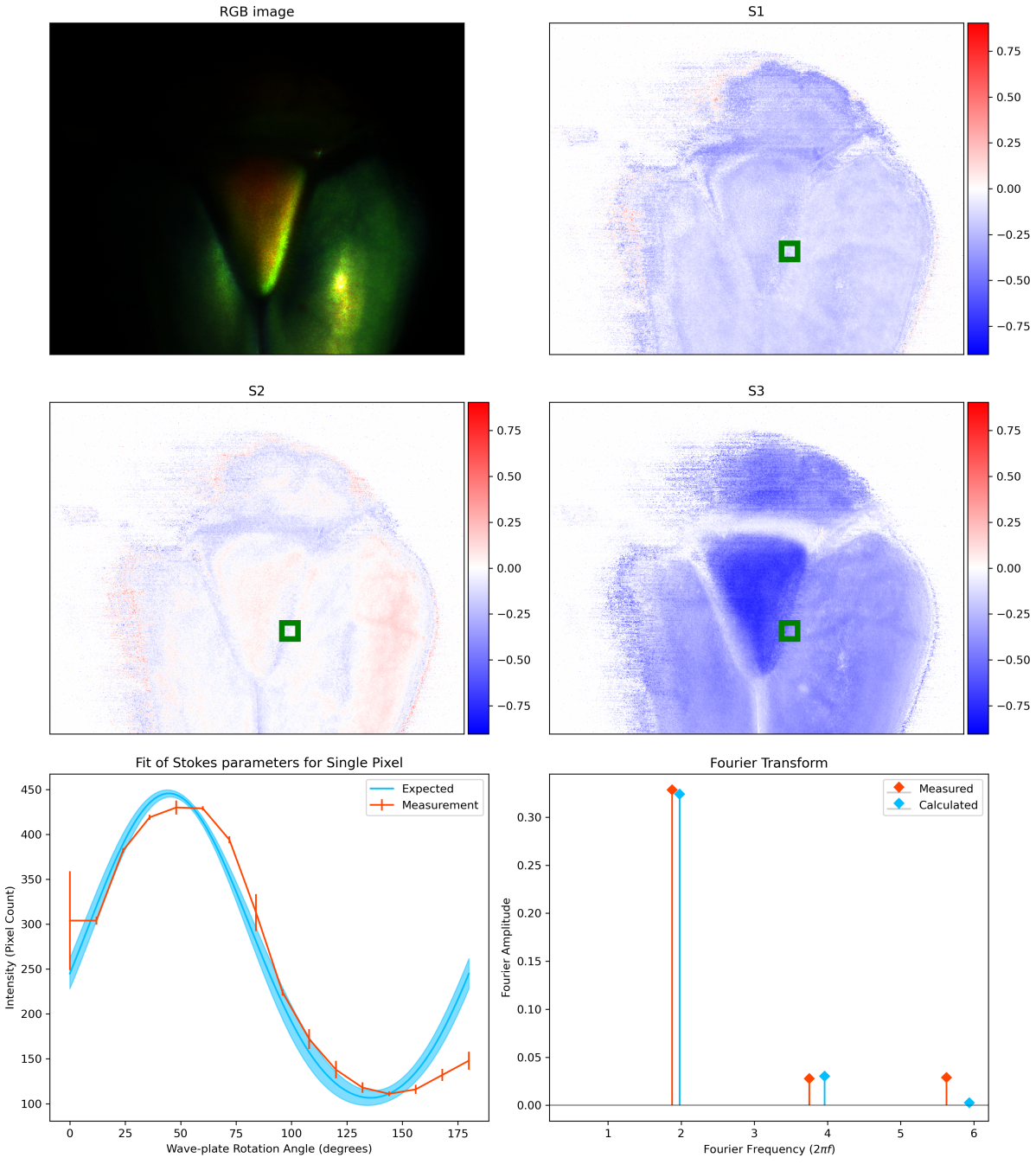

Figure 4.1: *Representation of the beetle data cube for* $(\phi_{in}, \phi_{out}) = (40°, 30°)$. *The first four images show the visual RGB image and the three Stokes parameters S1-S3, where the magnitude is represented as color intensity. The bottom two plots show the Stokes vector fit on one particular pixel, shown in the upper images as a green box. The shaded blue region in the fit shows what the intensity would look like for one standard error from the fitted value:* $I(\theta, S = S \pm s_S)$, *as defined in equation 2.21.*

