



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

**Polarímetro de Müller para la
detección de actividad
eléctrica neuronal**

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Cristopher Antonio Navarrete Acosta

DIRECTOR DE TESIS

Dr. Francisco Rafael Fernández de Miguel



Ciudad Universitaria, Cd. Mx., 2025



**PROTESTA UNIVERSITARIA DE INTEGRIDAD Y
HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)**



De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado POLARIMETRO DE MÜLLER PARA LA DETECCIÓN DE ACTIVIDAD ELÉCTRICA NEURONAL que presenté para obtener el título de INGENIERO EN COMPUTACIÓN es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

CRISTOPHER ANTONIO NAVARRETE ACOSTA
Número de cuenta: 418048393

Resumen

La detección óptica no invasiva de actividad eléctrica neuronal permitiría estudiar redes celulares sin alterar su fisiología, superando limitaciones de los electrodos. En esta tesis se implementó un polarímetro de Müller de doble retardador rotador, enfocado en registrar actividad eléctrica en imágenes mediante las propiedades polarimétricas neuronales y los cambios que sufren al ser estimuladas. Las mediciones se basan en controlar la polarización de entrada y medir la polarización a la salida. El instrumento realiza 36 mediciones de intensidad de cada momento, con distintos estados de polarización de entrada y salida. A partir de ellos se aproxima, por mínimos cuadrados, la matriz de Müller asociada a la muestra. Esta matriz se refina para mitigar los errores inherentes al método experimental. De la matriz corregida se extraen tres magnitudes ópticas: la **diatenuación**, la **retardancia** y la **depolarización**. De cada una se produce una imagen en la que cada pixel expresa la propiedad óptica de la muestra. Para probar el polarímetro se realizaron tres experimentos, en los cuales se adaptaron el protocolo de adquisición y el diseño experimental en diferentes preparaciones biológicas. Usamos: 1) Cortes histológicos, 2) neuronas fijadas antes y después de estimularlas eléctricamente y 3) neuronas aisladas con registro eléctrico intracelular *in vivo*. Los resultados mostraron que el polarímetro tiene una respuesta adecuada para muestras fijadas, ya sean las muestras de referencia o cortes de músculo, que mostraron claras diferencias entre el tejido sano y necrosado. Sin embargo, aún es poco preciso al medir muestras dinámicas (vivas) y en particular para detectar la actividad eléctrica, debido a un problema mecánico. Esta limitación podría mejorar utilizando un dispositivo óptico que pueda producir estas configuraciones a mayor velocidad sin necesidad de rotaciones mecánicas.

Índice general

1. Introducción	1
2. Fundamentos teóricos de polarización	7
2.1. Polarización de la luz	7
2.2. Formalismo de Stokes-Müller	17
3. Diseño del polarímetro de Müller	27
3.1. Polarímetro de doble retardador rotador (DRR)	27
3.2. Filtrado de los resultados experimentales	34
3.3. Interpretación de la matriz de Müller	38
3.3.1. Algoritmo de Lu y Chipman	40
4. Implementación experimental del polarímetro	45
4.1. Hardware	45
4.2. Software	47
4.2.1. Control de la cámara	48
4.2.2. Procesamiento de las 36 imágenes	48
4.3. Integración del sistema	53
5. Resultados	57
5.1. Pruebas. resultados teóricos vs. experimentales	57
5.1.1. Visualizaciones gráficas	63

ÍNDICE GENERAL

5.2. Experimentos con muestras biológicas	69
5.2.1. Experimento 1. Cortes de músculo	69
5.2.2. Experimento 2. Neuronas fijadas pre y post-estimulación	74
5.2.3. Experimento 3. Neuronas vivas	79
6. Conclusiones	83
A. Código fuente	85

Índice de figuras

1.1. Primer potencial de acción registrado	1
2.1. Campo óptico y eje de propagación	9
2.2. Polarizaciones lineales	11
2.3. Polarización circular	12
2.4. Polarización elíptica.	12
2.5. Elipse de polarización.	14
2.6. Interacción de un haz de luz con un medio.	19
2.7. Matriz de Müller equivalente a varios elementos en serie.	24
3.1. Arreglo de un polarímetro de Müller en transmisión.	28
3.2. Disposición de elementos ópticos del polarímetro	30
4.1. Diagrama y fotografía de la configuración experimental del polarímetro.	46
4.2. Modelo 3D del polarímetro. Vista axonométrica.	47
4.3. Diagrama de flujo de la operación del polarímetro.	49
4.4. Interfaz gráfica del software	51
4.5. Secuencia de acciones de los actuadores	55
4.6. Diagrama de flujo del programa que se ejecuta en Arduino	56
5.1. Modelado del polarímetro para la calibración	59
5.2. Problema de frontera de los polarizadores lineales	62
5.3. Matrices de Müller del espacio libre	63

ÍNDICE DE FIGURAS

5.4. Imágenes de las propiedades ópticas de $\mathbf{M}_{c0}$	64
5.5. Histogramas de las imágenes de las propiedades ópticas de $\mathbf{M}_{c0}$	64
5.6. Matrices de Müller del polarizador lineal horizontal (0°)	65
5.7. Imágenes de las propiedades ópticas de $\mathbf{M}_{c1}$	66
5.8. Histogramas de las imágenes de las propiedades ópticas de $\mathbf{M}_{c1}$	66
5.9. Matrices de Müller del polarizador lineal vertical (90°)	67
5.10. Imágenes de las propiedades ópticas de $\mathbf{M}_{c2}$	67
5.11. Histogramas de las imágenes de las propiedades ópticas de $\mathbf{M}_{c2}$	67
5.12. Matrices de Müller del retardador lineal $\lambda/4$ a 30°	68
5.13. Imágenes de las propiedades ópticas de $\mathbf{M}_{c3}$	68
5.14. Histogramas de las imágenes de las propiedades ópticas de $\mathbf{M}_{c3}$	69
5.15. Cortes histológicos. Nomarski	70
5.16. Propiedades ópticas en B/N de los cortes de músculo	71
5.17. Propiedades ópticas en pseudocolor de los cortes de músculo	72
5.18. Imágenes compuestas en RGB de los cortes de músculo	73
5.19. Propiedades ópticas de las neuronas control. B/N	75
5.20. Propiedades ópticas de las neuronas estimuladas. B/N	76
5.21. Propiedades ópticas de las neuronas control. RGB	77
5.22. propiedades ópticas de las neuronas estimuladas. RGB	78
5.23. Propiedades ópticas de las neuronas control vs estimuladas	78
5.24. Secuencia temporal de las propiedades ópticas de una neurona	80
5.25. Fotografía de la neurona con electrodo intracelular	81

Índice de tablas

2.1. Vectores de Stokes	20
5.1. Matrices de Müller y propiedades ópticas teóricas. Muestras de referencia	58
5.2. Matrices de Müller experimentales. Muestras de referencia	60
5.3. Propiedades ópticas experimentales. Muestras de referencia	61
5.4. Solución de Ringer y solución con alto potasio	75

ÍNDICE DE TABLAS

Capítulo 1

Introducción

El sistema nervioso se comunica mediante impulsos eléctricos breves, de apenas 1 o 2 milisegundos, conocidos como potenciales de acción. [1, 2] La generación de estos impulsos es un movimiento repentino y opuesto de iones sodio y potasio a través de la membrana celular, debido al aumento súbito en la permeabilidad de esta, primero al sodio que entra a la neurona y después al potasio que sale de ella. [3]

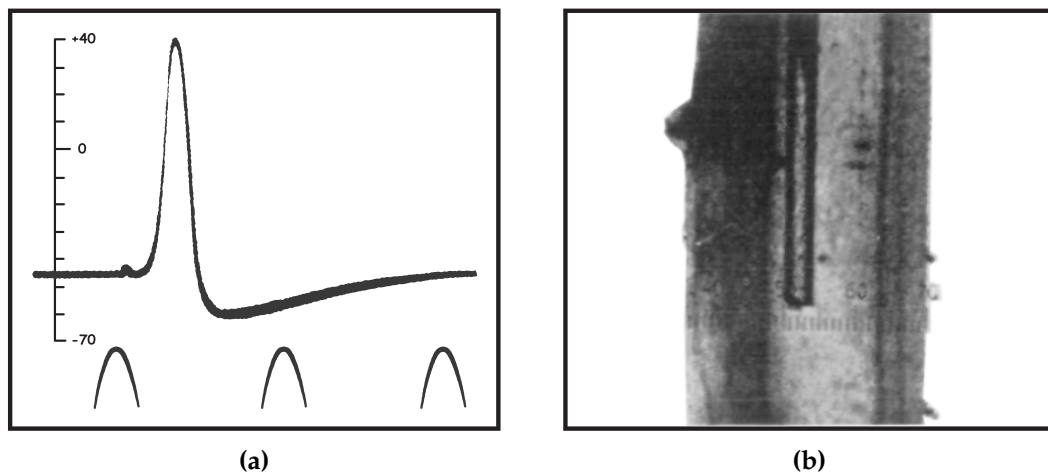


Figura 1.1: a) Primer potencial de acción registrado del interior del axón gigante de calamar *loligo forbesii*. Marcador de tiempo, 500 ciclos/seg. El electrodo interno detectó en milivoltios; el agua de mar del exterior se toma como referencia (GND). b) Fotografía del electrodo dentro del axón gigante. 1 división de la escala = $33\mu\text{m}$. Imágenes tomadas y modificadas de [1]

CAPÍTULO 1. INTRODUCCIÓN

Cuando colocamos dos electrodos, uno en el interior de una neurona y otro en el exterior como referencia, la diferencia entre ellos muestra un voltaje interno negativo, de aproximadamente -50 [mV]. Esta diferencia de potencial eléctrico se llama potencial de membrana o de reposo. El potencial de reposo depende de los gradientes electroquímicos de iones entre el interior y el exterior de la membrana. [3]

La membrana de las células excitables, en reposo, es resistente al flujo de corriente eléctrica debido a la baja permeabilidad iónica. Durante el potencial de acción, la apertura de canales de sodio y potasio dependientes de voltaje aumenta drásticamente su permeabilidad, permitiendo el flujo de iones según sus gradientes electroquímicos.

La medición del potencial de reposo o potenciales de acción requiere registrar del interior de la célula, ello permite un seguimiento muy claro de la actividad eléctrica. Sin embargo se complica cuando se pretende registrar la actividad de varias neuronas en conjunto. Durante el potencial de acción, los cambios en el flujo iónico modifican la organización estructural y las propiedades ópticas de la membrana y del citoesqueleto, lo cual puede detectarse mediante el uso de luz polarizada. En esta tesis se propone el uso de una técnica no invasiva para detectar la actividad eléctrica midiendo el nivel de intensidad de luz polarizada con el que se ilumina la muestra. Esto tendría la ventaja de no dañar las células y poder emplear un sensor fotográfico para filmar la actividad eléctrica de la muestra, que puede ser un circuito completo. Como primera aproximación, aquí trabajamos con neuronas aisladas en las que podemos controlar los niveles de actividad eléctrica.

Mediciones de cambios ópticos debidos a la actividad eléctrica

Los primeros antecedentes de cambios ópticos observados, producidos por la actividad eléctrica en células nerviosas fueron dos. El primero, reportado por Richard Keynes en 1948, mostró un cambio *lento y retrasado* de la opacidad de axones debido a un estímulo eléctrico inducido [4]. Tiempo después, en 1968 y 1970, Keynes en conjunto con Law-

rence Cohen y Bertil Hille muestran un cambio *rápido* en la birrefringencia ¹ del nervio de calamar, que correlaciona temporalmente al potencial de acción medido con electrodos. [5,6] Estudios posteriores se reportan en [7–10]. Estas investigaciones, por un lado, abrieron el estudio de las propiedades ópticas usando compuestos fotoquímicos, y por otro dejó pendiente el uso de las propiedades polarimétricas como reporteras potenciales de los niveles de actividad eléctrica. Para entender estos conceptos, es fundamental comprender los estados de polarización y los cambios que sufren al interactuar con la materia.

Polarización de la luz

Un haz de luz que proviene del sol y tiene su eje de transmisión, vector que indica la dirección del haz, hacia nosotros, contiene fotones vibrando en direcciones aleatorias. Al interactuar con un objeto, por transmisión o reflexión, estos fotones pueden cambiar su dirección de vibración y *ordenarse*, o polarizarse en el haz de salida, que adquiere un estado de polarización distinto al del haz que incidió en el objeto. Al material que logra esto se le llama polarizador.

La luz polarizada tiene varias formas de representación, una de las cuales es de especial importancia experimental: el vector de Stokes. Nombrado así en honor a quien lo propuso, George Gabriel Stokes (1819-1903). El vector de Stokes se compone de cuatro magnitudes de intensidad observables y medibles. Estas características le dieron relevancia para aplicaciones en instrumentación. El vector de Stokes nos permite representar cualquier estado de polarización y su *grado de polarización*, que representa la proporción del haz que se encuentra polarizado. Estos conceptos se explicarán con más detalle en el Capítulo 1. El vector de Stokes nos permite modelar la interacción de los haces de luz polarizados con objetos que cambian este estado de polarización, mediante una multiplicación matricial. Para que la multiplicación nos dé como resultado un vec-

¹Propiedad de algunos materiales de descomponer un haz de luz en dos componentes con velocidades distintas, polarizadas linealmente y perpendiculares entre sí

CAPÍTULO 1. INTRODUCCIÓN

tor de Stokes, de tamaño 4×1 , se debe multiplicar con una matriz real de tamaño 4×4 , que además debe de cumplir algunas restricciones físicas. Esta es la llamada matriz de Müller, dando crédito al físico Hans Müller (1900-1965) que desarrolló el formalismo algebraico que nos permite trabajar con estas matrices. La matriz de Müller contiene la información polarimétrica de una muestra que relaciona el haz de luz incidente con el haz de luz de salida.

El polarímetro de Müller

Un polarímetro es un instrumento capaz de medir los elementos de un vector de Stokes o una Matriz de Müller. El polarímetro de Müller nos permite calcular los 16 elementos de la matriz de Müller (4×4) de una muestra. El principio de funcionamiento, lo propuso Azzam en 1978 [11]. El polarímetro se compone de 5 elementos principales. En orden de aparición: 1) fuente de luz no coherente, 2) generador de estados de polarización (PSG), 3) muestra, 4) analizador de estados de polarización (PSA) y 5) detector de intensidad. Al generar diferentes estados de polarización en el PSG y PSA rotando dos retardadores de $\lambda/4$, obtenemos una señal modulada en intensidad, que contiene la información polarimétrica de la muestra. Esta relación se puede encontrar en los coeficientes en series de Fourier de ambas señales.

Para poner a punto el polarímetro y antes de los experimentos con muestras biológicas, se hacen pruebas con elementos de los cuales se conoce su matriz de Müller teórica [12, 13]. Las cuatro muestras de referencia por lo general son: un polarizador lineal horizontal, un polarizador lineal vertical, un retardador de $\lambda/4$ rotado 30° y el espacio libre (sin muestra). Los datos obtenidos en esta etapa sirven para caracterizar el instrumento y comprobar la fiabilidad de estos comparándolos con los resultados teóricos.

Alcances del trabajo

Esta tesis cubre la teoría, técnicas, procedimientos, integración, programación, construcción y pruebas del polarímetro de Müller, necesarios para comprender, utilizar y/o replicar este instrumento. Es la base para poder desarrollar una técnica prometedora para la medición de actividad eléctrica en el sistema nervioso que permita hacer registros *in vivo*. Durante el desarrollo de este trabajo de tesis logramos implementar un polarímetro, la técnica y la práctica para la obtención de imágenes de las propiedades ópticas, además del posterior análisis y posible interpretación de los resultados.

CAPÍTULO 1. INTRODUCCIÓN

Capítulo 2

Fundamentos teóricos de polarización

En este capítulo se describen los fundamentos teóricos para la construcción de un instrumento capaz de medir las propiedades polarimétricas de una muestra, conocido como Polarímetro de Müller. La comprensión de la polarización de la luz mediante la ecuación conocida como *Elipse de Polarización* será el objetivo de la primera sección. A continuación se analiza el formalismo de Stokes-Müller, una metodología que nos permite representar los complejos datos obtenidos a través de magnitudes físicas medibles como la intensidad luminosa.

2.1. Polarización de la luz

La polarización es una propiedad fundamental de las ondas electromagnéticas transversales, o sea, aquellas que vibran en dirección perpendicular al eje de propagación). El término polarización se refiere a la dirección de oscilación del campo eléctrico de la onda electromagnética que viaja a través de un medio [14]. Se explicará esta propiedad empleando la ecuación de una elipse, que permite definir cualquier estado de polarización en un haz de luz.

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS DE POLARIZACIÓN

Una onda electromagnética que viaja por el espacio puede vibrar en 3 direcciones distintas. Estas corresponden a los ejes x , y y z en coordenadas rectangulares, mientras la onda se propaga en dirección del vector $r = r(x, y, z)$. Esto lo definimos como:

$$u_x(r, t) = u_{0x} \cos(\omega t - k \cdot r + \delta_x) \quad (2.1)$$

$$u_y(r, t) = u_{0y} \cos(\omega t - k \cdot r + \delta_y) \quad (2.2)$$

$$u_z(r, t) = u_{0z} \cos(\omega t - k \cdot r + \delta_z) \quad (2.3)$$

Donde u_x, u_y y u_z son las componentes ortogonales de la onda electromagnética correspondientes a cada eje coordenado y u_{0x}, u_{0y} y u_{0z} son las amplitudes máximas de la onda cosenoidal. La frecuencia angular de la vibración de la onda, $\omega = 2\pi f$ (con f = frecuencia lineal y $k = 2\pi/\lambda$ el número de onda). El desfase entre las componentes de la onda electromagnética se representa como δ_x, δ_y y δ_z .

Cuando el vector de propagación r apunta en dirección al eje z positivo, la componente u_z no existe y se concluye que la onda solo tiene componentes en los ejes x e y , ecs. 2.1 y 2.2. La componente en el eje z se reescribe entonces como: $u_z(z, t) = 0$. Fresnel y Arago demostraron esta propiedad en 1818 usando luz polarizada en el experimento de interferencia de Young [15].

La Figura 2.1 muestra las dos componentes perpendiculares, E_x y E_y , del vector campo eléctrico E de un haz de luz. Estas se encuentran en el mismo plano que el campo óptico, representado con la línea punteada, y se propagan en la dirección del eje z . El campo óptico transversal es el plano en donde la punta del vector E dibuja su trayectoria a lo largo de un ciclo completo, oscilando con una frecuencia angular $\omega = 2\pi/\lambda$, donde λ es la longitud de onda de la luz.

2.1. POLARIZACIÓN DE LA LUZ

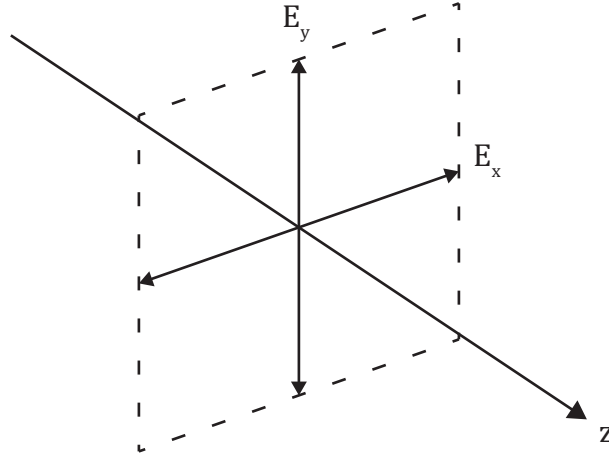


Figura 2.1: Campo óptico y eje de propagación

Se opta por definir la luz en términos del campo eléctrico en lugar del campo magnético, ya que en un haz típico, el campo magnético es considerablemente más débil que el campo eléctrico. Considerando esto podemos definir de forma general a las componentes E_x y E_y simplificando las ecuaciones 2.1 y 2.2:

$$E_x = A_x \cos(\tau) \quad (2.4)$$

$$E_y = A_y \cos(\tau + \delta) \quad (2.5)$$

Donde $\tau = \omega t - k \cdot z$ es llamado *propagador*, y $\delta = \delta_y - \delta_x$ es el desfase relativo de la componente E_y con respecto a E_x .

Cualquier estado de polarización puede describirse en términos de las amplitudes máximas de las componentes de la onda, A_x y A_y y su desfase en el tiempo δ . Podemos clasificar la polarización en tres casos: lineal, circular y elíptica. Posteriormente veremos que los casos lineal y circular son casos degenerados de la elipse.

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS DE POLARIZACIÓN

1. Polarización lineal

Una onda polarizada linealmente se produce cuando, dado un punto en el campo óptico, el vector del campo eléctrico siempre está orientado a lo largo de la misma recta. Esto se cumple en dos casos:

- a) Cuando una de las componentes es nula, como lo muestran las Figuras 2.2a y 2.2b.
- b) Cuando las dos componentes están en fase o con un desfase de π radianes

La Figura 2.2c muestra una onda polarizada linealmente en la que $E_x = E_y = A \cos(\tau)$. Por lo tanto, sus amplitudes máximas son iguales $A = A_x = A_y$, formando un ángulo de 45° entre el eje x y el vector E .

2. Polarización circular

Este particular estado de polarización puede producirse si y solo si, sus dos componentes tienen igual amplitud y una diferencia de fase δ , múltiplo de $\pm\pi/2$ entre sí. La intensidad del campo eléctrico será constante en el tiempo, mientras que su orientación cambiará para describir una circunferencia sobre el campo óptico. El sentido de giro lo podemos determinar con base en la fase de sus componentes. Si E_y está desfasada $-\pi/2$ de E_x se trata de una polarización circular izquierda; en cambio, si E_y se desfasa $\pi/2$ con respecto a E_x hablamos de polarización circular derecha. En la Figura 2.3 podemos observar las dos componentes del campo eléctrico con un desfase $\delta = \pi/2$, el cual sustituiremos en la ecuación 2.5.

$$E_x = A \cos(\tau) \quad E_y = A \cos\left(\tau + \frac{\pi}{2}\right)$$

2.1. POLARIZACIÓN DE LA LUZ

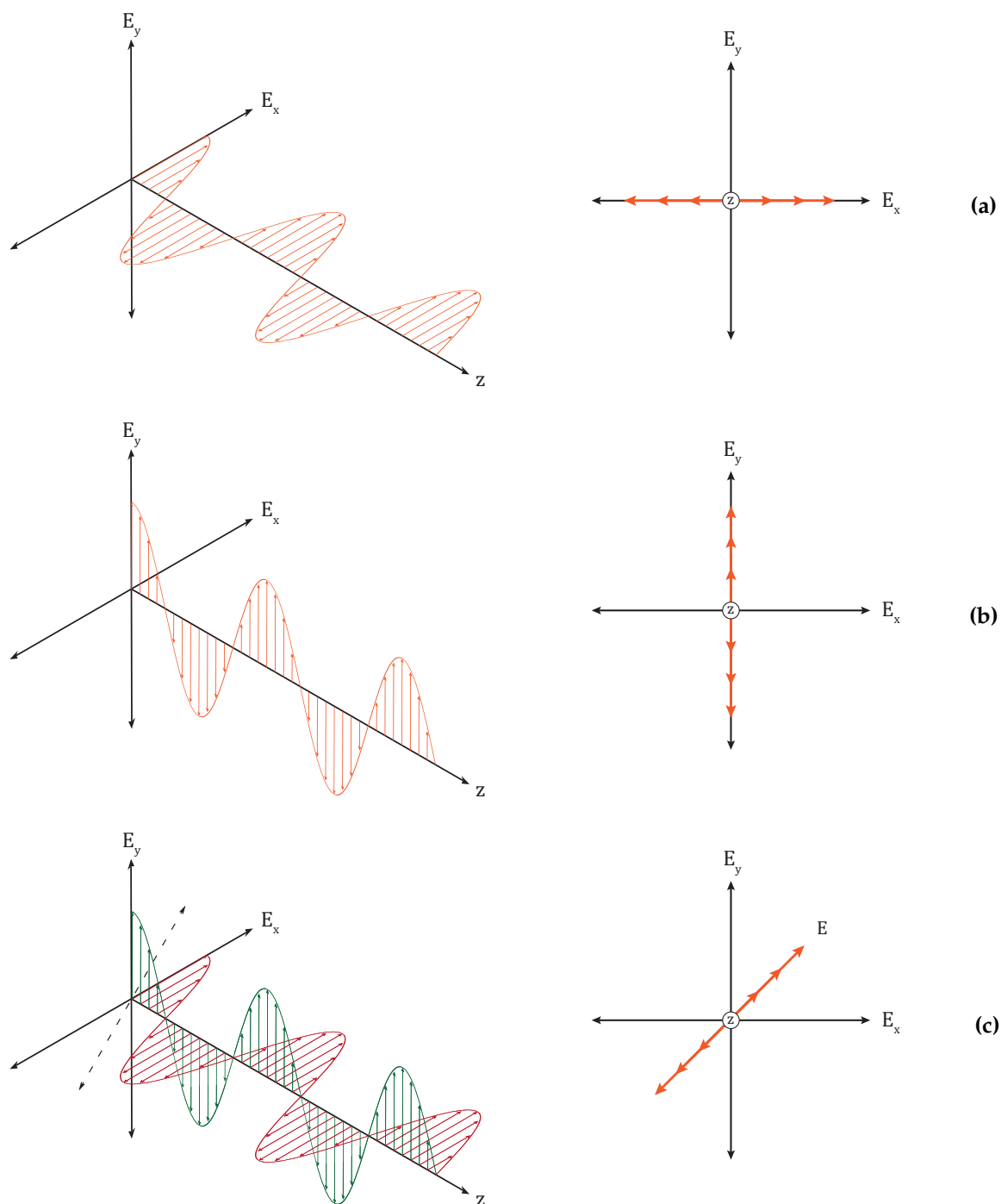


Figura 2.2: Polarizaciones lineales. (a) Polarización vertical (90°), (b) Polarización horizontal (0°), (c) Polarización a 45° .

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS DE POLARIZACIÓN

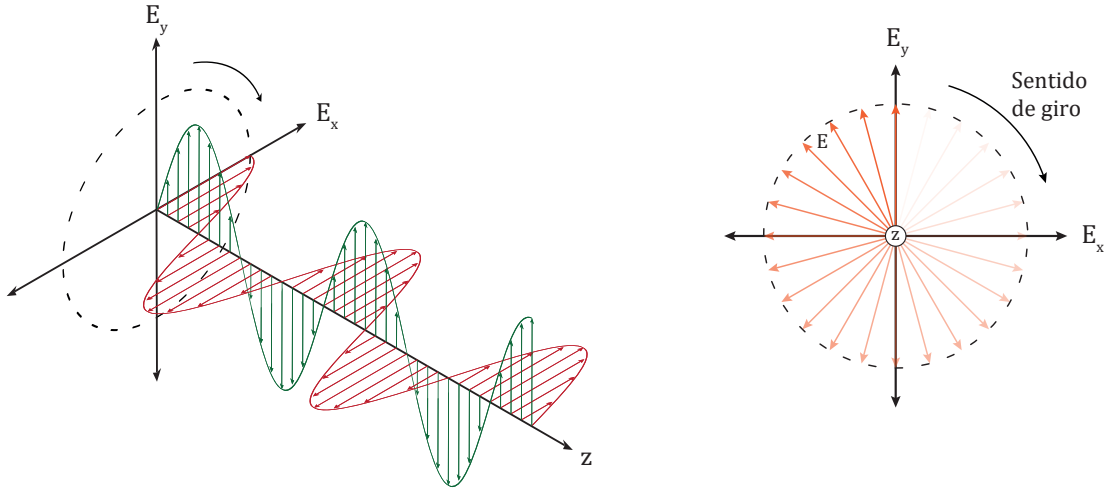


Figura 2.3: Polarización circular

3. Polarización elíptica

Cuando ninguna de las condiciones anteriormente mencionadas para generar estados de polarización se cumple, el vector del campo eléctrico describe una elipse al proyectarse en el campo óptico. Por lo tanto sus componentes ortogonales mantienen la forma general de las ecuaciones 2.4 y 2.5:

$$E_x = A_x \cos(\tau) \quad E_y = A_y \cos(\tau + \delta) \quad (2.6)$$

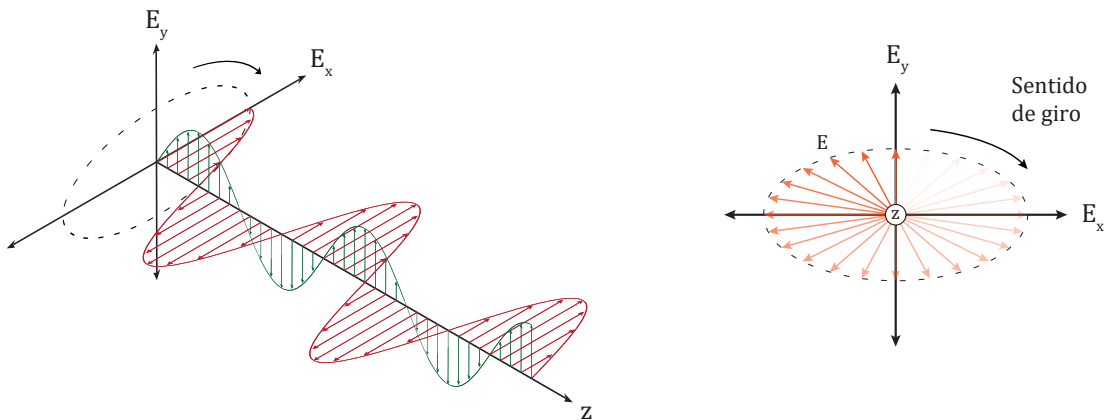


Figura 2.4: Polarización elíptica.

2.1. POLARIZACIÓN DE LA LUZ

Como podemos observar en la ecuación 2.6 las componentes pueden tener diferente amplitud y estar desfasadas cualquier ángulo. Esto deja claro que cuando A_x y A_y son iguales y $\delta = \pm\pi/2$ se produce una polarización circular. En cambio, cuando las componentes se encuentran en fase o con $\delta = \pi$, se genera la polarización lineal, en la que el ángulo con respecto al eje X positivo está dado por la amplitud de sus componentes.

Ahora desarrollaremos de forma algebraica la ecuación que nos permite calcular analíticamente la forma de la onda en el campo óptico. Partiremos con E_x y E_y en su forma más general, descritas en la ecuación 2.6 y las reescribiremos de la siguiente manera mediante identidades trigonométricas:

$$\frac{E_x}{A_x} = \cos(\tau) \quad (2.7)$$

$$\frac{E_y}{A_y} = \cos(\tau) \cos(\delta) - \sin(\tau) \sin(\delta) \quad (2.8)$$

Sustituimos 2.7 en 2.8 y agrupamos términos:

$$\begin{aligned} \frac{E_y}{A_y} &= \frac{E_x}{A_x} \cos(\delta) - \sin(\tau) \sin(\delta) \\ \frac{E_y}{A_y} - \frac{E_x}{A_x} \cos(\delta) &= -\sin(\tau) \sin(\delta) \end{aligned} \quad (2.9)$$

Empleando la identidad pitagórica, en 2.7 tenemos:

$$\sin(\tau) = \left[1 - \cos^2(\tau)\right]^{1/2} \quad (2.10)$$

Y sustituyendo de nuevo 2.7 en 2.10 nos queda:

$$\sin(\tau) = \left[1 - \left(\frac{E_x}{A_x}\right)^2\right]^{1/2} \quad (2.11)$$

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS DE POLARIZACIÓN

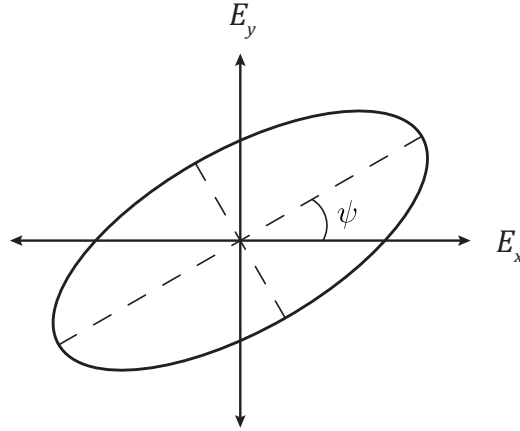


Figura 2.5: Elipse de polarización.

Al reemplazar la ecuación anterior en 2.9 y simplificar, llegamos a la ecuación general de la elipse de polarización.

$$\begin{aligned} \frac{E_y}{A_y} - \frac{E_x}{A_x} \cos(\delta) &= - \left[1 - \left(\frac{E_x}{A_x} \right)^2 \right]^{1/2} \sin(\delta) \\ \left(\frac{E_y}{A_y} - \frac{E_x}{A_x} \cos(\delta) \right)^2 &= \left[1 - \left(\frac{E_x}{A_x} \right)^2 \right] \sin^2(\delta) \\ \left(\frac{E_y}{A_y} \right)^2 - 2 \left(\frac{E_y}{A_y} \right) \left(\frac{E_x}{A_x} \right) \cos(\delta) + \left(\frac{E_x}{A_x} \right)^2 \cos^2(\delta) &= \sin^2(\delta) - \left(\frac{E_x}{A_x} \right)^2 \sin^2(\delta) \end{aligned}$$

Quedando finalmente como:

$$\left(\frac{E_x}{A_x} \right)^2 + \left(\frac{E_y}{A_y} \right)^2 - 2 \left(\frac{E_x}{A_x} \right) \left(\frac{E_y}{A_y} \right) \cos(\delta) = \sin^2(\delta) \quad (2.12)$$

El ángulo ψ que se forma entre el eje mayor de la elipse (o la recta si se trata de polarización lineal) y el eje x se puede obtener mediante la siguiente expresión, que relaciona

2.1. POLARIZACIÓN DE LA LUZ

el ángulo de rotación ψ con A_x , A_y y δ .

$$\tan(2\psi) = \frac{2A_x A_y \cos(\delta)}{A_x^2 - A_y^2} \quad (2.13)$$

Analizando la ecuación de la elipse (2.12) observamos que los parámetros A_x , A_y y δ determinan el tipo de polarización que presenta un haz de luz.

1. Solo existe una componente, la otra es nula.

$$E_x = 0$$

$$E_y = A \cos(\tau)$$

En este caso, es claro que la onda solo vibra en dirección del eje y , ya que la componente en x no existe, dando lugar a la polarización lineal vertical (Figura 2.2a). Algo similar sucede cuando $E_y = 0$ y $E_x \neq 0$, entonces tenemos que la onda oscila sobre el eje x y hablamos de polarización lineal horizontal (Figura 2.2b).

2. Componentes en fase $\delta = 0$ o $\delta = \pi$

La ecuación 2.12 se convierte en

$$\begin{aligned} \left(\frac{E_x}{A_x}\right)^2 + \left(\frac{E_y}{A_y}\right)^2 \pm 2\left(\frac{E_x}{A_x}\right)\left(\frac{E_y}{A_y}\right) &= 0 \\ \left(\frac{E_x}{A_x} \pm \frac{E_y}{A_y}\right)^2 &= 0 \\ E_y &= \pm \left(\frac{A_y}{A_x}\right) E_x \end{aligned} \quad (2.14)$$

Siendo esta la ecuación de una recta con pendiente $\pm \left(\frac{A_y}{A_x}\right)$ y ordenada al origen. El signo positivo corresponde con $\delta = 0$ y el signo negativo a $\delta = \pi$. Cuando

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS DE POLARIZACIÓN

$A_x = A_y = A$ la ecuación 2.14 se reduce a $E_y = \pm E_x$, que representa una recta a $\pm 45^\circ$ con pendiente igual a 1, como se aprecia en la Figura 2.2c.

3. Componentes desfasadas $\delta = \frac{\pi}{2}$, o $\delta = \frac{3\pi}{2}$

Sustituyendo el valor de δ en 2.12 nos queda:

$$\left(\frac{E_x}{A_x}\right)^2 + \left(\frac{E_y}{A_y}\right)^2 = 1 \quad (2.15)$$

Esta es la ecuación de una elipse orientada sobre el eje de la componente que tenga mayor amplitud.