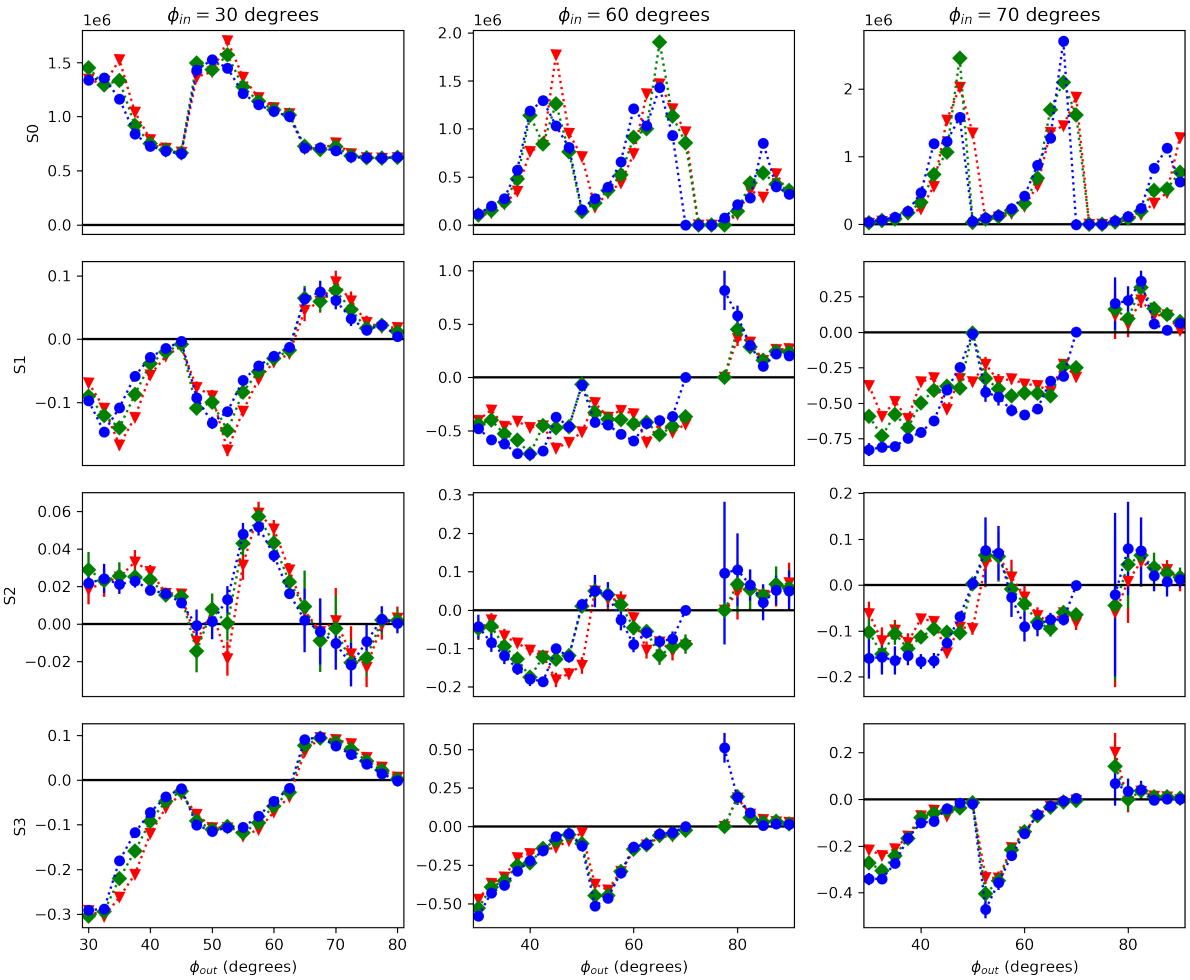

Figure 4.2: *Stokes parameters on the red triangle of the beetle surface for different angle configurations. The three columns show data for different incident angles and the rows show the four Stokes parameters. The three different colors and shapes represent the camera color channels red (triangle) blue (circle) and green (diamond). Parameters S_{1-3} are normalized such that at each value of ϕ_{out}, S_0 is scaled linearly to unity.*

and Hegedüs [9]. We see that at large angles, the circular polarization is inverted from left to right-handed. The result is most clear for $\phi_{\text{in}} = 30°$, where we have a clear signal (i.e sufficient reflected light) for all angle values. At larger angles we see that the signal vanishes between high intensity peaks that we infer are caused by very bright reflections. The fact that we see multiple peaks in each column is due to the fact that the red triangle is a part of the surface where multiple curvatures meet at a seam. Each peak in curvature causes its own specular reflection.

Can we see this inversion as well visually, similar to the images in figure 4.1? We performed a similar measurement to the one that gave the result of figure 4.2, but this time with the aperture fully opened and the camera exposure set to its maximal value of 10 s. In figures 4.3 and 4.4 we present a series of images showing the circular polarization on the beetle surface for increasing values of ϕ_{out} for an angle of incidence of 40° and 60°.

In both series of images we see that the beetle surface is overwhelmingly left-handed polarized. Only at $(\phi_{in}, \phi_{out}) = (60°, 70°)$ do we see a hint of right-handed polarization in the red