4. Componentes desfasadas con igual amplitud $\delta = \frac{\pi}{2}$ o $\delta = \frac{3\pi}{2}$ y $A_x = A_y = A$

Finalmente si las amplitudes máximas de las componentes son iguales tenemos el estado de polarización circular. La ecuación de la elipse se reduce a la ecuación de una circunferencia con centro en el origen y radio A .

$$\left(\frac{E_x}{A}\right)^2 + \left(\frac{E_y}{A}\right)^2 = 1 \quad (2.16)$$

Para los casos 3 y 4, correspondientes a polarización circular y elíptica, el sentido de giro se determina por el desfase de la componente E_y con respecto a E_x y la convención es la siguiente

$$0 \text{ [rad]} < \delta < \pi \text{ [rad]}$$

Polarización derecha

$$\pi \text{ [rad]} < \delta < 2\pi \text{ [rad]}$$

Polarización izquierda

2.2. Formalismo de Stokes-Müller

El vector eléctrico de la luz traza una elipse, una circunferencia o una línea recta en el campo óptico aproximadamente cada $10^{-15}[s]$. Este periodo es demasiado corto para ser observado. Además, la ecuación de la elipse de polarización no puede describir luz parcialmente polarizada, o sea aquella en la que solo una proporción del haz se encuentra polarizado. Estas limitaciones llevan a definir el estado de polarización en términos de magnitudes observables que puedan describir luz no polarizada [12].

Stokes describió en 1852 [16, 17] que la polarización de un haz de luz puede representarse con cuatro parámetros directamente medibles: la intensidad total, la intensidad de luz polarizada a 0° y 90° , la intensidad de luz polarizada a 45° y 135° y la intensidad de polarizaciones circular izquierda y derecha. La intensidad es el promedio en el tiempo del cuadrado de la amplitud de las ondas cosenoidales.

Considerando A_x^2 y A_y^2 un promedio temporal de las amplitudes instantáneas de las componentes E_x y E_y respectivamente, podemos definir como parámetros de Stokes a $S_1 - S_4$ de la siguiente manera:

$$s_0 = A_x^2 + A_y^2$$

$$s_1 = A_x^2 - A_y^2$$

$$s_2 = 2A_x A_y \cos(\delta)$$

$$s_3 = 2A_x A_y \sin(\delta)$$

Cuando un haz de luz está totalmente polarizado, se cumple la igualdad en la ecuación 2.17. En caso contrario, el signo de desigualdad indica que la intensidad total s_0 es mayor a la suma de intensidades de luz polarizada, ya que existe una aportación de

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS DE POLARIZACIÓN

intensidad de luz no polarizada.

$$s_0^2 \geq s_1^2 + s_2^2 + s_3^2 \quad (2.17)$$

Con la ecuación 2.17 podemos establecer una magnitud, llamada *grado de polarización*, P , que cuantifica la proporción de luz polarizada en el haz, de acuerdo con:

$$P = \frac{I_{\text{polarizada}}}{I_{\text{total}}} = \frac{(s_1^2 + s_2^2 + s_3^2)^{1/2}}{s_0}, \quad 0 \leq P \leq 1 \quad (2.18)$$

$P = 1$ describe un haz de luz completamente polarizado; $P = 0$ indica que no existe dirección de vibración preferencial, o sea, la luz no está polarizada).

En adelante denotaremos un haz de luz y su estado de polarización como el vector $\vec{S}$, en el que sus elementos son los parámetros de Stokes.

$$\vec{S} = \begin{bmatrix} A_x^2 + A_y^2 \\ A_x^2 - A_y^2 \\ 2A_x A_y \cos \delta \\ 2A_x A_y \sin \delta \end{bmatrix} = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad (2.19)$$

Algunos ejemplos de estados de polarización y el vector de Stokes que lo representa se muestran en la tabla 2.1.

Con ello, podemos plantear un análisis matricial que exprese las interacciones del haz de luz con el medio que interactúa, y que altera su estado de polarización. El planteamiento ilustrado en la figura 2.6 es el siguiente: un haz de luz incidente, $\vec{S}$, interactúa con un medio $\mathbf{M}$ por transmisión o reflexión, dando como resultado un haz de luz de salida $\vec{S}'$.

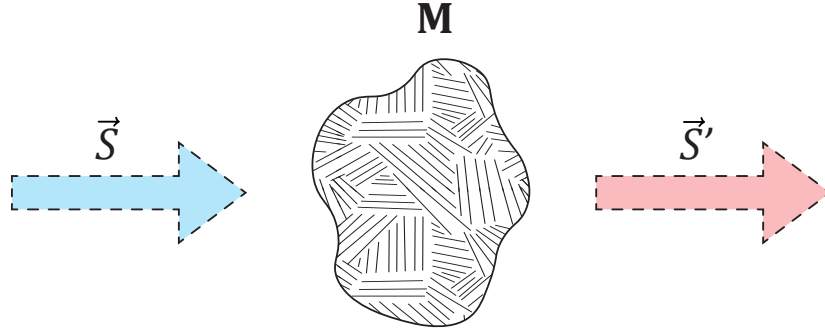


Figura 2.6: Interacción de un haz de luz con un medio.

Algebráicamente, el vector de Stokes de salida $\vec{S}'$ es igual a la multiplicación del vector de Stokes de entrada $\vec{S}$ por la matriz $\mathbf{M}$, ($\mathbf{M}\vec{S}$). Como el resultado debe ser otro vector columna de 4×1 (otro vector de Stokes), la matriz $\mathbf{M}$ debe tener dimensiones 4×4 . Esta es llamada la matriz de Müller.

$$\vec{S}' = \mathbf{M}\vec{S} \quad (2.20)$$

$$\begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \end{bmatrix} = \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

Una matriz de Müller caracteriza completamente la transformación lineal que ocurre en la polarización del haz de luz. A continuación, se listan matrices de Müller de distintos elementos ópticos y explica cómo transforman el vector de Stokes incidente $\vec{S}$ en el vector de Stokes de salida $\vec{S}'$. [13]

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS DE POLARIZACIÓN

Estado de polarización	Vector de Stokes
Lineal 0° (horizontal) $A_y = 0$	$A_x^2 \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$
Lineal 90° (vertical) $A_x = 0$	$A_y^2 \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix}$
Lineal 45° $A_x = A_y = A$ $\delta = 0$	$2A^2 \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$
Lineal -45° $A_x = A_y = A$ $\delta = \pi$	$2A^2 \begin{bmatrix} 1 \\ 0 \\ -1 \\ 0 \end{bmatrix}$
Circular derecha $A_x = A_y = A$ $\delta = \frac{\pi}{2}$	$2A^2 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$
Circular izquierda $A_x = A_y = A$ $\delta = \frac{3\pi}{2}$	$2A^2 \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$

Tabla 2.1: Estados de polarización y su correspondiente vector de Stokes normalizado.

2.2. FORMALISMO DE STOKES-MÜLLER

■ Elemento no polarizador

Es un elemento que no cambia el estado de polarización. La polarización en la entrada es la misma a la salida. Como ejemplo tenemos el *aire*.

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad (2.21)$$

■ Atenuador

Este elemento tampoco cambia el estado de polarización del haz incidente, pero sí reduce su intensidad. Un filtro de densidad neutra se comporta como un atenuador.

$$\begin{bmatrix} a \cdot s_0 \\ a \cdot s_1 \\ a \cdot s_2 \\ a \cdot s_3 \end{bmatrix} = \begin{bmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & a \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad (2.22)$$

■ Polarizador lineal

Polariza el haz de luz linealmente en la dirección del eje de transmisión del elemento. La constante $1/2$ se debe a que este elemento solo deja pasar una de las dos componentes ortogonales del campo eléctrico, absorbiendo la otra mitad de la intensidad.

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS DE POLARIZACIÓN

Cuando el eje de transmisión se encuentra a 0° :

$$\frac{1}{2} \begin{bmatrix} s_0 + s_1 \\ s_0 + s_1 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad (2.23)$$

Eje de transmisión a 90° :

$$\frac{1}{2} \begin{bmatrix} s_0 - s_1 \\ s_1 - s_0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad (2.24)$$

■ Retardador lineal de $\frac{\lambda}{4}$

Introduce un desfase relativo de $\pi/4$, igual a un cuarto del periodo de la onda entre sus componentes. Tiene dos ejes ortogonales denominados *rápido* y *lento*. La componente del campo eléctrico que se alinea con el eje rápido atraviesa la placa sin cambios; para la componente alineada con el eje lento se retrasa $\delta = \pi/2$. Al igual que el polarizador, su matriz de Müller depende del ángulo azimutal, pero en este no hay pérdida de intensidad ya que $s'_0 = s_0$. En adelante nos referiremos al retardador lineal de cuarto de onda como QWP por su nombre en inglés (*quarter wave plate*).

Eje rápido a 0° :

$$\begin{bmatrix} s_0 \\ s_1 \\ s_3 \\ -s_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad (2.25)$$

■ Matriz de rotación

Cualquier matriz de Müller que dependa de un ángulo azimutal, p. ej. el polarizador o el retardador, puede ser *rotada* para obtener una matriz equivalente a tener el eje de transmisión o el eje rápido a un ángulo θ . Para esto multiplicaremos la matriz de rotación $\mathbf{Rot}(\theta)$ por la matriz de Müller del elemento de la siguiente manera:

$$\mathbf{M}(\theta) = \mathbf{Rot}(\theta)\mathbf{M}(0^\circ)\mathbf{Rot}(-\theta) \quad (2.26)$$

Donde $\mathbf{M}(0^\circ)$ es la matriz a 0° del elemento en cuestión. La matriz de rotación es:

$$\mathbf{Rot}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(2\theta) & -\sin(2\theta) & 0 \\ 0 & \sin(2\theta) & \cos(2\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.27)$$

Por ejemplo, para obtener la matriz de Müller de un retardador de $\lambda/4$ con su eje rápido en 30° , a la que llamaremos $\mathbf{M}_{\text{QWP}}(30^\circ)$, multiplicaremos la matriz que aparece en la eq. 2.25 por la matriz rotadora, como la indicada en la expresión 2.26, sustituyendo $\theta = 30^\circ$.

$$\mathbf{M}_{\text{QWP}}(30^\circ) = \mathbf{Rot}(30^\circ)\mathbf{M}_{\text{QWP}}(0^\circ)\mathbf{Rot}(-30^\circ)$$

$$\mathbf{M}_{\text{QWP}}(30^\circ) =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(60^\circ) & -\sin(60^\circ) & 0 \\ 0 & \sin(60^\circ) & \cos(60^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-60^\circ) & -\sin(-60^\circ) & 0 \\ 0 & \sin(-60^\circ) & \cos(-60^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS DE POLARIZACIÓN

$$\mathbf{M}_{\text{QWP}}(30^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5 & -0.86 & 0 \\ 0 & 0.86 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0.86 & 0 \\ 0 & -0.86 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{M}_{\text{QWP}}(30^\circ) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.25 & 0.43 & -0.86 \\ 0 & 0.43 & 0.75 & 0.5 \\ 0 & 0.86 & -0.5 & 0 \end{bmatrix} \quad (2.28)$$

$\mathbf{M}_T$ es la matriz que representa a los tres medios o elementos con los que interactúa el haz de luz: $\mathbf{M}_1$, $\mathbf{M}_2$ y $\mathbf{M}_3$. También se puede entender a $\mathbf{M}_T$ como un medio que se puede descomponer en varios elementos independientes: $\mathbf{M}_1$, $\mathbf{M}_2$ y $\mathbf{M}_3$.

■ Matriz de varios elementos

Finalmente, si el medio que queremos representar con una matriz de Müller se compone de varios elementos que pueden interactuar con la polarización, la matriz que representa la transformación final que sufrirá el haz de luz es igual a la multiplicación de las matrices de cada elemento, multiplicadas en el orden que aparecen.

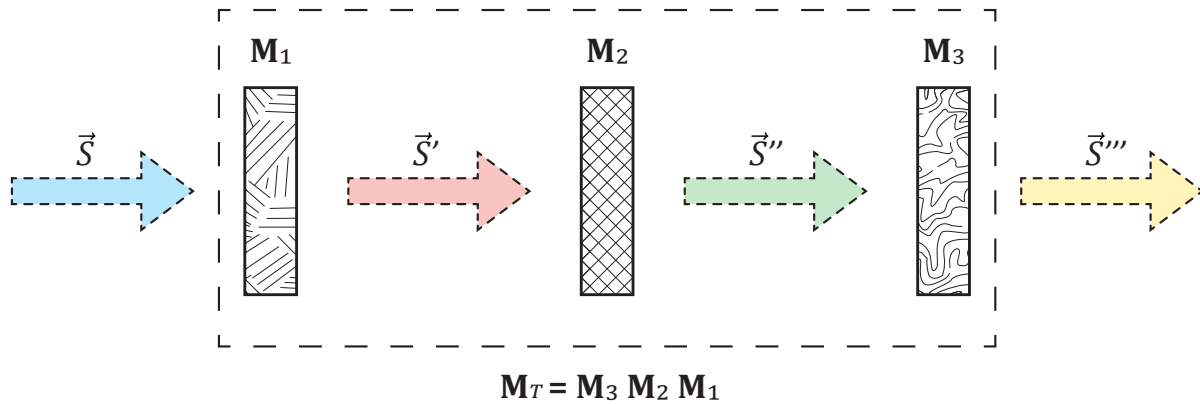


Figura 2.7: Matriz de Müller equivalente a varios elementos en serie.

2.2. FORMALISMO DE STOKES-MÜLLER

Modelando la interacción mostrada en la figura 2.7 tenemos:

$$\vec{S}' = \mathbf{M}_1 \vec{S}$$

$$\vec{S}'' = \mathbf{M}_2 \vec{S}'$$

$$\vec{S}''' = \mathbf{M}_3 \vec{S}''$$

$$\vec{S}''' = \mathbf{M}_3 \mathbf{M}_2 \mathbf{M}_1 \vec{S}$$

$\therefore$

$$\mathbf{M}_T = \mathbf{M}_3 \mathbf{M}_2 \mathbf{M}_1$$

$$\vec{S}''' = \mathbf{M}_T \vec{S}$$

Una característica muy importante de los vectores de Stokes y las matrices de Müller es que representan elementos reales. Esto significa que un vector de Stokes representa un haz de luz, si y solo si su grado de polarización está en el intervalo cerrado de 0 a 1. Este es llamado el criterio de Stokes. Como lo indica la ecuación 2.17, el cuadrado de la intensidad total de un haz de luz debe ser mayor o igual a la suma de los cuadrados de las intensidades de luz polarizada. En un sentido físico, el grado de polarización en este rango indica que la proporción de intensidad de luz polarizada en el haz no puede ser negativa ni superar la intensidad total. La segunda condición que debe cumplirse se refiere a la *pasividad* en la interacción del medio con la luz y su estado de polarización, esto implica que no se amplifique la intensidad de la onda electromagnética en el haz de salida.

Considerando que una matriz de Müller representa una transformación en la polarización de la luz, $\mathbf{M}$ es formalmente una matriz de Müller si: para todo $\vec{S}$ que cumpla con el criterio de Stokes, el vector resultante $\vec{S}' = \mathbf{M}\vec{S}$ también cumple con este criterio. Además su intensidad total es igual o menor que la del haz incidente $s'_0 \leq s_0$.

CAPÍTULO 2. FUNDAMENTOS TEÓRICOS DE POLARIZACIÓN

El instrumento que mide parcial o completamente los elementos de estas dos estructuras matemáticas se conoce como *polarímetro* y su clasificación dependerá de lo que es capaz de medir. Un polarímetro de Stokes mide, como su nombre lo indica, los elementos del vector de Stokes, que describen el estado de polarización presente en el haz de luz. El polarímetro de Müller mide los elementos de la matriz de Müller, es decir, caracteriza la transformación que sufrirá el estado de polarización cuando la luz se refleje o transmita en una muestra. Si el polarímetro mide los cuatro elementos del vector o los dieciséis elementos de la matriz se considera que es *completo*. [13]

Capítulo 3

Diseño del polarímetro de Müller

Uno de los objetivos de esta tesis es caracterizar la actividad eléctrica de una muestra biológica por medio de la matriz de Müller. Esto requiere de la construcción de un equipo capaz de obtener una matriz 4×4 a partir de mediciones de intensidad luminosa. El polarímetro debe ser caracterizado para eliminar sus errores sistemáticos y comprobar que la matriz resultante cumpla con la definición matemática de una matriz de Müller descrita en el capítulo anterior.

3.1. Polarímetro de doble retardador rotador (DRR)

El polarímetro de doble retardador rotador es una de varias configuraciones posibles para un polarímetro de Müller capaz de calcular todos los elementos de la matriz de Müller de una muestra a partir de varias mediciones de la intensidad de la luz que llega al sensor. Estas mediciones conforman una señal simétrica, que es modulada por las rotaciones de dos placas retardadoras de cuarto de onda, con una razón de giro $\theta : 5\theta$. Los elementos de la matriz son obtenidos mediante la relación de estos con los coeficientes de la serie de Fourier de la señal modulada.

CAPÍTULO 3. DISEÑO DEL POLARÍMETRO DE MÜLLER

Para construir un polarímetro de Müller completo se necesita un generador de estados de polarización, PSG por sus siglas en inglés (Polarization State Generator), capaz de producir un haz de luz con cualquier tipo de polarización. La luz incide en la muestra y sale hacia el analizador de estados de polarización y un PSA por sus siglas en inglés (Polarization State Analyzer). El PSA es un arreglo con una configuración invertida respecto al PSG, que también le permite configurar cualquier estado de polarización. [18]

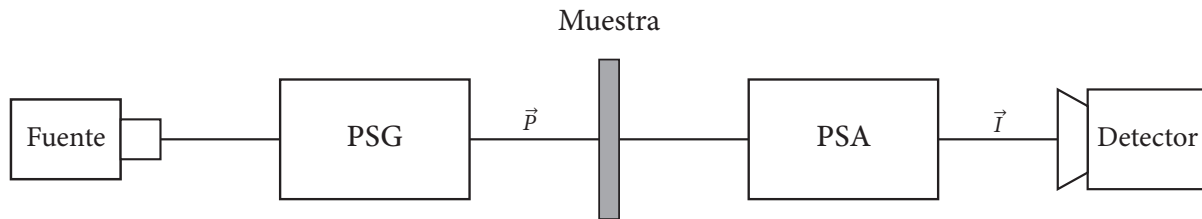


Figura 3.1: Arreglo de un polarímetro de Müller en transmisión.

La figura 3.1 ilustra un haz de luz proveniente de una fuente no coherente que es polarizado por el PSG. El estado $\vec{P}$ incide en una muestra M que modifica la polarización. Al pasar por el PSA, la luz adquiere el estado de polarización $\vec{I}$ con la que llega al detector.