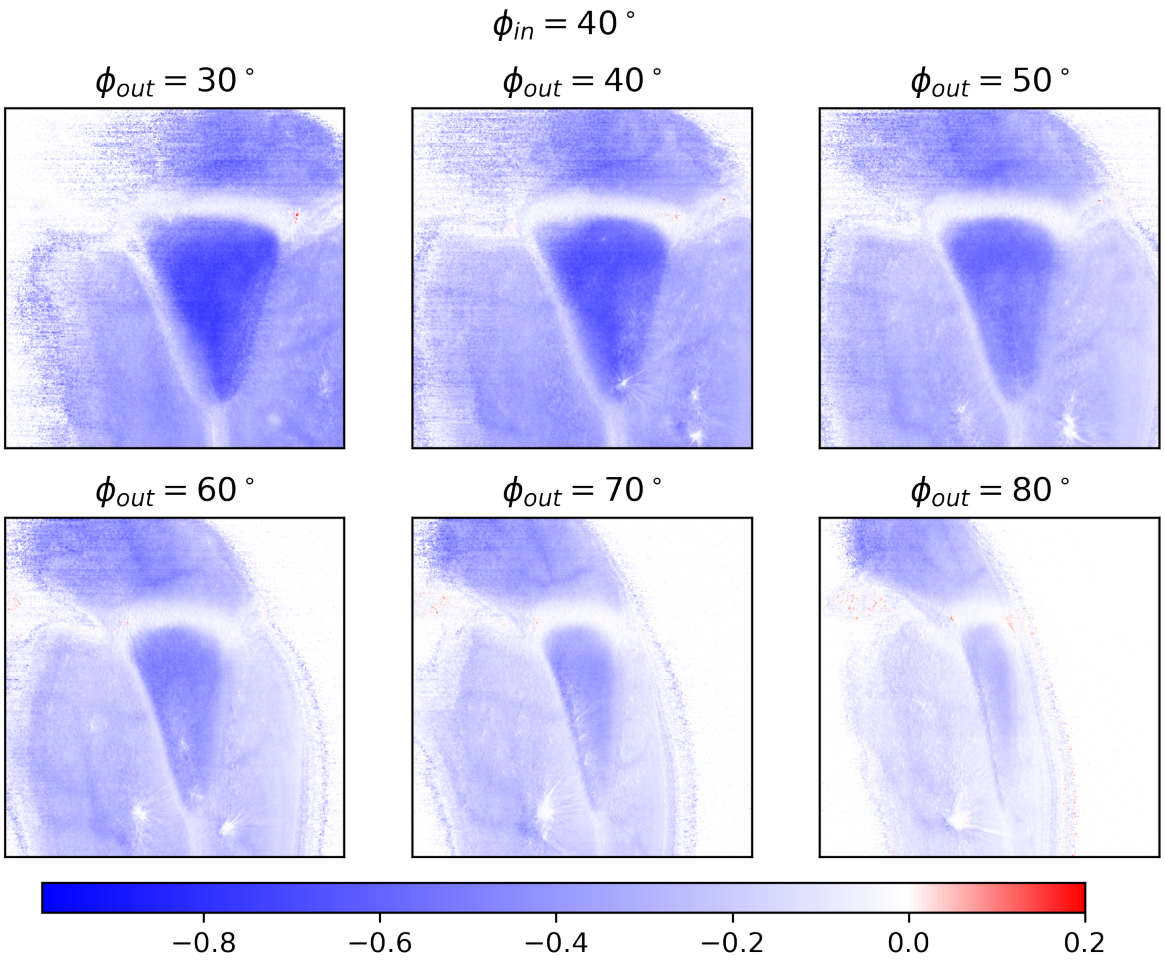

Figure 4.3: *Series of images of increasing value of ϕ_{out} at $\phi_{in} = 40°$. The color scale represents the magnitude and sign of Stokes parameter S_3.*

triangular area. The signal is faint but it is certainly there. This seems to contradict the result from figure 4.2. However, that measurement literally focused only on a small part of the beetle surface, while the last measurement is more 'broad-band'. Direct comparison is therefore challenging, especially since the object in question is highly curved. Nevertheless, because both experiments yield significant results, we feel like this project adds validity to earlier claims of handedness inversion in beetle species *jousellini*.

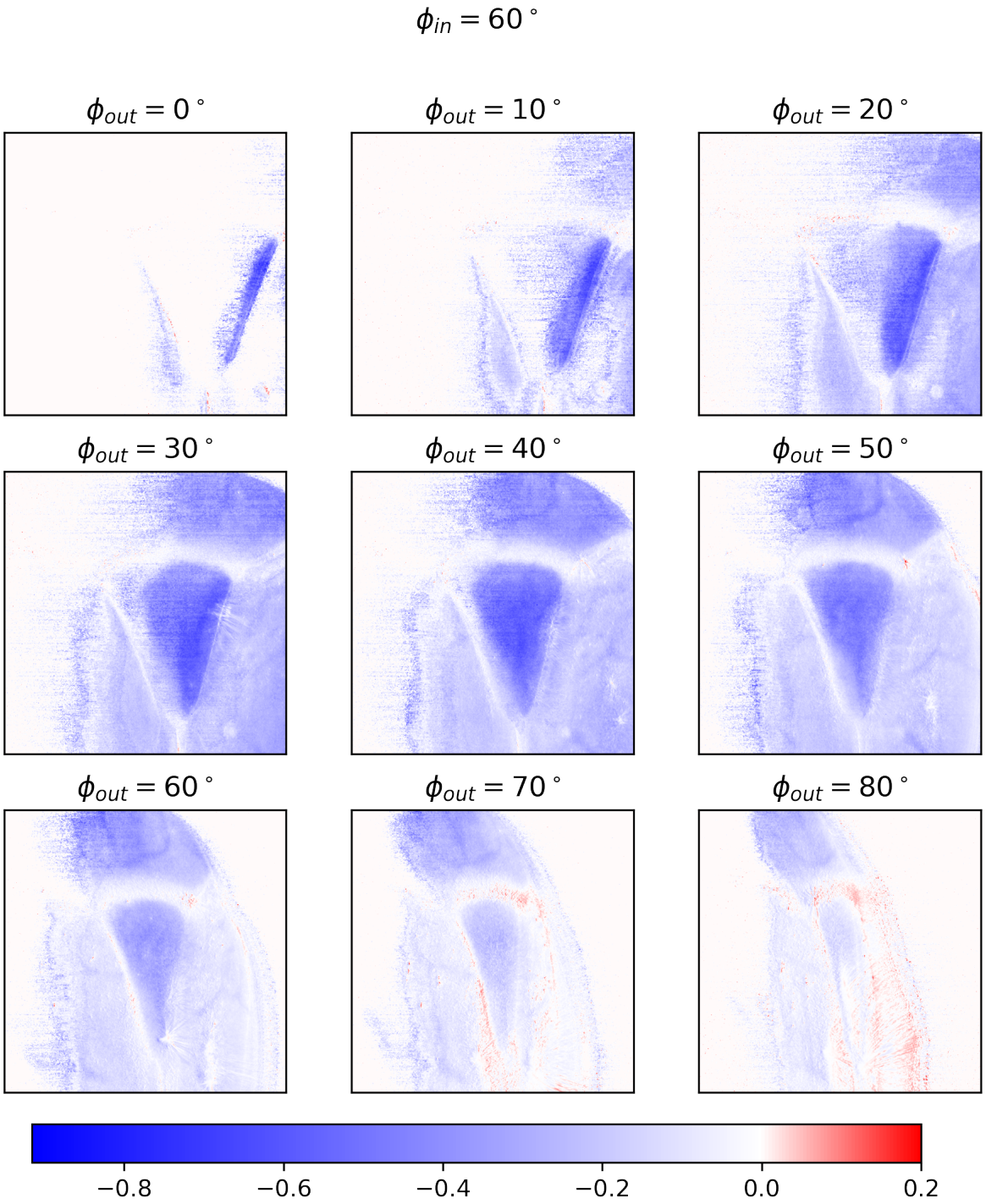

Figure 4.4: *Series of images of increasing value of ϕ_{out} at $\phi_{in} = 60°$. The color scale represents the magnitude and sign of Stokes parameter S_3.*

Chapter 5

Conclusion

At the start of this project we set out to see if Stokes parameters can be used as a classification scheme for coleoptera with a polarizing iridescence mechanism. To this end, we constructed a measurement setup capable of producing pixel-specific Stokes vectors of light reflected by an arbitrary specimen. Furthermore this setup has multiple rotational axes, which allow for control of the angle of incidence and angle of observation.

After analyzing light reflected by the beetle specimen with our setup, we conclude that Stokes parameters are not a suitable framework for classifying a animal species, as in practice the values of the four Stokes parameters are inhomogeneous on the surface of said animal and vary depending on lighting and observational perspective. Our measurement setup can however be used to investigate qualitatively which parts on a beetle surface contain a (circularly) polarizing iridescence mechanism.

An additional goal of this project was to verify earlier claims of handedness inversion in the beetle species *protaetia speciosa jousselini* at large angles of observaton an incidence. By observing a specimen of said beetle at multiple combinations of angles, we conclude that indeed the beetle shows this handedness inversion. We conclude that the handedness inversion occurs at angles of observation of at least 70° normal to the surface. However, we do not have enough evidence to to quantify if this phenomenon occurs for all or a subset of angles of incidence. Finally, we observed that the magnitude of the right-handed polarized light, in terms of Stokes parameter S_3, to be small compared to the magnitude of left-handed polarized light.

Chapter 6

Acknowledgements

Behind every student completing a project, there is a full cast of unsung heroes who contributed to the success of their achievements. From family and friends to teachers and professors, many added a brick to the metaphorical building that is the person who now writes this thesis. While I cannot thank everybody, I would like to hightlight a few.

First and foremost I would like to thank my supervisor Michiel de Dood for his support during this project. Not only did we have fruitful conversations about physics, Michiel more than tolerated my endeavours in politics that inadvertently (although in honesty unavoidably) delayed the completion of this research. I saw Michiel more than many a student see their formal supervisor during a project, for which I'm deeply grateful. Despite being director of education, Michiel always managed to find time to help me in the lab upon request.

Furthermore, I would like to thank my second corrector Tjerk Oosterkamp for taking the time to judge my work. He also provided encouraging words about the nature of experimental research and its inevitable frustrations about non-functioning equipment.

Finally, I would like to thank my partner Emmy Stevens for her linguistic support and valuable encouragement during this project. Without her teachings over the years, my English writing would have approximated Dunglish much more than it does now.

As I leave the masters program in physics by writing this thesis, the coronavirus pandemic seems to leave the daily lives of people in Europe. I am lucky to finish my degree and start working life in a time of relative normalcy, a privilege many of my friends did not have. I am grateful to have studied at a prestigious institution as Leiden University as a resident of one of the most prosperous nations on our planet, and I would like to use this final opportunity to acknowledge that.

Appendix A

Python Interface

Commanding all moving parts in the setup, performing measurements with the camera and analyzing the data was all done using various Python scripts. An object-oriented approach was used, where seperate parts of the measurement setup were coded as instances of various Python classes. This mainly provides benefits to the user experience as connections to physical devices can be initialized easily, and connections to multiple similar devices can be maintained by creating multiple instances of the same class. The measurement setup as a whole is controlled through a simple Jupyter Notebook pseudo-interface.

The Python project as a whole is lovingly dubbed *Beetle Classifier Robot* or BCR for short. The master file `BCR.py`, containing the `BCR` class, is the main script an experimenter interfaces with. Upon initialization, it connects to all necessary devices. Once connected, the user is free to control all devices individually. The main attraction however, are the different class functions representing automatic measurement sequences. In section A.1 we provide a step-by-step overview of running a experiment.