La magnitud de un haz de luz que podemos medir directamente es su intensidad total, o sea, el primer elemento de su vector de Stokes asociado. Por lo tanto, la ecuación a resolver es la siguiente:

$$\vec{I} = c \mathbf{A} \mathbf{M} \vec{P}$$

$$\begin{bmatrix} I_0 \\ \bullet \\ \bullet \\ \bullet \end{bmatrix} = c \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix} \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \quad (3.1)$$

3.1. POLARÍMETRO DE DOBLE RETARDADOR ROTADOR (DRR)

Donde $\vec{I}$ es el vector de Stokes que representa el haz de luz que incide en el detector, mientras que I_0 es la intensidad que se registra. $\mathbf{A}$ es la matriz de Müller del PSA. $\mathbf{M}$ la matriz de Müller de la muestra y $\vec{P}$ es el vector del haz de luz polarizado que sale del PSG e ilumina la muestra. Nótese que en la multiplicación matricial, el orden de los elementos está a la inversa que en la fig. 3.1, ya que el orden de multiplicación se efectúa, de izquierda a derecha. Además, la multiplicación de matrices no es conmutativa. La constante c es un factor de proporcionalidad del sensor que multiplica el vector de polarización por un escalar, y este es la intensidad del haz de luz. En la ecuación 3.1 los elementos que se marcan con puntos no influyen en la resolución de la matriz.

Para configurar el PSG y PSA se emplean un polarizador lineal y un retardador de $\lambda/4$ para cada módulo, dispuestos como lo muestra la figura 3.2. Para obtener los 16 elementos de la matriz $\mathbf{M}$ se necesitan como mínimo 16 configuraciones del generador y analizador de estados, pues la estrategia es completar un sistema de ecuaciones lineales. Más adelante se describe el método para obtener estas 16 incógnitas, aunque se recurre a una solución por mínimos cuadrados con la cual tendremos infinitas soluciones que se aproximan al resultado, esto es útil especialmente en la aplicación experimental, donde encontrar una solución exacta es prácticamente imposible.

La matriz de Müller de todo el sistema es la multiplicación de las matrices de cada elemento que atraviesa el haz de luz, recordando que el orden de multiplicación es por orden de aparición. Representando con elementos físicos, la figura 3.1 se transforma en la figura 3.2 donde podemos observar la siguiente matriz característica:

$$\mathbf{P}_2 \mathbf{R}_2(\theta) \mathbf{M} \mathbf{R}_1(\theta) \mathbf{P}_1 \quad (3.2)$$

Donde $\mathbf{P}_1$ y $\mathbf{P}_2$ son los polarizadores lineales con su eje de transmisión orientado a 0° . Un polarizador fijo delante del sensor asegura que el estado de polarización de la luz medida sea constante, eliminando cualquier sesgo del sensor hacia una polarización específica. Las matrices $\mathbf{R}_1(\theta)$ y $\mathbf{R}_2(\theta)$ representan a los dos retardadores y dependen

CAPÍTULO 3. DISEÑO DEL POLARÍMETRO DE MÜLLER

de un ángulo azimutal θ , ya que sus rotaciones generan las variaciones en la intensidad detectada.

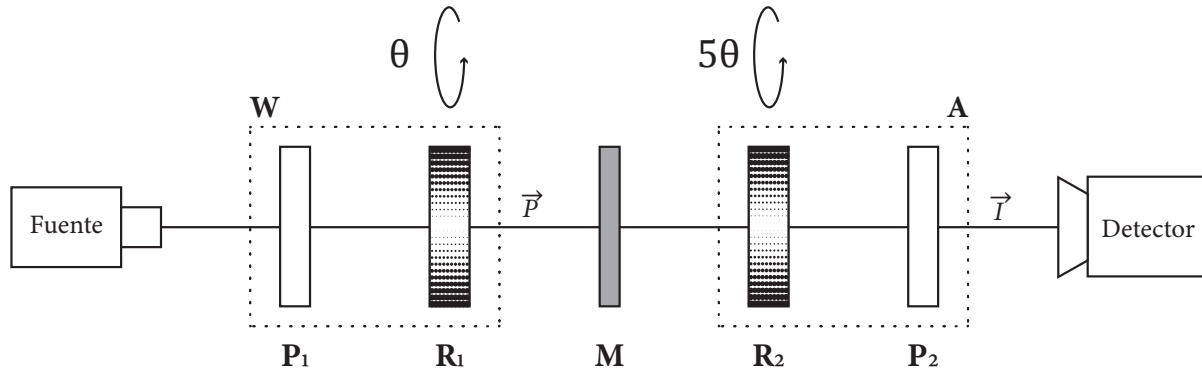


Figura 3.2: Disposición de los polarizadores (P_1 y P_2) y retardadores (R_1 y R_2). W y A representan el PSG y PSA respectivamente. M es la muestra a caracterizar. $\vec{P}$ e $\vec{I}$ son los vectores de Stokes con la polarización que tiene el haz de luz en ese punto del recorrido.

Suponiendo que $\vec{S}$ es un haz de luz no polarizado proveniente de la fuente, el vector $\vec{P}$, la matriz A de la ecuación 3.1, la matriz W del PSG se definen como:

$$\vec{P} = R_1 P_1 \vec{S}$$

$$A = P_2 R_2$$

$$W = R_1 P_1$$

Si operamos los términos de la expresión 3.1 encontraremos que se puede modificar el orden de la multiplicación de las matrices de la siguiente manera:

$$I_0 = a_0(m_{00}p_0 + m_{01}p_1 + m_{02}p_2 + m_{03}p_3) + a_1(m_{10}p_0 + m_{11}p_1 + m_{12}p_2 + m_{13}p_3) \\ + a_2(m_{20}p_0 + m_{21}p_1 + m_{22}p_2 + m_{23}p_3) + a_3(m_{30}p_0 + m_{31}p_1 + m_{32}p_2 + m_{33}p_3)$$

3.1. POLARÍMETRO DE DOBLE RETARDADOR ROTADOR (DRR)

Que es igual a:

$$I_0 = \sum_{i,j=0}^3 a_i p_j m_{ij} = \sum_{i,j=0}^3 \mu_{ij} m_{ij} \quad (3.3)$$

Cuando la relación entre la rotación de los retardadores del PSA y del PSG es 5:1, los dieciséis términos $\mu_{ij} = a_i p_j$ son:

$$\mu_{00} = 1$$

$$\mu_{01} = \cos^2 2\theta$$

$$\mu_{02} = \sin 2\theta \cos 2\theta$$

$$\mu_{03} = \sin 2\theta$$

$$\mu_{10} = \cos^2 10\theta$$

$$\mu_{11} = \cos^2 2\theta \cos^2 10\theta$$

$$\mu_{12} = \sin 2\theta \cos 2\theta \cos^2 10\theta$$

$$\mu_{13} = \sin 2\theta \cos^2 10\theta$$

$$\mu_{20} = \sin 10\theta \cos 10\theta$$

$$\mu_{21} = \cos^2 2\theta \sin 10\theta \cos 10\theta$$

$$\mu_{22} = \sin 2\theta \cos 2\theta \sin 10\theta \cos 10\theta$$

$$\mu_{23} = \sin 2\theta \sin 10\theta \cos 10\theta$$

$$\mu_{30} = -\sin 10\theta$$

$$\mu_{31} = -\cos^2 2\theta \sin 10\theta$$

$$\mu_{32} = -\sin 2\theta \cos 2\theta \sin 10\theta$$

$$\mu_{33} = -\sin 2\theta \sin 10\theta$$

CAPÍTULO 3. DISEÑO DEL POLARÍMETRO DE MÜLLER

La intensidad (I_0) es medida como niveles de grises por el sensor. Los retardadores del PSG y PSA rotan con pasos de 5° y 25° respectivamente, hasta que el PSG completa 180° . El resultado de estas ecuaciones es transformado en una serie de Fourier con las primeras 12 frecuencias, ya que los coeficientes de esta serie son función de los elementos de la matriz de Müller. La relación de rotación 5:1 es la más pequeña que podemos usar para transformar el sistema al dominio de fourier y de nuevo a su dominio original, de manera que siga siendo equivalente. Este paso es fundamental para obtener los elementos de la matriz. Si escribimos los elementos de $\mathbf{M}$ en términos de los coeficientes de la serie de Fourier, quedan las siguientes expresiones [11]:

$$\begin{aligned} m_{00} &= a_0 - a_2 + a_8 - a_{10} + a_{12} \\ m_{01} &= 2a_2 - 2a_8 - 2a_{12} \\ m_{02} &= 2b_2 + 2b_8 - 2b_{12} \\ m_{03} &= b_1 - 2b_{11} = b_1 + 2b_9 = b_1 + b_9 - b_{11} \\ m_{10} &= -2a_8 + 2a_{10} - 2a_{12} \\ m_{11} &= 4a_8 + 4a_{12} \\ m_{12} &= -4b_8 + 4b_{12} \\ m_{13} &= -4b_9 = 4b_{11} = 2(-b_9 + b_{11}) \\ m_{20} &= -2b_8 + 2b_{10} - 2b_{12} \\ m_{21} &= 4b_8 + 4b_{12} \\ m_{22} &= 4a_8 - 4a_{12} \\ m_{23} &= 4a_9 = -4a_{11} = 2(a_9 - a_{11}) \\ m_{30} &= 2b_3 - b_5 = -b_5 + 2b_7 = b_3 - b_5 + b_7 \\ m_{31} &= -4b_3 = -4b_7 = -2(b_3 + b_7) \\ m_{32} &= -4a_3 = 4a_7 = 2(-a_3 + a_7) \\ m_{33} &= -2a_4 = 2a_6 = a_6 - a_4 \end{aligned} \tag{3.4}$$

3.1. POLARÍMETRO DE DOBLE RETARDADOR ROTADOR (DRR)

Tomando la primera medición en 0° y los incrementos de 5° en el retardador del PSG se requieren 36 pasos para completar 180° . Esto genera un sistema sobredeterminado con 25 incógnitas (coeficientes de Fourier) y 36 ecuaciones que provienen de los estados de polarización configurados por el PSG y el PSA. El método de mínimos cuadrados aproxima una solución al resolver la ecuación:

$$\mathbf{x}a = I \quad (3.5)$$

Donde $\mathbf{x}$ es una matriz 36×25 que representa la estructura de 36 series de fourier expandidas en 12 términos de senos y cosenos, es decir 25 coeficientes (a_0 más los 24 coeficientes que corresponden a 12 senos y 12 cosenos). Cada renglón tiene la siguiente forma:

$$\left[1 \quad \cos 2\theta \quad \cos 4\theta \quad \dots \quad \cos 24\theta \quad \sin 2\theta \quad \sin 4\theta \quad \dots \quad \sin 24\theta \right]$$

El vector columna a de tamaño 25×1 contiene los 25 coeficientes de estas series y es nuestra incógnita a resolver en la ecuación 3.5.

I es la señal modulada de la intensidad en forma de vector columna 36×1 . Cada elemento corresponde a la medición de intensidad del haz de luz detectada experimentalmente.

A cada renglón de $\mathbf{x}$ le corresponde un valor θ de las 36 posiciones del retardador del PSG, es decir $\theta = 0^\circ, 5^\circ, \dots, 175^\circ$.

Como $\mathbf{x}$ es una matriz rectangular, no podemos multiplicar ambos lados de la ecuación por su inversa (ya que no existe). La solución para encontrar los coeficientes por mínimos cuadrados es entonces:

$$a = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T I \quad (3.6)$$

CAPÍTULO 3. DISEÑO DEL POLARÍMETRO DE MÜLLER

Una vez encontrado el vector columna a que contiene los 25 coeficientes de Fourier, los sustituimos en las expresiones 3.4 para obtener los 16 elementos de $\mathbf{M}$.

Al tratarse de un sistema sobredeterminado en el que buscamos la solución con menor error, la matriz de Müller obtenida no es exactamente la matriz de la muestra, sino que está contaminada por los errores sistemáticos del polarímetro, como: ángulos azimutales distintos para los ejes de transmisión y ejes rápidos, falta de alineación de todos los elementos con respecto al eje de transmisión del haz de luz y aberraciones propias de la fabricación de los elementos ópticos. Además, existen errores producidos aleatoriamente por el ruido térmico en el sensor, la cuantificación de la intensidad en un sensor con un rango de valores discretos, la iluminación ambiental no deseada o por interferencia electromagnética.

3.2. Filtrado de los resultados experimentales

La matriz experimental se somete a un método de *calibración* para compensar el error sistemático desarrollado a la par de este trabajo por Issac Chavez y Dilia Aguirre en [19]. Esto nos da como resultado una aproximación más cercana de la matriz de Müller de la muestra. Sin embargo, aún no cumple con los criterios de Stokes y de pasividad mencionados en el capítulo 2 para representar un elemento óptico físicamente realizable.

El método de Cloude [20], nos permite, separar sistemáticamente una matriz de Müller experimental en dos componentes. Uno que es físicamente realizable y el otro que no lo es. Debido a que no suponemos de antemano propiedades o características ópticas de la muestra, este método es la mejor vía para llegar a una aproximación físicamente realizable de la matriz de Müller de la muestra.

Este método se basa en representar a la matriz de Müller, $\mathbf{M}$, como una suma convexa de matrices Müller puras $\mathbf{M}_{ji}$, que tienen equivalencia en matrices de Jones [21]. Esto es, una combinación lineal de matrices que representan transformaciones *puras* de la pola-

3.2. FILTRADO DE LOS RESULTADOS EXPERIMENTALES

rización, ponderadas con coeficientes no negativos que al sumarlos da como resultado 1, es decir:

$$\mathbf{M} = \sum_{i=1}^n p_i \mathbf{M}_{ji} \quad ; \quad 0 \leq p_i, \quad \sum_{i=1}^n p_i = 1$$

Las matrices puras representan transformaciones deterministas de la polarización de la luz (traslaciones de un punto sobre la esfera de Poincaré que representa el estado de polarización). Por lo tanto su combinación lineal produce una matriz que representa un elemento óptico polarimétricamente puro, que no modifica el grado de polarización de un haz de luz [22].

Para cada matriz de Müller, $\mathbf{M}$, existe una matriz de coherencia asociada $\mathbf{C}(\mathbf{M})$, cuya característica es ser una matriz hermítica, es decir, es igual a su adjunta (conjugada transpuesta) y es positiva semidefinida. Ello significa que sus valores propios son mayores o iguales a cero. Esto es de suma importancia ya que los valores propios de la matriz de coherencia son los coeficientes que ponderan las matrices puras en las que se descompone $\mathbf{M}$. Esta se define explícitamente en la ref. [23] como:

$$\mathbf{M} = \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{bmatrix}$$

$$\mathbf{C}(\mathbf{M}) = \frac{1}{2} \begin{bmatrix} m_{00}+m_{11} & m_{01}+m_{10} & m_{02}+m_{20} & m_{03}+m_{30} \\ +m_{22}+m_{33} & -i(m_{23}-m_{32}) & +i(m_{13}-m_{31}) & -i(m_{12}-m_{21}) \\ m_{01}+m_{10} & m_{00}+m_{11} & m_{12}+m_{21} & m_{13}+m_{31} \\ +i(m_{23}-m_{32}) & -m_{22}-m_{33} & +i(m_{03}-m_{30}) & -i(m_{02}-m_{20}) \\ m_{02}+m_{20} & m_{12}+m_{21} & m_{00}-m_{11} & m_{23}+m_{32} \\ -i(m_{13}-m_{31}) & -i(m_{03}-m_{30}) & +m_{22}-m_{33} & +i(m_{01}-m_{10}) \\ m_{03}+m_{30} & m_{13}+m_{31} & m_{23}+m_{32} & m_{00}-m_{11} \\ +i(m_{12}-m_{21}) & +i(m_{02}-m_{20}) & -i(m_{01}-m_{10}) & -m_{22}+m_{33} \end{bmatrix} \quad (3.7)$$

CAPÍTULO 3. DISEÑO DEL POLARÍMETRO DE MÜLLER

Las matrices de coherencia $\mathbf{C}$ y de Müller $\mathbf{M}$ contienen la misma información polarimétrica. Sin embargo, el que $\mathbf{C}$ sea una matriz positiva semidefinida hermitiana (PSDH), permite diagonalizarla de acuerdo con:

$$\mathbf{C} = \mathbf{U}\boldsymbol{\lambda}\mathbf{U}^\dagger \quad (3.8)$$

Donde $\mathbf{U}$ es una matriz unitaria que contiene a los vectores propios de $\mathbf{C}$ como columnas y $\boldsymbol{\lambda}$ es la matriz diagonal conformada por los valores propios de $\mathbf{C}$. Finalmente $\mathbf{U}^\dagger$ es la matriz adjunta (conjugada transpuesta) de $\mathbf{U}$. Los valores propios λ_i y sus vectores asociados $\vec{e}_i$ deben ordenarse de la siguiente forma:

$$\lambda_0 \geq \lambda_1 \geq \lambda_2 \geq \lambda_3 \quad (3.9)$$

$$\mathbf{C} = \begin{bmatrix} \vec{e}_0 & \vec{e}_1 & \vec{e}_2 & \vec{e}_3 \end{bmatrix} \begin{bmatrix} \lambda_0 & 0 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & \lambda_2 & 0 \\ 0 & 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} \vec{e}_0^{*T} \\ \vec{e}_1^{*T} \\ \vec{e}_2^{*T} \\ \vec{e}_3^{*T} \end{bmatrix} \quad (3.10)$$

El símbolo $*$ representa el conjugado y T la transpuesta del vector.

Debido a las propiedades de $\mathbf{C}$, sus cuatro vectores propios son ortogonales entre sí. Cada uno se asocia a un mecanismo de dispersión de la luz en una dirección específica. La matriz de coherencia equivale a tener cuatro elementos puros en serie. Estos modifican la luz en direcciones ortogonales, ponderados por la magnitud de su valor propio asociado.