In section A.2 we dive deeper into the individual classes that were written for this project. We will also discuss hardware bugs that were discovered and attempts to circumvent them through software.

The entire script including documentation can be found on the author's GitHub repository.

A.1 Program Flow

In this section we will walk through the process of initializing the BCR script and performing an angle-dependent measurement of the Stokes parameters of a specimen. These steps are also described in the Jupyter Notebook `master.ipynb` in the root directory of the project.

First we import the `BCR` module and create an instance for our experiment.

```
from BCR import BCR
user = 'Naor Scheinowitz'
exp = BCR(user)
```

```
Welcome to the Beetle Classifier Robot. Great to have you back!
Resource manager used:     Resource Manager of Visa Library at C:\Windows\
    system32\visa32.dll
```

```
Detected devices:          ('ASRL1::INSTR', 'GPIB0::1::INSTR', 'GPIB0::2::INSTR
    ')
Initializing new connections with ESP with identifiers:      ['GPIB0::1', '
    GPIB0::2'].
Initializing connection with camera
```

With everything loaded, the user can interface directly with the different devices. The following commands rotate the specimen by 10 degrees, move the imaging stage to position 20 degrees, and take 5 images, respectively.

```
exp.sample.move(10,'relative')
exp.big_arm.move(20,'absolute')
exp.cam.take_images(5)
```

It is highly recommended to check the camera exposure settings before actually starting a measurement sequence. This can be done automatically:

```
exp.cam.auto_expose(E_start=0.01, target=220, margin=20)
```

The script will start taking images at exposure `E_Start` (in seconds) and adjust the exposure time until the brightest pixel has reached a value of `target`$\pm$`margin`. The exposure can also be set manually:

```
exp.cam.set_settings({'exposure': 0.1})
```

Once a satisfactory exposure is achieved, one can start a measurement sequence with one of the built-in functions of the `BCR` class.

```
readme = 'Test measurement'
exp.beetle_polarization(
    mode='create',
    angle_in=30,  # phi_in
    angles_out=[0, 90],  # bounds of phi_out
    step_size=10,  # step size of phi_out
    readme=readme,
    name='Test measurement',
    repeats=10
    )
```

```
Working directory changed to: N:\Beetle Project\Beetle Classifier Robot\
    Experiments\
Created file Test measurement.hdf5
Created HDF5 group 30 degree reflection
Starting measurement.
100%|----------------------------------| 10/10 [16:28:55<00:00, 5933.51s/it]
Measurement sequence completed!
Closed file Test measurement.hdf5.
```

An important note is that any amount of metadata that is passed with the `metadata` keyword argument is saved to the HDF5 data file of this experiment. It is advised to write important information either there or in the `readme` field.

A.2 Custom Classes

The goal of this section is twofold: firstly we will give an overview of the most important functions of each of the written Python classes. Secondly, we will discuss which hardware

bugs we encountered and how we attempted to write code to circumvent these bugs. Each Python class was written to include the boolean parameter `testflight` during initialization. When set to `True`, the code can be run without having any of the physical devices actually connected. Some code snippets will be shown to illustrate how problems were tackled but to save space only the relevant parts will be presented. Please refer to the actual Python files to see the full code.

A.2.1 Camera

We used a ThorLabs DCC1645C* CMOS camera for our measurements due to its compactness and USB interface. Controlling the camera is done through the `Cam` class, which combines features from the pylablib package with custom functions specific to the scope of this project.

Connecting and Disconnecting

Connecting to the camera is done by initializing an instance of the `Cam` class. Camera settings can be passed upon initialization in dictionary form.

```
cam = Cam(settings={})
```

Any settings not passed will be reset to their default value. If the script exits without properly disconnecting the camera, it cannot be reconnected to without manually removing and reattaching the USB cable. To ensure proper disconnection upon script exit, without human intervention, a `close()` command is added in the deletion dunder.

```
def __del__(self):
    self.instrument.close()
```

Changing Camera Settings

In changing the camera settings we encounter our first series of hardware bugs. Some settings seems to reset one another, without any seemingly logical reason behind it. For example, changing the ROI (Region Of Interest) of the sensor sometimes, but not always, resets the exposure time. We also found that when increasing the exposure time, it is useful to change frame period first, while the opposite order is more reliable when decreasing the exposure time. This is because the camera often (but not always) refuses exposure times longer than the set frame period. It would make sense that the frame period scale automatically with the exposure time, but such luxuries are not reserved voor scientists. To combat these bugs, the function that changes settings prints all camera settings after a change attempt by default. This allows the user to verify if the change was successful and no other settings were altered.

Another bug that we encountered is that the first few images taken after a change in exposure time still use the old time despite reporting otherwise. The simple solution to this bug is to take number of images after each settings change and throw them away immediately.

```
def set_settings(self, settings={}, test_img=True):
    if type(settings) is not dict:
        raise TypeError('Variable "settings" should be of type dict.')
    s = self.instrument.apply_settings(settings)
    if test_img:
        self.take_images(nframes=1, show=False)
```

*ThorLabs documentation refers to this class of camera as a 'DCx' camera, but it is equivalent to the classification 'UC480', which occasionally shows up in other documentation.

Taking Pictures

Managing to reliably take pictures with the camera has been one of the most challenging aspects of this research project. One would expect that taking an image would be as simple as (1) sending a command to take an image and (2) reading out that image. Unfortunately, that was not the case. Using that naïve approach, one would often, but not always, receive an image where a proportion of rows were devoid of pixel values. In other words, it seems like the camera self-interrupts exposures or read-outs at random, even when manually commanding the script to wait until a full frame is available.

After many attempts at narrowing down the origin of the problem, we decided to circumvent the problem instead of solving it. We found that the problem does not occur as frequently when the camera is commanded to take multiple images before reading out to system memory. The camera has an internal image buffer that the full frames are then temporarily saved to.

This multi-frame approach did have its own downside: sometimes the camera takes a bit longer to process the images then the set frame period, which would lead to time-out errors. Luckily, this problem was easily solved by manually setting a large maximum time-out.

Despite these efforts, the 'partial-frame bug' would now and then appear. As a final mitigatory measure, a user can choose to take a per-pixel median through a stack of frames. A single ruined frame in a stack will not factor in heavily when taking a median, making it a better operation than taking a mean.

Finally, the user can choose to immediately plot the image taken by camera to check if the image is satisfactory and if any pixels are overexposed.

```
def take_images(self, nframes=20, median=True, show=True):
    frame_period = self.instrument.get_frame_timings()[1]
    max_TO = frame_period * 2 + 1.0
    self.instrument.start_acquisition()
    self.instrument.wait_for_frame(
        nframes = nframes, timeout=(max_TO * nframes, max_TO))
    img = self.instrument.read_multiple_images()
    self.instrument.stop_acquisition()

    img = np.array(img)

    if median and nframes > 1:
        img = np.median(img, axis=0)

    if show:
        # Plotting functions...
    return img
```

A.2.2 Motion Controller

The specimen-holder, imaging stage, linear polarizer and quarter-wave plate used during the project were all controlled using two Newport ESP300 motion controllers. Despite their age, the motion controllers proved very reliable (with a few caveats, more on that later). Interfacing with the motion controllers is done through a GPIB to USB connector. A neat feature of GPIB is that many devices can be daisy-chained together with clever multi-input GPIB-to-GPIB connectors. Only a single motion controller is connected to a measurement PC with a GPIB-to-USB cable, but the PC is able to address all devices through device-specific identifiers.

The motion controllers can receive commands and send status messages through a ASCII input bus. Managing this bus was done using the PyVisa Python package.

On top of the functionality provided by PyVisa, two Python classes were written to control the motion controllers: `ESP` and `Motor`. The former handles connections to a physical controller, while the latter represents individual motors. Users will primarily interface with the `Motor` class as it contains all the functions for moving the measurement stages.

Connecting to Devices

The way the code is setup, all motion controllers are represented by a single instance of the `ESP` class, while each separate motor is represented by separate instances of the `Motor` class. The procedure for setting everything up is as follows. First create a `ESP` object:

```
controller = ESP.ESP(identifiers=['GPIB0::1', 'GPIB0::2'], name='ESP')
```

The identifiers that are passed are the GPIB addresses of the connected motion controllers. This class instance contains the class variable `instruments`, which is a list of PyVisa instances (one for each physical device). To initialize a specific motor, the corresponding PyVisa instance must be passed to the `Motor` class. For example, the imaging stage is connected to the first axis of the motion controller that is second on the list of addresses, so the correct syntax to connect to it is:

```
imaging_stage = ESP.Motor(controller.instruments[1], axis=1, bounds=[-35, 100],
    velocity=10, name="imaging_stage")
```

The argument `bounds` defines the maximum positions the stage can rotate to. This can be useful if collisions between stages are a risk. The `velocity` argument sets the angular velocity in degrees per second.

Unlike the camera, properly disconnecting is less of a concern with the motion controllers. Nevertheless, the deletion dunder of `ESP` also includes the PyVisa function to close the connection.

Moving the Stages

Rotating the different connected stages can be done with a single class function: `Motor.move()`. For the example stage in the previous section, the following command will first move it to position -20, after which it will move +10 degrees from there.

```
imaging_stage.move(-20, 'absolute')
imaging_stage.move(10, 'relative')
```

```
Moving axis 1 to position -20 degrees.
Moving axis 1 to position -10 degrees.
```

If a command is sent to a stage that would move it beyond its bounds, the movement is cancelled.

```
imaging_stage.move(300, 'absolute')
```

```
Desired position 300 is out of operating bounds of axis 1.
```

Sending Commands and Error Handling

The motion controller can interpret a wide range of commands, each encoded with two ASCII characters. The syntax for a command is `aaXXbbb`, which encodes axis number, command, and optional value, in that order. For example, the command to move to an absolute position is `PA`, so the first command of the previous section would be encoded as `1PA-20`. Commands can be strung together in a single line by separating them with a semicolon. For example, to command the controller to move an axis and then wait until the motion has stopped, one could send the command `1PA-20;1WS0`.

When a string containing commands is sent to the controller, all commands up to a newline character will be executed immediately. This means one must be careful which commands are sent after one another. If one sends the command to ask for an error code while a motion is still under way, the result is unpleasant. To prevent these scenarios, the generic `Motor.send_command` function is written in such a way that every command is followed by a wait command and terminated by a error query. If the query returns a non-zero value, something has gone wrong and the user is informed. The downside of this approach is that stages cannot move in parallel, but this is a small price to pay for the enhanced reliability.