Dado que los valores propios negativos de una matriz de coherencia representarían una aportación negativa de uno de estos mecanismos de dispersión, no tienen sentido físico y necesariamente provienen de errores experimentales. Entonces podemos eliminar su aportación sustituyéndolos por cero. Por otro lado, los valores positivos indican la aportación de mecanismos que son transformaciones puras de la polarización, por

3.2. FILTRADO DE LOS RESULTADOS EXPERIMENTALES

lo que se espera que estos valores sean mucho mayores que los valores negativos. Esta relación de magnitudes es interpretada como la relación señal a ruido SNR, del inglés (*signal to noise ratio*) de la matriz experimental. Aunque no existe una convención de la SNR mínima para validar los datos crudos, se espera que al menos λ_0 sea un valor positivo y/o que la suma de valores positivos sea mayor a la suma de valores absolutos negativos. Si esto no se cumple, es mejor descartar los datos por estar demasiado contaminados de ruido y no ser confiables.

Cuando se han sustituido los valores propios negativos por cero, se recalcula la matriz de coherencia que ya ha sido *filtrada* de transformaciones carentes de interpretación física. El subíndice *filt* indica que ya se ha asegurado que se cumple lo indicado en la expresión 3.12.

$$\mathbf{C}_{filt} = \mathbf{U} \boldsymbol{\lambda}_{filt} \mathbf{U}^\dagger$$

$$\mathbf{C}_{filt} = \begin{bmatrix} \vec{e}_0 & \vec{e}_1 & \vec{e}_2 & \vec{e}_3 \end{bmatrix} \begin{bmatrix} \lambda_{filt0} & 0 & 0 & 0 \\ 0 & \lambda_{filt1} & 0 & 0 \\ 0 & 0 & \lambda_{filt2} & 0 \\ 0 & 0 & 0 & \lambda_{filt3} \end{bmatrix} \begin{bmatrix} \vec{e}_0^{*T} \\ \vec{e}_1^{*T} \\ \vec{e}_2^{*T} \\ \vec{e}_3^{*T} \end{bmatrix} \quad (3.11)$$

$$\lambda_{filt0} \geq \lambda_{filt1} \geq \lambda_{filt2} \geq \lambda_{filt3} \geq 0 \quad (3.12)$$

La matriz de coherencia filtrada $\mathbf{C}_{filt}$ se transforma de vuelta en una matriz de Müller de la siguiente manera:

$$\mathbf{C}_{filt} = \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{01}^* & c_{11} & c_{12} & c_{13} \\ c_{02}^* & c_{12}^* & c_{22} & c_{23} \\ c_{03}^* & c_{13}^* & c_{23}^* & c_{33} \end{bmatrix}$$

$$\mathbf{M}(\mathbf{C}_{filt}) = \frac{1}{2} \begin{bmatrix} c_{00}+c_{11} & c_{01}+c_{01}^* & c_{02}+c_{20}^* & c_{03}+c_{03}^* \\ +c_{22}+c_{33} & -i(c_{23}-c_{23}^*) & +i(c_{13}-c_{13}^*) & -i(c_{12}-c_{12}^*) \\ c_{01}+c_{01}^* & c_{00}+c_{11} & c_{12}+c_{12}^* & c_{13}+c_{13}^* \\ +i(c_{23}-c_{23}^*) & -c_{22}-c_{33} & +i(c_{03}-c_{03}^*) & -i(c_{02}-c_{02}^*) \\ c_{02}+c_{02}^* & c_{12}+c_{12}^* & c_{00}-c_{11} & c_{23}+c_{23}^* \\ -i(c_{13}-c_{13}^*) & -i(c_{03}-c_{03}^*) & +c_{22}-c_{33} & +i(c_{01}-c_{01}^*) \\ c_{03}+c_{03}^* & c_{13}+c_{13}^* & c_{23}+c_{23}^* & c_{00}-c_{11} \\ +i(c_{12}-c_{12}^*) & +i(c_{02}-c_{02}^*) & -i(c_{01}-c_{01}^*) & -c_{22}+c_{33} \end{bmatrix} \quad (3.13)$$

La matriz $\mathbf{M}(\mathbf{C}_{filt})$, ya es una matriz de Müller que cumple los criterios de pasividad y de Stokes mencionados en el capítulo 2.

La implementación e integración de estas tres técnicas se describe detalladamente a continuación. En primer lugar se muestra el montaje experimental del polarímetro descrito en la primera sección de este capítulo, así como los componentes de hardware y software para controlarlos. El siguiente paso es el preprocesamiento de la información, que incluye la calibración y el filtrado de los resultados. Ello nos asegura que nuestros datos son fiables y que cumplen con las restricciones matemáticas de las matrices de Müller.

3.3. Interpretación de la matriz de Müller

Para interpretar la información contenida en la matriz de Müller, visualizamos tres magnitudes ópticas: diatenuación, retardancia y depolarización. La técnica para obtener estas magnitudes es la descomposición polar de matrices de Müller depolarizantes propuesta por Lu y Chipman [24]. Este procedimiento nos permite analizar nuestra muestra como un elemento capaz de convertir luz completamente polarizada en luz parcialmente polarizada. Esto se espera para el caso específico en el que la muestra es una neurona ya que se compone de estructuras irregulares que pueden dispersar la polarización.

3.3. INTERPRETACIÓN DE LA MATRIZ DE MÜLLER

La diatenuación cambia la intensidad de los estados de polarización. El polarizador lineal es un ejemplo de diatenuador ya que deja pasar la luz cuya polarización está alineada al eje de transmisión del polarizador. Además, atenúa por completo la polarización ortogonal a su eje de transmisión. [12]

La magnitud de la diatenuación (**D**), se define como:

$$\mathbf{D} \equiv \frac{T_{max} - T_{min}}{T_{max} + T_{min}} \quad (3.14)$$

T_{max} y T_{min} son las transmitancias máxima y mínima, o sea, los cocientes de la intensidad de haz de salida I_{out} entre la intensidad del haz incidente I_{in} .

$$T \equiv \frac{I_{out}}{I_{in}} \quad (3.15)$$

La retardancia desfasa una componente del vector de campo eléctrico. Su magnitud R nos indica el desplazamiento de una componente respecto a la otra. Típicamente se expresa como un ángulo en términos de la longitud de onda λ , o mediante su medida en radianes $2\pi[\text{rad}] = \lambda[\text{nm}]$. Este desfase puede ser un adelanto o un atraso de una componente sobre otra (ortogonal). Por eso la retardancia tiene el rango de valores: $0 \leq R \leq \pi$ y se define como: [24]

$$R \equiv |\delta_q - \delta_r| \quad (3.16)$$

donde δ_q y δ_r son los cambios rápidos de fase producidos en el estado de polarización alineado al eje rápido del retardador y su estado ortogonal respectivamente.

Estas dos propiedades son *polarimétricamente puras*, es decir, son transformaciones lineales de un estado de polarización a otro y no modifican el grado de polarización (DoP).

CAPÍTULO 3. DISEÑO DEL POLARÍMETRO DE MÜLLER

La depolarización en cambio es la capacidad del elemento para quitar polarización de un haz de luz incidente. Se utiliza el parámetro Δ para describirla y toma valores de 0 a 1, que indica el *poder* depolarizante del objeto. El valor 0 indica que no se afecta el DoP; el valor 1 significa que reduce el DoP en un cien por ciento [12].

La depolarización cambia la dirección de vibración de cada fotón, pero este cambio no es completamente aleatorio. Esto puede favorecer una dirección preferencial entre ellos, por lo que también actúa de cierta manera como un polarizador. Esta propiedad es llamada polarizancia, cuya magnitud es el módulo del vector $\vec{P}$ descrito a continuación como parte el método para descomponer la matriz de Müller. [24]

3.3.1. Algoritmo de Lu y Chipman

La matriz de Müller puede representarse como el producto de tres matrices de Müller, con propiedades ópticas distintas cada una.

$$\mathbf{M} = \mathbf{M}_{\Delta} \mathbf{M}_{\mathbf{R}} \mathbf{M}_{\mathbf{D}} \quad (3.17)$$

$\mathbf{M}$ es la matriz de Müller original, $\mathbf{M}_{\mathbf{D}}$ solo tiene diatenuación; $\mathbf{M}_{\mathbf{R}}$ solo presenta retardancia y $\mathbf{M}_{\Delta}$ es un remanente que representa la depolarización de $\mathbf{M}$ y a su vez, la polarizancia presente en un elemento depolarizador.

La estructura de las cuatro matrices en la ecuación 3.17 es:

$$\mathbf{M} = m_{00} \begin{bmatrix} 1 & \vec{D}^T \\ \vec{P} & \mathbf{m} \end{bmatrix} = \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{bmatrix} \quad (3.18)$$

$$\mathbf{M}_{\mathbf{D}} = m_{00} \begin{bmatrix} 1 & \vec{D}^T \\ \vec{D} & \mathbf{m}_{\mathbf{D}} \end{bmatrix} \quad (3.19)$$

3.3. INTERPRETACIÓN DE LA MATRIZ DE MÜLLER

$$\mathbf{M}_{\mathbf{R}} = \begin{bmatrix} 1 & \vec{0}^T \\ \vec{0} & \mathbf{m}_{\mathbf{R}} \end{bmatrix} \quad (3.20)$$

$$\mathbf{M}_{\Delta} = \begin{bmatrix} 1 & \vec{0}^T \\ \vec{P}_{\Delta} & \mathbf{m}_{\Delta} \end{bmatrix} \quad (3.21)$$

El primer paso consiste obtener los vectores de diatenuación y polarizancia, $\vec{D}$ y $\vec{P}$ respectivamente, a partir de 3.18:

$$\vec{D} = \frac{1}{m_{00}} \begin{bmatrix} m_{01} \\ m_{02} \\ m_{03} \end{bmatrix} \quad ; \quad \vec{P} = \frac{1}{m_{00}} \begin{bmatrix} m_{10} \\ m_{20} \\ m_{30} \end{bmatrix} \quad (3.22)$$

La magnitud de la diatenuación solo depende del vector de diatenuación, por lo tanto se define como:

$$\mathbf{D} = \frac{\sqrt{m_{01}^2 + m_{02}^2 + m_{03}^2}}{m_{00}} \quad ; \quad 0 \leq \mathbf{D} \leq 1 \quad (3.23)$$

Después calculamos $\mathbf{m}_{\mathbf{D}}$ con la siguiente fórmula, donde $\mathbf{I}_3$ es la matriz identidad 3×3 :

$$\mathbf{m}_{\mathbf{D}} = \sqrt{1 - \mathbf{D}^2} \mathbf{I}_3 + \frac{1 - \sqrt{1 - \mathbf{D}^2}}{\mathbf{D}^2} \vec{D} \vec{D}^T \quad (3.24)$$

En este punto tenemos todos los elementos necesarios para construir la matriz de diatenuación $\mathbf{M}_{\mathbf{D}}$ basándonos en la expresión 3.19. El siguiente paso consiste en calcular las matrices de depolarización y de retardancia.

CAPÍTULO 3. DISEÑO DEL POLARÍMETRO DE MÜLLER

Llamaremos $\mathbf{M}'$ a la matriz que resulta de multiplicar la matriz original por la inversa de su matriz de diatenuación:

$$\mathbf{M}' \equiv \mathbf{M}\mathbf{M}_D^{-1} \quad (3.25)$$

Nota: Una manera simple de interpretar la multiplicación de matrices de Müller es pensarla como una forma de combinar los cambios producidos por estas. Cuando multiplicamos una matriz por otra, los efectos se 'suman'. Es como aplicar una serie de transformaciones que se acumularan para producir un resultado final. Por otro lado, multiplicar por la matriz inversa, 'deshace' o 'resta' esos efectos, lo que nos permite regresar al estado original o neutralizar las transformaciones previas.

$\mathbf{M}'$ contiene depolarización y retardancia. Expresadas de la siguiente forma:

$$\mathbf{M}' = \mathbf{M}_\Delta \mathbf{M}_R = \begin{bmatrix} 1 & \vec{0}^T \\ \vec{P}_\Delta & \mathbf{m}' \end{bmatrix} = \begin{bmatrix} 1 & \vec{0}^T \\ \vec{P}_\Delta & \mathbf{m}_\Delta \mathbf{m}_R \end{bmatrix} = \begin{bmatrix} 1 & \vec{0}^T \\ \vec{P}_\Delta & \mathbf{m}_\Delta \end{bmatrix} \begin{bmatrix} 1 & \vec{0}^T \\ \vec{0} & \mathbf{m}_R \end{bmatrix} \quad (3.26)$$

Además las submatrices 3×3 $\mathbf{m}'$ y $\mathbf{m}_\Delta$ tienen la siguiente relación:

$$\mathbf{m}_\Delta^2 = \mathbf{m}'(\mathbf{m}')^T \quad (3.27)$$

Si los valores propios de $\mathbf{m}_\Delta^2$ son λ_1 , λ_2 y λ_3 , el teorema de Cayley-Hamilton predice que los valores propios de $\mathbf{m}_\Delta$ son $\sqrt{\lambda_1}$, $\sqrt{\lambda_2}$ y $\sqrt{\lambda_3}$. De ahí que la expresión para obtener $\mathbf{m}_\Delta$ es:

$$\mathbf{m}_\Delta = \pm \left[\mathbf{m}'(\mathbf{m}')^T + \left(\sqrt{\lambda_1 \lambda_2} + \sqrt{\lambda_2 \lambda_3} + \sqrt{\lambda_3 \lambda_1} \right) \mathbf{I}_3 \right]^{-1} \times \left[\left(\sqrt{\lambda_1} + \sqrt{\lambda_2} + \sqrt{\lambda_3} \right) \mathbf{m}'(\mathbf{m}')^T + \sqrt{\lambda_1 \lambda_2 \lambda_3} \mathbf{I}_3 \right] \quad (3.28)$$

3.3. INTERPRETACIÓN DE LA MATRIZ DE MÜLLER

El signo de la expresión 3.28 depende del determinante de $\mathbf{m}'$. Si el determinante es mayor a cero, corresponde al signo positivo de la expresión. De otra manera deberá aplicarse el signo negativo.

La última parte que requerimos para completar $\mathbf{M}_\Delta$ es el vector $\vec{P}_\Delta$. Este se obtiene como:

$$\vec{P}_\Delta = \frac{\vec{P} - \mathbf{m}\vec{D}}{1 - \mathbf{D}^2} \quad (3.29)$$

Donde $\mathbf{m}$ es la submatriz 3×3 de $\mathbf{M}$ que aparece en la expresión 3.18.

De la matriz de depolarización $\mathbf{M}_\Delta$, extraemos la magnitud de depolarización con la siguiente fórmula:

$$\Delta = 1 - \frac{|\text{tr}(\mathbf{M}_\Delta) - 1|}{3} \quad ; \quad 0 \leq \Delta \leq 1 \quad (3.30)$$

Finalmente, encontramos la matriz de retardancia $\mathbf{M}_R$ como:

$$\mathbf{M}_R = \mathbf{M}_\Delta^{-1} \mathbf{M}' \quad (3.31)$$

La magnitud de la retardancia se obtiene con la expresión:

$$\mathbf{R} = \cos^{-1} \left(\frac{\text{tr}(\mathbf{M}_R)}{2} - 1 \right) \quad ; \quad 0 \leq \mathbf{R} \leq \pi \quad (3.32)$$

Las magnitudes de las expresiones 3.23, 3.30 y 3.32 nos permiten visualizar los resultados en forma de imágenes, en las que cada pixel expresa el valor de la magnitud que se está representando. El rango de cada magnitud debe ser escalado a un rango de 0 a 255 para formar imágenes de 8 bits de profundidad.

CAPÍTULO 3. DISEÑO DEL POLARÍMETRO DE MÜLLER

Capítulo 4

Implementación experimental del polarímetro

4.1. Hardware

La figura 4.1 ilustra la configuración del polarímetro de doble retardador rotador (DRR) construido en el laboratorio. El sistema consta de:

- Fuente de luz: LED rojo (620 nm).
- PSG y PSA: Polarizadores LPVIS100-MP2 y retardadores AQWP05M-600 (Thorlabs).
- Detección: Cámara CMOS (AmScope MU043M-FL).

Los polarizadores y retardadores fueron colocados en monturas motorizadas (actuadores) para controlar su rotación con precisión. El fotosensor es alimentado y comunicado mediante un puerto USB 3.0 a una computadora. Todo el arreglo está fijo en una estructura de aluminio y cubierto con una carcasa fotodensa impresa en PLA.

El arreglo óptico se hizo en transmisión, disponiendo sus elementos en forma vertical como lo ilustra la Figura 4.1

CAPÍTULO 4. IMPLEMENTACIÓN EXPERIMENTAL DEL POLARÍMETRO

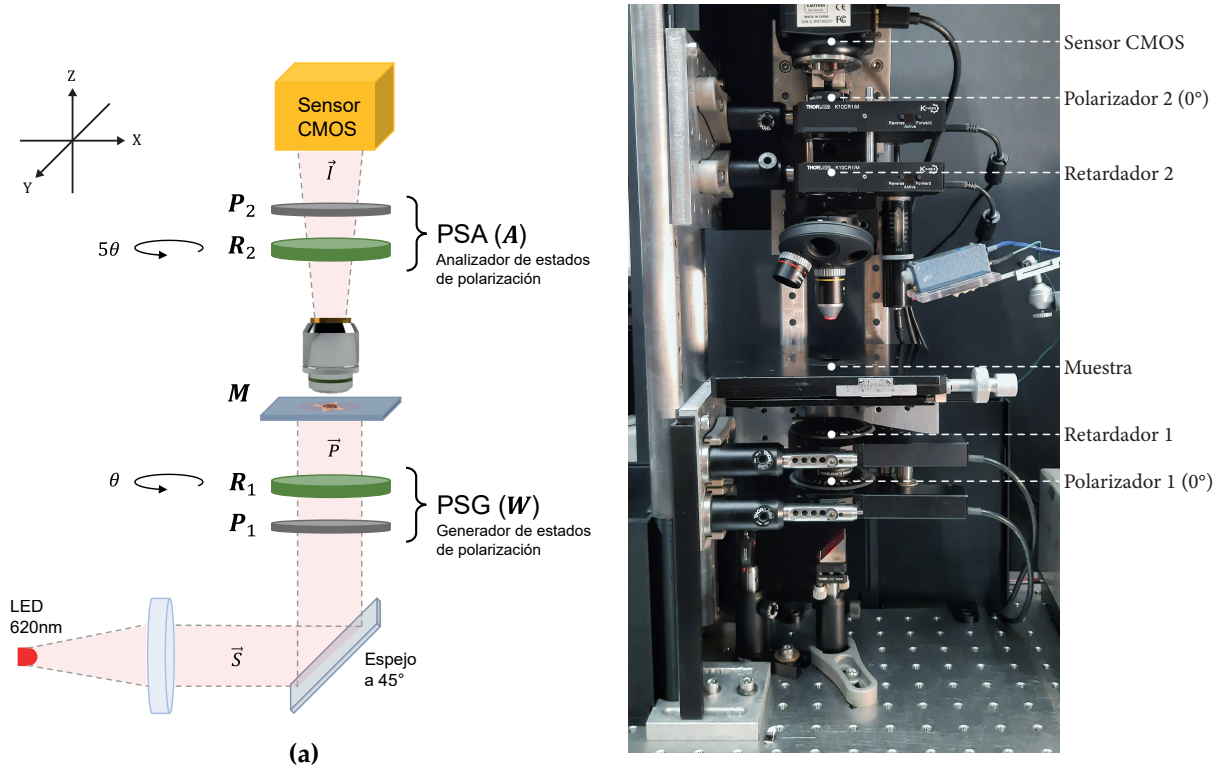


Figura 4.1: Diagrama y fotografía de la configuración experimental del polarímetro.

El instrumento cuenta con una platina XY en la que se coloca y manipula la muestra. Usamos un objetivo Nikon 10x con una apertura numérica N/A : de 0.25 y tener la suficiente distancia de trabajo en el eje Z que nos permite encuadrar la muestra. La región de interés (ROI) es de 256x256 píxeles, equivalentes a un área de 200x200 [μm]. El objetivo se desliza sobre el eje Z con un controlador de movimiento lineal Newport.

El sistema de control, captura y despliegue, usa dos computadoras PC y un microcontrolador (Arduino Nano) como interfaz TTL-USB para la señal de control. La primera computadora cuenta con sistema Windows 10 para ejecutar las secuencias de los actuadores de los elementos ópticos, esto es: alinear los polarizadores y retardadores sobre un mismo eje (0°); y rotar a la vez, el primer y segundo retardador con incrementos $\theta = 5^\circ$ y el segundo con incrementos $5\theta = 25^\circ$, respectivamente. La segunda PC opera en Linux, y tiene mejores prestaciones en cuanto a procesador, memoria RAM y almacenamiento para adquirir hasta 520 imágenes (540x720) por segundo.

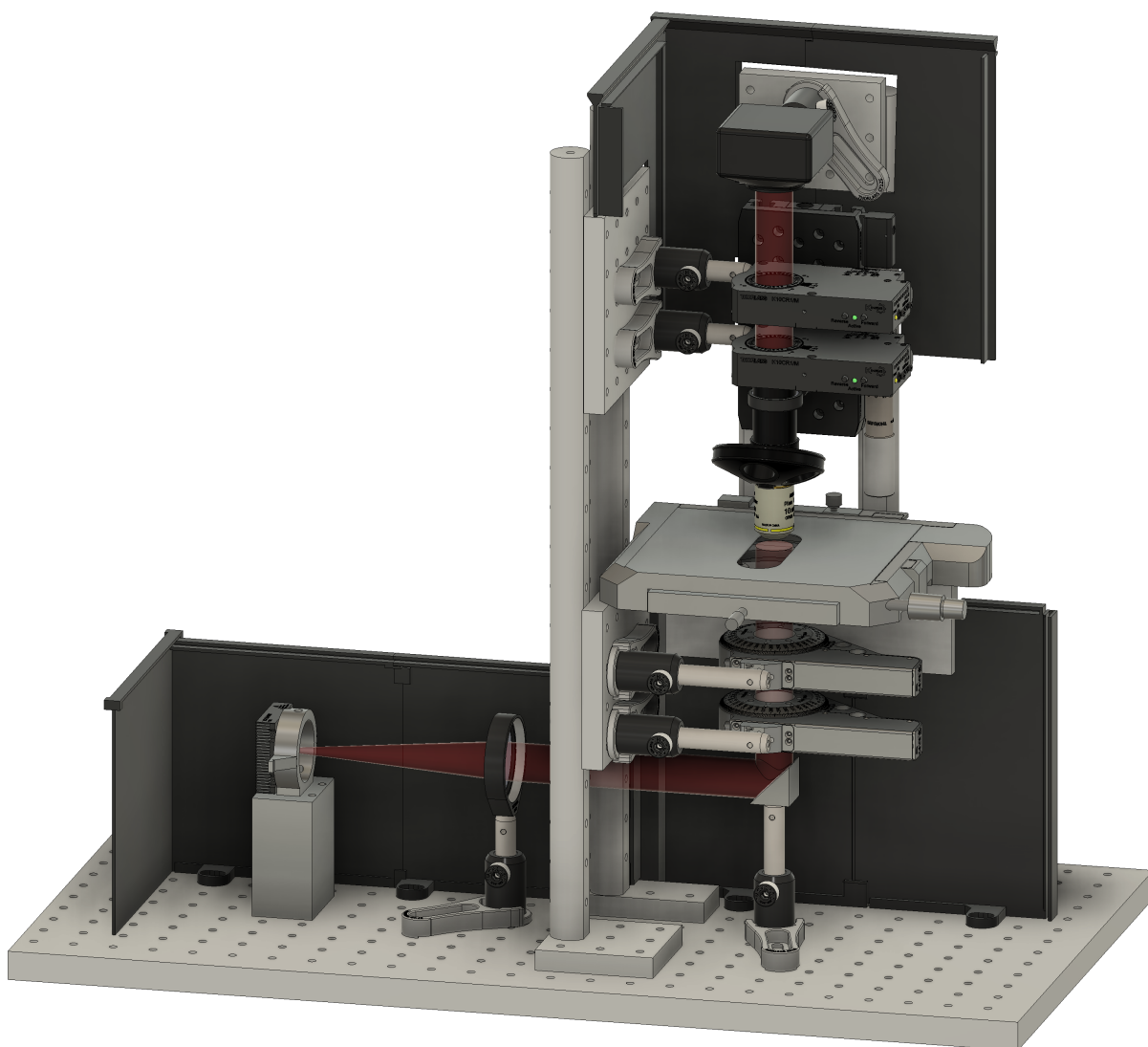


Figura 4.2: Modelo 3D del polarímetro. Vista axonométrica.

4.2. Software

Siguiendo el método de DRR descrito en la sección 3.1, el polarímetro adquiere la información de intensidad luminosa en 36 posiciones distintas del PSA y PSG combinados. Cada posición está determinada por el ángulo θ correspondiente al primer retardador. El uso de una cámara fotográfica como detector de intensidad implica que tengamos varios valores de intensidad para un ángulo θ . Entonces existirá una matriz de Müller

CAPÍTULO 4. IMPLEMENTACIÓN EXPERIMENTAL DEL POLARÍMETRO

por uno de los 540×720 píxel del sensor. La cámara cuenta con un obturador global que captura la luz simultáneamente en todos los píxeles, evitando variaciones en los valores de intensidad debido a fluctuaciones de la fuente.

El flujo de trabajo en la operación del polarímetro se ilustra en la figura 4.3. El software fue desarrollado en código abierto usando Python, versión 3.10, y con el paradigma de programación orientada a objetos. Este paradigma permite organizar del código de manera modular. Ello facilita escalar el software si fuera necesario agregar más funcionalidades a futuro. El código fuente se incluye en el apéndice A. El software se compone de tres bloques funcionales: 1) Control de la cámara y actuadores, 2) Procesamiento de datos y 3) Interfaz gráfica.

4.2.1. Control de la cámara

El primer bloque ejecuta la rutina de adquisición. Inicia alineando a los componentes en la posición inicial de 0° , también denominada máxima transmisión, ya que la intensidad que llega al sensor será la máxima de entre todas las posiciones del PSG y PSA. Una vez ahí, se abre una vista previa en tiempo real de lo que detecta la cámara para posicionar y enfocar la muestra. Una vez que la muestra está colocada, el experimentador inicia la secuencia de adquisición de los actuadores, que consiste en repetir 36 veces lo siguiente: 1) enviar una señal de control a la cámara para tomar la fotografía; 2) rotar $\theta = 5^\circ$ el primer retardador y $5\theta = 25^\circ$ el segundo para avanzar a la siguiente posición; 3) Regresar a 1.

4.2.2. Procesamiento de las 36 imágenes

El bloque de procesamiento de datos contiene los atributos y métodos de control del flujo general del programa y para procesar la información después de la adquisición. Al iniciarlo ejecuta las tareas a continuación mencionadas:

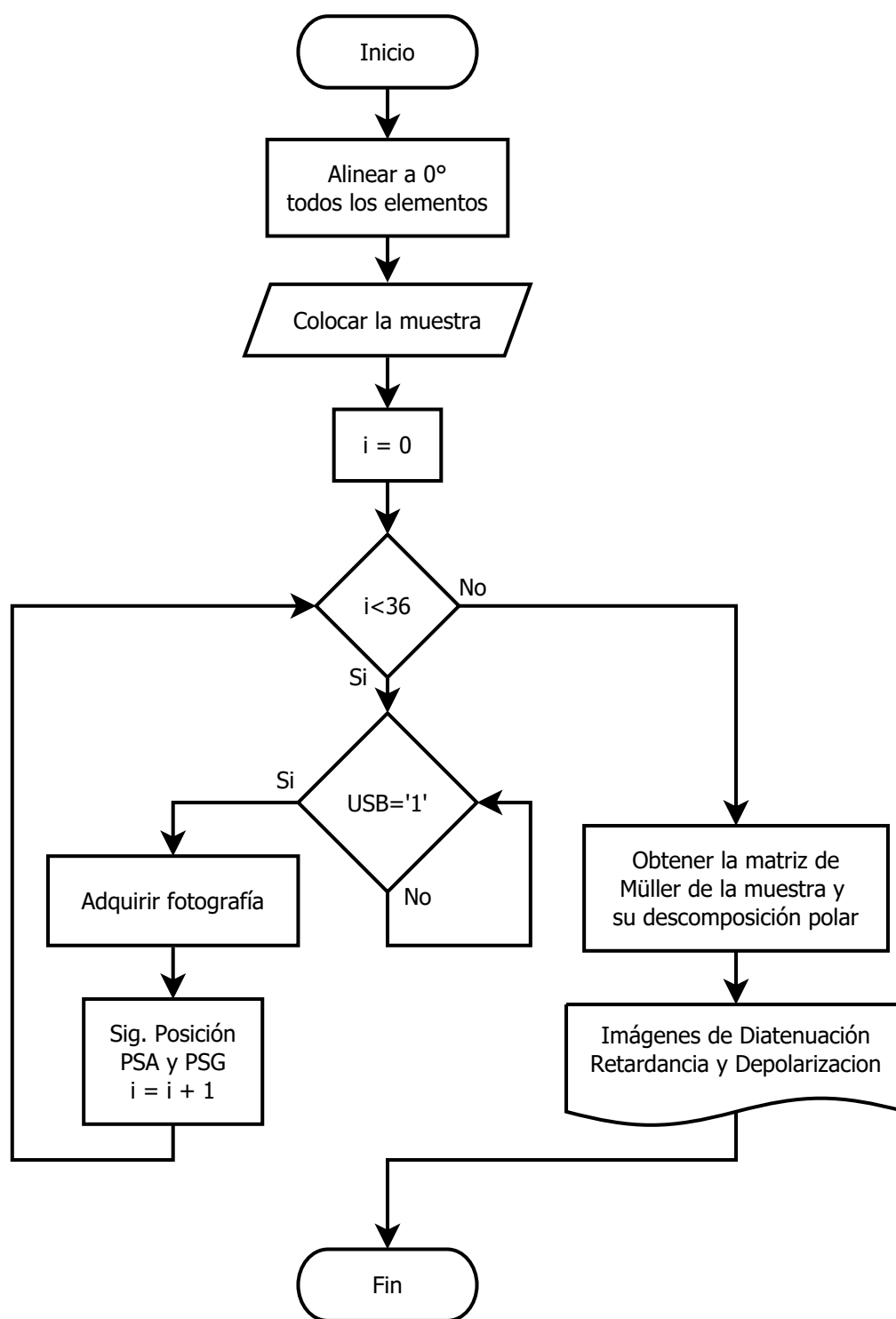


Figura 4.3: Diagrama de flujo de la operación del polarímetro.

CAPÍTULO 4. IMPLEMENTACIÓN EXPERIMENTAL DEL POLARÍMETRO

1. **Inicio**
2. **Instancia de nuevo experimento:** asigna un nombre o id al experimento nuevo, crea los directorios necesarios para el almacenamiento de los datos en disco duro y controla el inicio de la ejecución de la rutina de adquisición.
3. **Cálculo de la matriz de Müller:** Lee los valores de intensidad de las 36 imágenes y calcula la matriz de Müller para cada píxel del sensor. Devuelve la matriz **M** de tamaño $540 \times 720 \times 16$, que corresponde al numero de píxeles en filas y columnas del sensor y los elementos de su matriz asociada.
4. **Filtrado:** Recibe como entrada a la matriz **M** y mediante el algoritmo de la sección 3.2 convierte la matriz de Müller de cada píxel en su matriz de coherencia correspondiente, la cual se diagonaliza y sustituyen sus valores propios negativos por cero. Devuelve la matriz **M** asegurando que cumple con los criterios de Stokes y de pasividad.
5. **Normalización:** Recibe la matriz **M** y divide los elementos de la matriz de Müller de cada píxel entre su primer elemento m_0 . Devuelve a la matriz **M** con un rango de valores comprendidos entre 0 y 1.
6. **Descomposición:** Toma una matriz **M** filtrada y normalizada para obtener la magnitud de diatenuación, retardancia y depolarización de cada píxel siguiendo el método descrito en 3.3. Retorna una tupla de 3 matrices de tamaño 540×720 , que corresponden a la magnitud de las propiedades ópticas mencionadas.
7. **Fin**

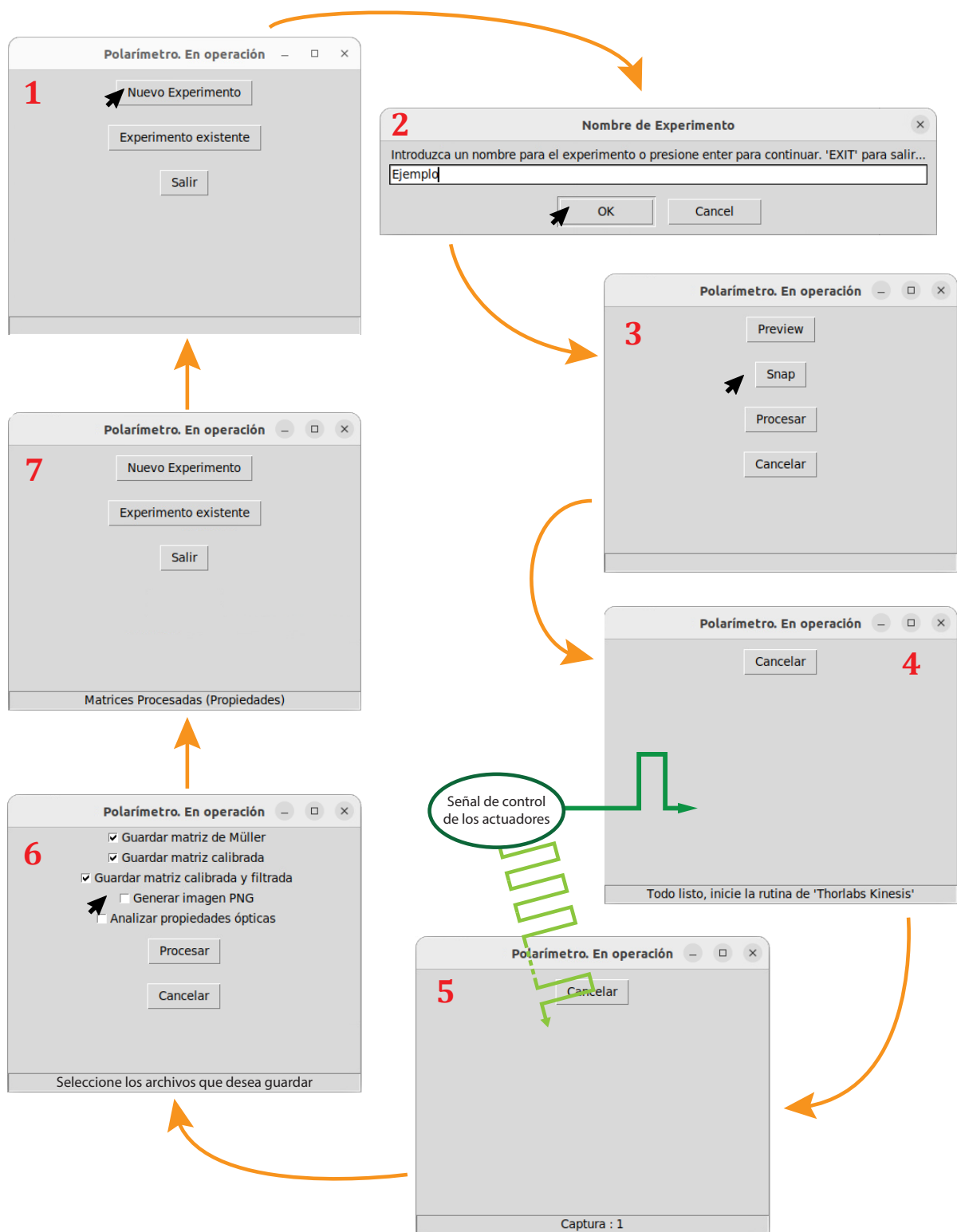


Figura 4.4: Interfaz gráfica del software a través del flujo de operación del polarímetro.

CAPÍTULO 4. IMPLEMENTACIÓN EXPERIMENTAL DEL POLARÍMETRO

El tercer componente del software es una interfaz gráfica (GUI) que integra los dos bloques anteriores y brinda un medio de interacción humano-computadora. Guía al usuario a través del flujo de trabajo, desplegando las indicaciones a seguir y acciones que se encuentran disponibles.