Common functions and error codes are saved as class constants in dictionary form so that a user need not memorize two-character commands or two-digit error codes. This transforms the command-line interface from:

```
imaging_stage.send_ASCII_command('1VA10')
```

```
Error code 13 on device imaging_stage.
```

to a more readable:

```
imaging_stage.send_command('set velocity', 10)
```

```
Error code 13: MOTOR NOT ENABLED on device imaging_stage.
```

Caveats

The main caveat of the Newport ESP300 motion controllers is its age. At unpredictable moments, one of the controllers will give a time-out error. Our best guess to the origin of these errors is that the communication from speed modern PC is so much greater than what the controllers were designed for in the nineties that they occasionally get overloaded. But at this point that is pure conjecture.

The time-out errors completely halt the execution of a measurement sequence, making them extremely inconvenient. They were reduced by adding manual delays in the class function that sends commands to the controller, `Motor.send_command`, but no amount of delays prevented these errors completely.

Like we did with the Thorlabs camera, we circumvented the problem instead of solving it. This time we wrote a simple try/except statement that will re-send a command up to three times. Since implementing this statement, no measurement-interrupting time-out errors have occurred.

The command flow follows the following order: `move -> send_command -> write_command`, with the order going from high to low level. The most important code snippets of these three commands are shown below.

```
def move(self, degrees, mode='relative'):
    if mode == 'absolute':
        new_position = degrees
    elif mode == 'relative':
        new_position = self.get_current_position() + degrees

    allowed = self.bounds[0] <= new_position <= self.bounds[1]
    if allowed:
        self.send_command('move to absolute position', new_position)
    else:
        print(f'Desired position {new_position} is out of operating bounds of
    axis {self.axis}.')

def send_command(self, command, parameter=''):
    time.sleep(1)
    full_command = f'{self.axis}{self.COMMANDS[command]}{parameter}'
    if 'move' in command:
        wait = f'{self.axis}{self.COMMANDS["wait for motion stop"]}0'
        full_command += ';' + wait

    self.write_command(full_command)
    err = self.query_command(self.COMMANDS['read error code'])
    # remove first number from error code if the first number is the
    # same as the axis number.
    if err[0] == str(self.axis) and len(err) > 2:
        err = err[1:]
    if err != '0':
        if err in self.ERROR_CODES:
            print(
                f'Error code {err}: {self.ERROR_CODES[err]} on device '
                f'"{self.name}" while executing command '
                f'"{full_command}".')
        else:
            print(
                f'Error code {err} on device {self.name}.')
        return False
    else:
        return True

def write_command(self, command, attempt=1):
    try:
        self.instrument.write(command)
    except pyvisa.VisaIOError:
        print(
            f'Got a VisaIOError on device "{self.name}" while '
            f' executing command "{command}" (attempt {attempt}).')
        if attempt < 3:
            self.write_command(command, attempt=attempt + 1)
        else:
            print('Three attempts failed. Will continue with '
                  'next command.')
            return 'VisaIOError'
```

A.2.3 HDF5

As described in chapter 3.1.2, we used the HDF5 data format during this project. The two main benefits of this format for us are the flexible way of saving metadata and automatic data compression. A drawback of HDF5 is that is not the most intuitive format to use. To make life a bit easier, often used code snippets were combined into a custom `HDF5` class.

Creating New Files

The first thing to note is that HDf5 files are very fragile while opened in 'write mode'. A python script crash or copying the file while open can be enough to completely corrupt the embedded data. Caution is certainly advised. We therefore strongly advise to only use the custom class with the with/as syntax.

HDF5 uses a folder-like internal structure where 'groups' and 'datasets' are the analogues of folders and files. Each object in the hierarchy can have metadata assigned to it. With the custom `HDF5` class, creating a new file and filling it with data can be done as follows:

```
# Create some (meta)data
data = np.random.randint(0, 255, (500,500))
metadata = {'meta1': 5, 'meta2': [1, 2, 3]}
with HDF5('test.hdf5', 'create', user='Naor Scheinowitz', date=True) as f:
    f.group('test group')
    dset = f.create_dataset('test set', data, parent='test group', metadata=
    metadata)
    HDF5.read_metadata(dset, print=True)
```

```
Created file 2022.04.12 test.hdf5
Created HDF5 group test group

meta1:  5
meta2:  [1 2 3]

Closed file 2022.04.12 test.hdf5.
```

Viewing and Editing Metadata

The beauty of saving metadat cannot be overstated. It is incredibly useful for data analysis that all relevant information is bundled with the raw data itself. Using the custom functions `read_metadata` and `print_structure`, the metadata can be easily viewed.

```
with HDF5('test.hdf5', 'read only') as f:
    f.print_structure()
    dset = f.file['40 degree reflection/Frames 0']
    HDF5.read_metadata(dset, True)
```

```
user:  Naor Scheinowitz

Groups:
40 degree reflection
60 degree reflection

Datasets of group "40 degree reflection":
Frames 0

Metadata of group "40 degree reflection":
angle_in:  40
angles_out:  [ 0 80]
date:  2022.02.23
median:  True
nframes:  3
repeats:  5
software_verion:  0.8
step_size:  10
```

```
angles_out:  [20 30 40 50 60 70 80]
color_mode:  rgb8p
exposure:  9.99915
frame_period:  9.9997492
gains:  [1. 1. 1. 1.]
pixel_rate:  5000000.0
polarizer_angles:  [  0.  12.  24.  36.  48.  60.  72.  84.  96. 108. 120. 132.
    144. 156.
 168. 180.]
roi:  [   0 1280    0 1024    1    1]
```