La figura 4.4 muestra cómo se comporta la interfaz durante un experimento. La rutina inicia en la ventana número 1 en donde elegimos las opciones de crear un nuevo experimento o cargar la información de uno ya existente. Si se selecciona la segunda opción, el programa salta la etapa de adquisición y va directamente a la ventana 6. Al crear un nuevo experimento la interfaz muestra la ventana 2, donde se solicita al usuario ingresar un nombre para identificar y guardar la información. Una vez añadido el nombre, aparece la ventana 3. El botón 'Preview' despliega una visualización en tiempo real de lo que detecta la cámara, permitiendo que el experimentador coloque su muestra en el área de trabajo y con la ayuda de la platina pueda encuadrarla en la imagen. La opción 'Snap' debe seleccionarse cuando la muestra esté correctamente posicionada y enfocada. Esto preparará la cámara y la PC para esperar las señales de control provenientes de los actuadores (ventana 4), que rotarán secuencialmente incrementando el ángulo preestablecido y genera un pulso para la toma de la fotografía en esa posición. Durante este proceso, la ventana 5 nos muestra el contador de capturas durante la adquisición. Una vez tomadas las dieciséis fotografías, la interfaz desplegará una lista de *checkboxes* para seleccionar qué archivos finales se desean guardar en disco duro. En orden en que se muestran en la ventana 6, estos son: matriz de Müller sin procesamiento (.mat), matriz de Müller calibrada (.mat), matriz de Müller filtrada y calibrada (.mat), visualización de la matriz de Müller calibrada y filtrada (.png) y las propiedades ópticas extraídas de la última matriz de Müller que se haya generado (.mat y .png). Durante el procesamiento la interfaz indicará en qué etapa se encuentra y al terminar volverá a mostrar el menú de inicio.

Como se ilustra en la figura 4.4, nuestro software de adquisición depende de las señales de control que envían los actuadores que rotan los elementos ópticos, ya que el

software que los controla se ejecuta en otra PC. En la siguiente sección de este capítulo se describirá cómo y porqué se integra de esta forma el sistema completo.

Consideraciones técnicas del software

- La exposición y ganancia del sensor fotográfico en la función 'Preview' se determinan por la cámara de manera automática, pero en la función de adquisición 'Snap' estos valores se deben fijar porque su variación introduce errores en la intensidad modulada que queremos obtener.
- Debido a que el nombre del experimento debe ser un identificador único no debe usarse el mismo nombre en otro, ya que la información existente se sobrescribirá, perdiendo la posibilidad de recuperar los datos del experimento anterior.
- La función de adquisición 'Snap' debe su nombre a que la cámara toma una sola fotografía por cada posición de los actuadores. Existe otra función en el código llamada 'Record' que puede implementarse como otro botón en la interfaz gráfica. Esta permite tomar una secuencia de imágenes por cada posición de los actuadores, dando como resultado una secuencia temporal de matrices de Müller. El diseño experimental determina que tipo de adquisición usar, como se discute en el siguiente capítulo.

4.3. Integración del sistema

Como se mencionó al inicio de este capítulo, el polarímetro es controlado por dos computadoras (PC's). Una se encarga del control de los actuadores que rotan los elementos ópticos; la otra controla la cámara y procesa los datos adquiridos.

El fabricante de los actuadores (Thorlabs) provee una aplicación de escritorio con interfaz gráfica llamada 'Kinesis' para el control de sus actuadores. Esta existe solo para el

CAPÍTULO 4. IMPLEMENTACIÓN EXPERIMENTAL DEL POLARÍMETRO

sistema operativo Windows y nos permite crear secuencias de acciones que ejecutarán los actuadores, incluyendo: alineación de todos ellos a una misma posición de referencia (home), movimientos a un ángulo concreto a partir de esta referencia (move), incrementos en la rotación (step), generación de señales lógicas de salida (out), entre otras como estructuras de control y tiempos de espera.

La cámara puede ser controlada por la aplicación de escritorio AmLite disponible para sistemas Windows, Linux y MacOS, proveída por el fabricante AmScope. Sin embargo solo brinda funciones básicas y no soporta la utilización de rutinas ni la integración con otros programas. Por esta razón se optó por utilizar el SDK (software development kit) disponible en la misma página del fabricante. Este brinda los archivos de bibliotecas para controlar la cámara en cualquiera de estos tres sistemas operativos e incluye ejemplos en diferentes lenguajes de programación (incluido Python). El modelo de cámara empleado en el polarímetro tiene la capacidad de adquirir con una tasa de hasta 520 fotografías por segundo. Para aprovechar esta característica decidimos controlar la cámara desde un equipo independiente con mayores prestaciones. Además, el sistema Linux nos permitió controlar precisamente los intervalos de captura de cada imagen, que a la máxima velocidad de la cámara es de 2[ms] entre toma y toma, de los cuales 1.5[ms] corresponden a la exposición del sensor.

Debido a la imposibilidad de los actuadores de registrar pulsos de entrada, no existe una comunicación bidireccional entre ambos sistemas. Esta solo ocurre desde los actuadores hacia la cámara. Los actuadores ejecutan la secuencia de acciones mostrada en la Figura 4.5

4.3. INTEGRACIÓN DEL SISTEMA

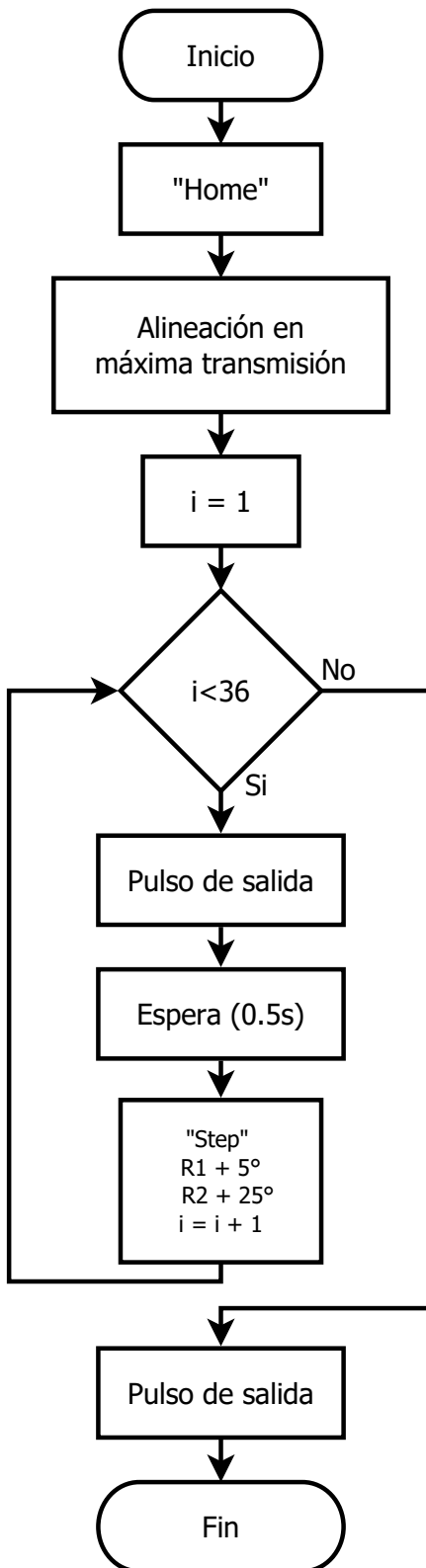


Figura 4.5: Secuencia de acciones de los actuadores

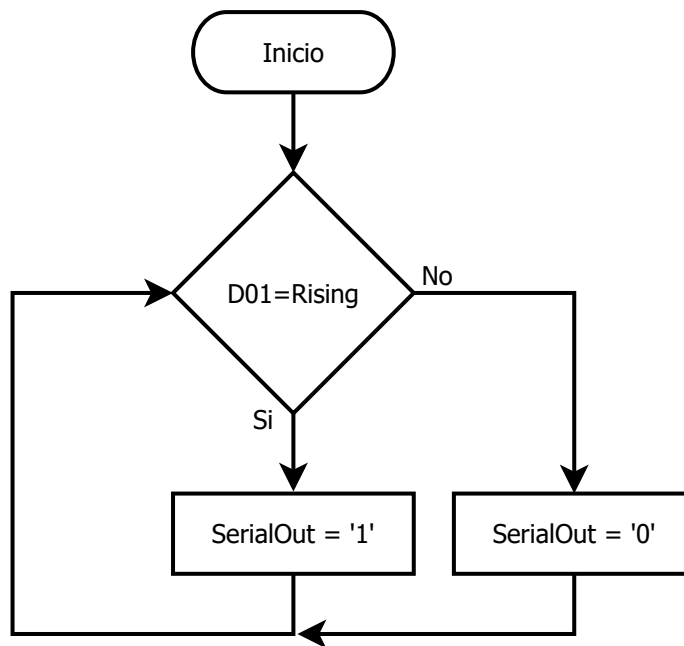


Figura 4.6: Diagrama de flujo del programa que se ejecuta en Arduino

Los pulsos de salida de los actuadores son señales lógicas TTL de 0[V] y 5[V]. Para que la computadora lea estas señales utilizamos un Arduino Nano que convierte la señal de 5[V] en un carácter '1' y lo envía por el puerto USB de la placa. La figura 4.6 muestra el flujo del programa de Arduino.

Capítulo 5

Resultados

5.1. Pruebas. resultados teóricos vs. experimentales

Como requisito indispensable, antes de realizar experimentos con muestras desconocidas, caracterizamos la respuesta del polarímetro con un conjunto de cuatro muestras de referencia, de las cuales conocemos la matriz de Müller y las magnitudes ópticas teóricas. Estas son: Espacio libre (sin muestra), polarizador lineal horizontal, polarizador lineal vertical y Retardador lineal de $1/4$ de onda.

Antes de obtener las magnitudes finales, las matrices de Müller pasan por un par de procesos que refinan y formalizan los datos experimentales. Estas son: la calibración y el filtrado. La calibración del instrumento, contrarresta los errores sistemáticos del equipo, provenientes de la fabricación de los elementos ópticos, la falta de alineación en los ejes o ángulos, la diferencia de sensibilidad entre los píxeles del sensor fotográfico y cualquier otro error constante en los experimentos. Esta etapa fue desarrollada a la par de esta tesis por Isaac Eduardo Chávez Ramírez [19].

La idea general de la calibración de matrices de Müller [25] es la siguiente:

CAPÍTULO 5. RESULTADOS

Muestra de referencia	Matriz de Müller teórica	D	R	Δ
Espacio libre (sin muestra)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	0.000	0.000	0.000
Polarizador lineal horizontal (0°)	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	1.000	0.000	0.000
Polarizador lineal vertical (90°)	$\begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	1.000	0.000	0.000
Retardador lineal $\lambda/4$ (30°)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.25 & 0.43 & -0.87 \\ 0 & 0.43 & 0.75 & 0.50 \\ 0 & 0.87 & -0.50 & 0 \end{bmatrix}$	0.000	1.571	0.000

Tabla 5.1: Muestras de referencia: matrices de Müller y magnitudes ópticas teóricas. Diatenuación (**D**), Retardancia (**R**) y Depolarización (**Δ**)

La matriz de Müller que calcula el polarímetro se define de la siguiente manera:

$$\mathbf{B} = \mathbf{A}\mathbf{M}\mathbf{W} \quad (5.1)$$

Despejando a **M** de la ecuación matricial:

$$\begin{aligned}
 \mathbf{A}\mathbf{M}\mathbf{W} &= \mathbf{B} \\
 \mathbf{A}\mathbf{M}(\mathbf{W}\mathbf{W}^{-1}) &= \mathbf{B}\mathbf{W}^{-1} \\
 \mathbf{A}\mathbf{M} &= \mathbf{B}\mathbf{W}^{-1} \\
 (\mathbf{A}^{-1}\mathbf{A})\mathbf{M} &= \mathbf{A}^{-1}\mathbf{B}\mathbf{W}^{-1} \\
 \mathbf{M} &= \mathbf{A}^{-1}\mathbf{B}\mathbf{W}^{-1} \quad (5.2)
 \end{aligned}$$

5.1. PRUEBAS. RESULTADOS TEÓRICOS VS. EXPERIMENTALES

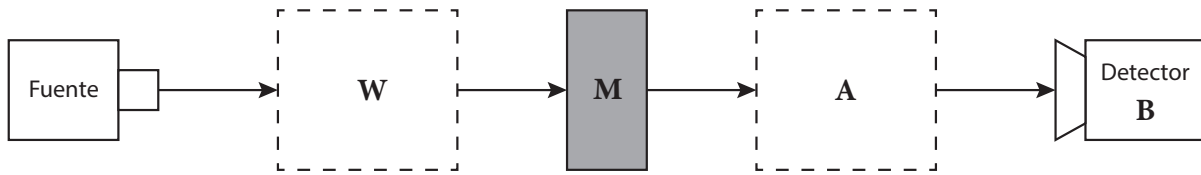


Figura 5.1: Modelado del polarímetro para la calibración. En negritas las matrices de Müller: **W** todo elemento entre la fuente y la muestra, **M** la muestra, **A** todo elemento entre la muestra y el detector y **B** la matriz obtenida por el polarímetro.

Por lo tanto, para obtener la matriz **M** habrá que encontrar las matrices **A** y **W** que sean invertibles y que al multiplicar a **B** eliminen la mayor cantidad de error contenido en ella y que no pertenece a **M**.

Así pues, la calibración es una función de optimización multivariada, en la que se imponen varias restricciones, como la elección del conjunto de matrices de referencia o la condición de que las soluciones sean invertibles. El conjunto de soluciones analíticas para **A** y **W** es un conjunto infinito, pero la solución computacional solo puede ser un conjunto finito. El conjunto de matrices **A** y **W** disponibles, resultado del cálculo numérico, depende principalmente del conjunto de muestras de referencia, además del método empleado, de sus parámetros asociados y de la capacidad de cómputo disponible.

Las matrices de Müller obtenidas por el polarímetro para un solo píxel de las muestras de referencia se presentan en la tabla 5.2. La tabla muestra las matrices experimentales (sin post procesamiento) y las matrices corregidas (calibradas y filtradas). Las propiedades ópticas: Diatenuación (**D**), retardancia (**R**) y depolarización (**Δ**) extraídas de la matriz corregida para cada muestra de referencia (tabla 5.2), se presentan en la tabla 5.3 junto a su valor teórico. Se indican las unidades de cada magnitud entre paréntesis, (rad) para radianes y (1) para indicar que no tiene unidades y se trata de una medida normalizada.

CAPÍTULO 5. RESULTADOS

Muestra de referencia	Matriz de Müller experimental	Matriz de Müller corregida
Espacio libre (sin muestra)	$\begin{bmatrix} 0.783 & -0.109 & 0.006 & 0.007 \\ -0.148 & 1.000 & -0.009 & -0.002 \\ -0.006 & -0.026 & 0.993 & 0.010 \\ 0.004 & 0.002 & 0.000 & 0.7530 \end{bmatrix}$	$\begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$
Polarizador lineal horizontal (0°)	$\begin{bmatrix} 0.541 & 0.830 & -0.100 & 0.020 \\ 0.774 & 1.000 & -0.248 & 0.013 \\ -0.100 & -0.203 & 0.088 & -0.072 \\ 0.014 & -0.005 & 0.024 & 0.010 \end{bmatrix}$	$\begin{bmatrix} 1.000 & 0.954 & 0.083 & 0.000 \\ 0.951 & 0.952 & 0.037 & -0.002 \\ 0.075 & 0.036 & 0.039 & -0.038 \\ 0.011 & -0.009 & 0.060 & -0.004 \end{bmatrix}$
Polarizador lineal vertical (90°)	$\begin{bmatrix} 1.000 & -0.992 & 0.190 & -0.022 \\ -0.983 & 1.000 & -0.151 & 0.029 \\ 0.198 & -0.223 & 0.030 & -0.024 \\ -0.011 & 0.011 & -0.009 & -0.019 \end{bmatrix}$	$\begin{bmatrix} 1.000 & -0.958 & -0.188 & 0.007 \\ -0.967 & 0.967 & 0.185 & -0.003 \\ -0.143 & 0.159 & 0.013 & -0.006 \\ 0.003 & -0.009 & 0.006 & -0.015 \end{bmatrix}$
Retardador lineal $\lambda/4$ a 30°	$\begin{bmatrix} 0.956 & -0.082 & -0.039 & -0.032 \\ -0.131 & 1.000 & 0.447 & 0.496 \\ -0.078 & 0.433 & 0.335 & -0.907 \\ 0.069 & -0.504 & 0.947 & 0.120 \end{bmatrix}$	$\begin{bmatrix} 1.000 & 0.027 & 0.004 & -0.018 \\ 0.004 & 0.535 & 0.447 & 0.636 \\ -0.019 & 0.403 & 0.541 & -0.672 \\ 0.011 & -0.664 & 0.682 & 0.090 \end{bmatrix}$

Tabla 5.2: Muestras de referencia y sus matrices de Müller para un solo píxel.

5.1. PRUEBAS. RESULTADOS TEÓRICOS VS. EXPERIMENTALES

Muestra de referencia	Valores experimentales			Valores teóricos		
	D (1)	R (rad)	Δ (1)	D (1)	R (rad)	Δ (1)
Espacio libre (sin muestra)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Polarizador lineal horizontal (0°)	0.9541	1.6824	0.6974	1.0000	0.0000	0.0000
Polarizador lineal vertical (90°)	0.9774	2.7034	0.6213	1.0000	0.0000	0.0000
Retardador lineal $\lambda/4$ a 30°	0.0455	1.4634	0.0479	0.0000	1.5708	0.0000

Tabla 5.3: Valores experimentales y teóricos de las propiedades ópticas de las muestras analizadas.

Se puede observar en la tabla 5.3, que los valores experimentales de las magnitudes de la retardancia (**R**) y la depolarización (**Δ**) de los polarizadores lineales horizontal y vertical se alejan considerablemente de sus valores teóricos esperados. Las matrices de Müller forman un subconjunto del espacio vectorial de las matrices 4×4 reales. Sin embargo, no todas las matrices 4×4 pueden ser matrices de Müller, ya que estas deben cumplir ciertas restricciones, como representar transformaciones físicamente realizables de la luz. El problema radica en que las matrices de Müller de los polarizadores lineales se encuentran en el límite de lo físicamente posible (Figura 5.2), lo que implica que estamos tratando con un *problema de frontera*. Este tipo de problemas, debido a su naturaleza, son inherentemente complejos y difíciles de resolver. Al aproximar una matriz de Müller con el método empleado (DRR), obtenemos un conjunto de soluciones que se encuentran en el espacio circundante al lugar donde debería ubicarse la solución exacta. A este espacio que rodea la solución exacta (la matriz teórica) se le denomina *vecindad*. Cuando la matriz que estamos tratando de resolver se encuentra en el límite del espacio de las matrices de Müller, como ocurre con los polarizadores lineales, no es posible trazar una vecindad alrededor de la solución exacta para aproximarnos al resultado ideal. Además debemos considerar que estas matrices teóricas tienen un

CAPÍTULO 5. RESULTADOS

determinante igual a cero, lo que implica que las matrices experimentales también presentan determinantes muy cercanos a cero. Esto provoca que pequeñas variaciones en los datos adquiridos generen grandes errores numéricos, ya que cualquier perturbación durante la experimentación puede tener un impacto significativo en la solución debido a la inestabilidad numérica que genera la singularidad de la matriz.