Using the `read_metadata` and `write_metadata` functions, metadata can easily be copied between files. This can be useful when creating a new file with corrected data from a raw data file.

```
with HDF5('raw.hdf5', 'read only') as raw:
    with HDF5('corrected.hdf5', 'open') as f:
        for dset in raw.file:
            metadata = HDF5.read_metadata(raw.file[dset])
            HDF5.write_metadata(f.file[dset], metadata)
```

Appending Datasets

HDF5 can be very rigid once datasets have been created. In fact, upon creation, the dimensions of a dataset are fixed. This implies that during measurement runs, each piece of new data must be saved as a separate dataset, or the entire dataset should be stored in system memory until the measurement terminates. Both not attractive options.

Luckily, there is a loophole: upon creating a dataset, at maximum one axis can have an underfined size. This opens the door to appending data to an existing dataset. The following function achieves such functionality.

```
def append_dataset(dataset, new_data):
    current = dataset.shape[0]
    new = new_data.shape[0]
    total = current + new
    dataset.resize(total, axis=0)
    dataset[current:total] = new_data
```

Bibliography

[1] A. E. Seago, P. Brady, J.-P. Vigneron, and T. D. Schultz, *Gold bugs and beyond: a review of iridescence and structural colour mechanisms in beetles (Coleoptera)*, Journal of The Royal Society Interface **6** (2009).

[2] B. G. Ferguson and J. L. Cleary, *In situ source level and source position estimates of biological transient signals produced by snapping shrimp in an underwater environment*, The Journal of the Acoustical Society of America **109**, 3031 (2001).

[3] J. Sun, B. Bhushan, and J. Tong, *Structural coloration in nature*, RSC Advances **3**, 14862 (2013).

[4] T. Hunt, J. Bergsten, Z. Levkanicova, A. Papadopoulou, O. S. John, R. Wild, P. M. Hammond, D. Ahrens, M. Balke, M. S. Caterino, J. Gómez-Zurita, I. Ribera, T. G. Barraclough, M. Bocakova, L. Bocak, and A. P. Vogler, *A Comprehensive Phylogeny of Beetles Reveals the Evolutionary Origins of a Superradiation*, Science (2007).

[5] None, *Global biodiversity : status of the earth's living resources : a report*, London ; New York : Chapman & Hall, 1992.

[6] G. Horváth, M. Blahó, A. Egri, R. Hegedüs, and G. Szél, *Circular Polarization Vision of Scarab Beetles*, in *Polarized Light and Polarization Vision in Animal Sciences*, edited by G. HorvÃ¡th, pages 147–170, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[7] J. D. Pye, *The distribution of circularly polarized light reflection in the Scarabaeoidea (Coleoptera)*, Biological Journal of the Linnean Society **100**, 585 (2010).

[8] I. J. Hodgkinson, M. W. McCall, and Q. Wu, *Birefringent thin films and polarizing elements: Martin W. McCall, Imperial College London, UK, Ian J. Hodgkinson, University of Otago, New Zealand, Qihong Wu, Finisar Australia, Australia*, Imperial College Press, London, 2nd edition edition, 2015.

[9] R. Hegedüs, G. Szél, and G. Horváth, *Imaging polarimetry of the circularly polarizing cuticle of scarab beetles (Coleoptera: Rutelidae, Cetoniidae)*, Vision Research **46**, 2786 (2006).

[10] E. J. Warrant, *Polarisation Vision: Beetles See Circularly Polarised Light*, Current Biology **20**, R610 (2010).

[11] T.-H. Chiou, S. Kleinlogel, T. Cronin, R. Caldwell, B. Loeffler, A. Siddiqi, A. Goldizen, and J. Marshall, *Circular Polarization Vision in a Stomatopod Crustacean*, Current Biology **18**, 429 (2008).

[12] J. Marshall and T. W. Cronin, *Polarisation vision*, Current Biology **21**, R101 (2011).

[13] M. Schilthuizen and A. Davison, *The convoluted evolution of snail chirality*, Naturwissenschaften **92**, 504 (2005).

[14] H. Peeters and K. Devriendt, *Human laterality disorders*, European Journal of Medical Genetics **49**, 349 (2006).

[15] T. Reisinger, *Investigating the Iridescence and Polarisation Properties of Butterflies and Beetles*, Bachelor Thesis, Leiden University, 2020.

[16] S. Richtberg and R. Girwidz, *Use of Linear and Circular Polarization: The Secret LCD Screen and 3D Cinema*, The Physics Teacher **55**, 406 (2017).

[17] H. Hong, J. Jang, D. Lee, M. Lim, and H. Shin, *Analysis of angular dependence of 3-D technology using polarized eyeglasses*, Journal of the Society for Information Display **18**, 8 (2010), _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1889/JSID18.1.8.

[18] B. Schaefer, E. Collett, R. Smyth, D. Barrett, and B. Fraher, *Measuring the Stokes polarization parameters*, Am. J. Phys. **75**, 6 (2015).

[19] E. Collett, *Field Guide to Polarization*, SPIE, 2005.

[20] H. Ku, *Notes on the use of propagation of error formulas*, Journal of Research of the National Bureau of Standards, Section C: Engineering and Instrumentation **70C**, 263 (1966).