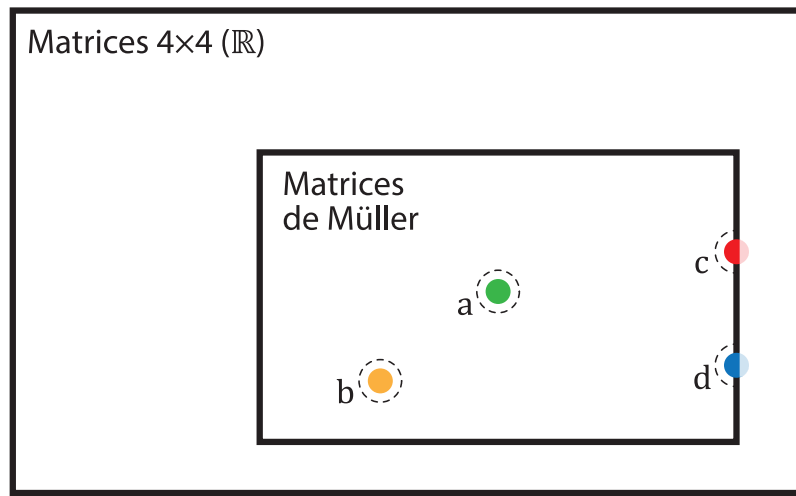


Figura 5.2: Representación del espacio vectorial de las matrices 4×4 reales ($\mathbb{R}$) y del subconjunto que forman las matrices de Müller. Los puntos representan las matrices de las muestras de referencia. a) Aire, b) Placa retardadora $\lambda/4$ a 30° , c) Polarizador lineal a 0° y d) polarizador lineal a 90° . La línea punteada representa la vecindad de cada matriz.

A continuación, se presentan, en forma de visualización gráfica o imágenes, datos de las matrices corregidas y las propiedades ópticas de todos los píxeles, en comparación con su contraparte teórica, en forma de visualización gráfica o imágenes. Esta representación permite una interpretación fácil y rápida de una gran cantidad de datos, en la que los colores nos indican el valor numérico de cada píxel.

5.1. PRUEBAS. RESULTADOS TEÓRICOS VS. EXPERIMENTALES

5.1.1. Visualizaciones gráficas

Por cada píxel en el sensor fotográfico obtenemos una matriz de Müller con 16 elementos, por lo tanto al emplear todos los píxeles de la cámara para realizar mediciones de forma simultánea tendremos en total 388,800 matrices de Müller. Cada una caracteriza una región diferente del espacio que encuadramos. Las visualizaciones de las matrices de Müller (figuras 5.3, 5.6, 5.9 y 5.12) se componen de 16 recuadros, numerados desde m_{11} hasta m_{44} según su fila y columna, cada uno es una imagen de 720×540 píxeles y representa un elemento de las matrices de Müller. Los píxeles se colorean de acuerdo a su valor con la escala indicada por la barra a la derecha.

Espacio libre (sin muestra)

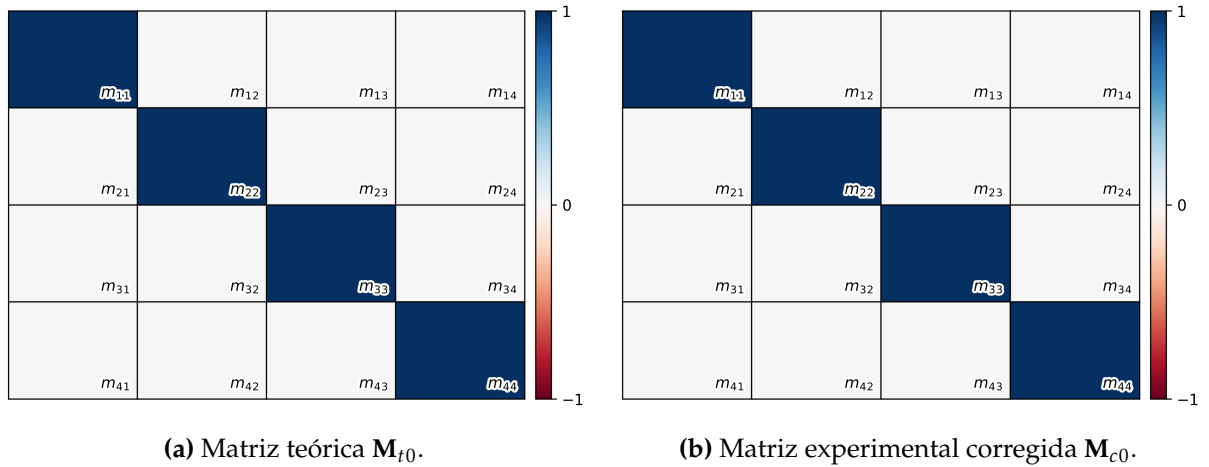


Figura 5.3: Visualización de las matrices de Müller del espacio libre.

Para el espacio libre, las figuras 5.3a y 5.3b muestran que la matriz obtenida por el polarímetro ($\mathbf{M}_{c0}$) iguala a la matriz teórica ($\mathbf{M}_{t0}$). Esto se debe a que los procesos de calibración y filtrado usan a esta muestra como su propia *referencia del error* y los cálculos parten de asumir que ahí se encuentra la totalidad de este error.

Al extraer las propiedades ópticas de $\mathbf{M}_{c0}$, las imágenes resultantes son las mostradas en la Figura 5.4.

CAPÍTULO 5. RESULTADOS

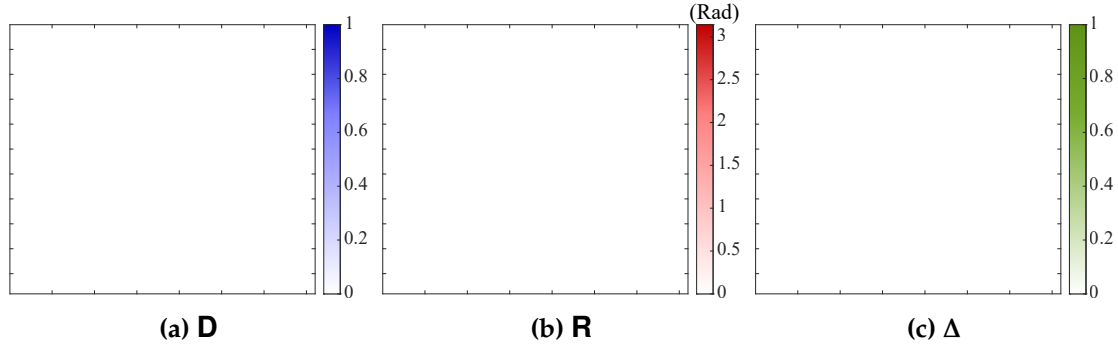


Figura 5.4: Imágenes de las propiedades ópticas de $\mathbf{M}_{c0}$. Diatenuación (**D**), retardancia (**R**) y depolarización (Δ).

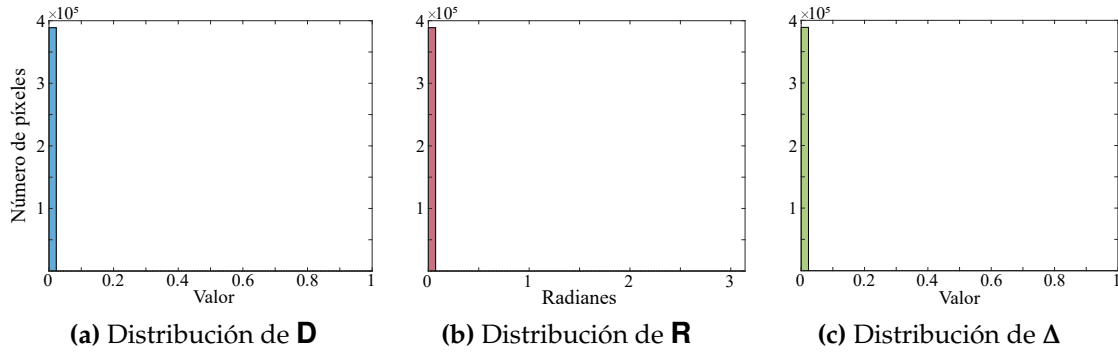


Figura 5.5: Histogramas de las imágenes de las propiedades ópticas de $\mathbf{M}_{c0}$. Diatenuación (**D**), retardancia (**R**) y depolarización (Δ).

Como podemos ver en las imágenes de las propiedades ópticas y sus histogramas en las figuras 5.4 y 5.5, las tres propiedades tienen valores en cero en todos sus píxeles. Esto es de esperarse ya que si las matrices de Müller experimentales eran iguales a las matrices teóricas, las magnitudes de sus propiedades ópticas también serán iguales a las teóricas (Tabla 5.1)

Polarizador lineal horizontal (0°)

A partir de esta muestra se podrán apreciar las diferencias entre los datos teóricos y los experimentales. A simple vista las matrices de Müller de la figura 5.6 son muy parecidas, sin embargo al extraer las propiedades ópticas de $\mathbf{M}_{c1}$ y generar las imágenes

5.1. PRUEBAS. RESULTADOS TEÓRICOS VS. EXPERIMENTALES

de la figura 5.7, la distribución de estos valores (figura 5.8) nos revela que no todas las propiedades ópticas tienen una distribución normal con una media en el valor teórico.

Para la diatenuación, que es la primer propiedad óptica en ser calculada, la figura 5.8a muestra una distribución de valores muy cercanos a su valor teórico que es 1, además de una baja dispersión de los datos. Consideramos este resultado favorable dentro de nuestro rango aceptable de error para una primer versión del instrumento.

La segunda propiedad óptica calculada es la depolarización, que depende del cálculo de la diatenuación. Su distribución, mostrada en la figura 5.8c, revela que el valor medio obtenido es aproximadamente 0.75, bastante alejado del valor teórico que es 0. Esto se debe a la solución escogida en el método de calibración. La función de optimización desvió los datos de esta propiedad hacia valores erróneos, por lo tanto la retardancia, que depende del cálculo de la diatenuación y depolarización, acumula el error de las dos propiedades previas. En la gráfica de distribución de la retardancia 5.8b apreciamos la gran dispersión de los datos debido al error producido en el cálculo de la depolarización. Ya que los valores de esta propiedad son tan variables de pixel a pixel y tienen error acumulado de propiedades anteriores, no podemos tomar como válidos estos resultados para compararlos con los teóricos ni seguir aplicando algoritmos de corrección del error.

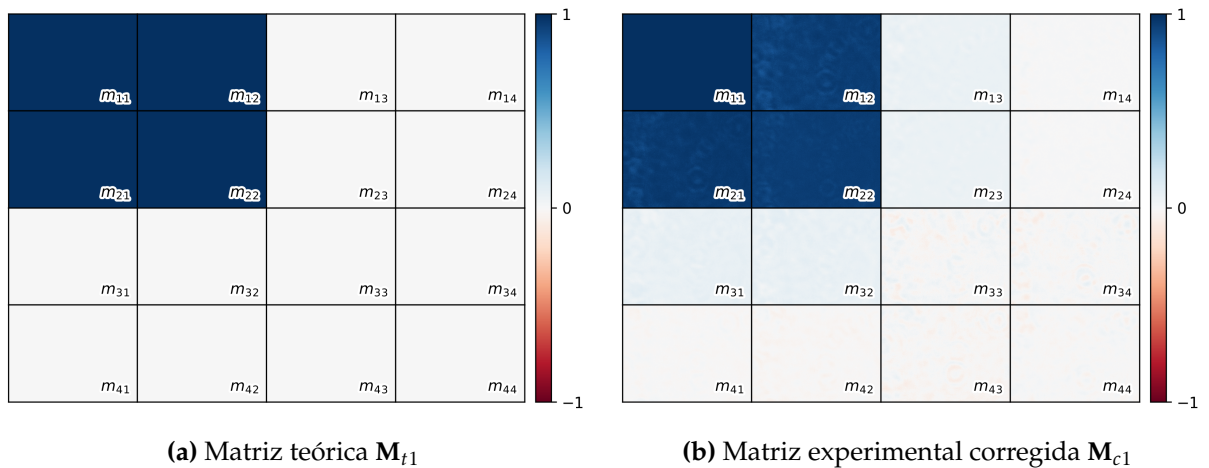


Figura 5.6: Visualización de las matrices de Müller del polarizador lineal horizontal (0°).

CAPÍTULO 5. RESULTADOS

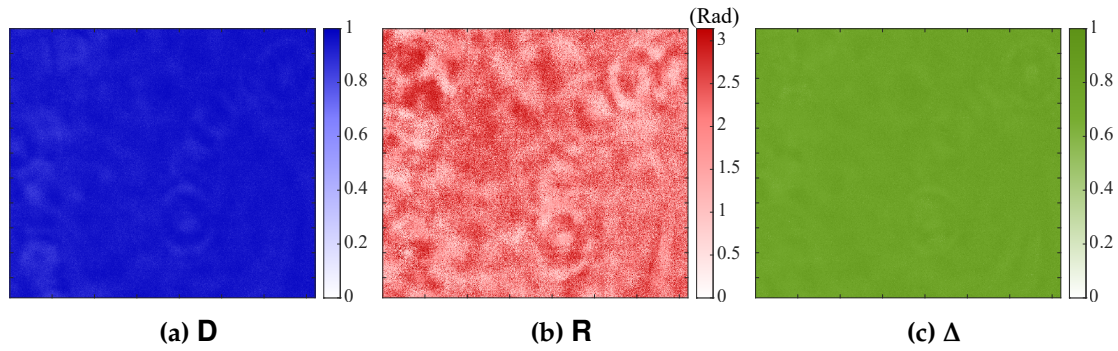


Figura 5.7: Imágenes de las propiedades ópticas de M_{c1} . Diatenuación (D), retardancia (R) y depolarización (Δ).

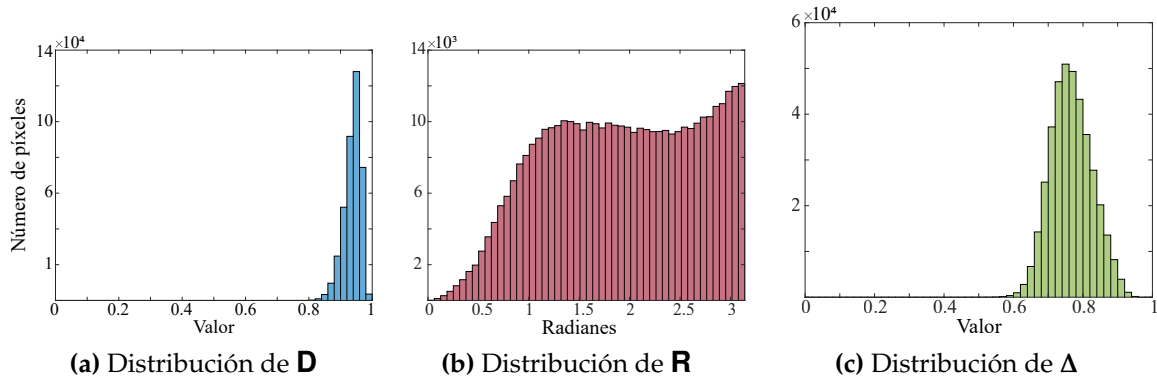


Figura 5.8: Histogramas de las imágenes de las propiedades ópticas de M_{c1} . Diatenuación (D), retardancia (R) y depolarización (Δ).

Polarizador lineal vertical (90°)

De manera similar al polarizador horizontal, el polarizador vertical (90°) presenta valores de diatenuación alrededor de 1 con poca dispersión, cercanos al valor teórico. La depolarización se desvía a un valor totalmente erróneo y por lo tanto la retardancia también presenta valores incorrectos.

En los dos polarizadores solo se consideró válido el resultado de la diatenuación. Los resultados obtenidos de la depolarización y retardancia, tienen error introducido en la etapa de calibración que aumenta con los cálculos subsecuentes.

5.1. PRUEBAS. RESULTADOS TEÓRICOS VS. EXPERIMENTALES

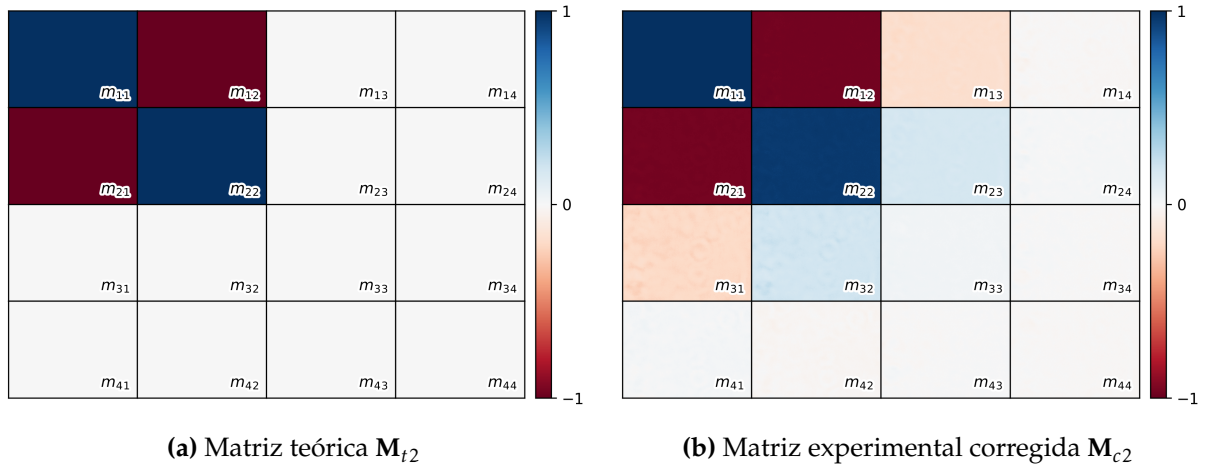


Figura 5.9: Visualización de las matrices de Müller del polarizador lineal vertical (90°).

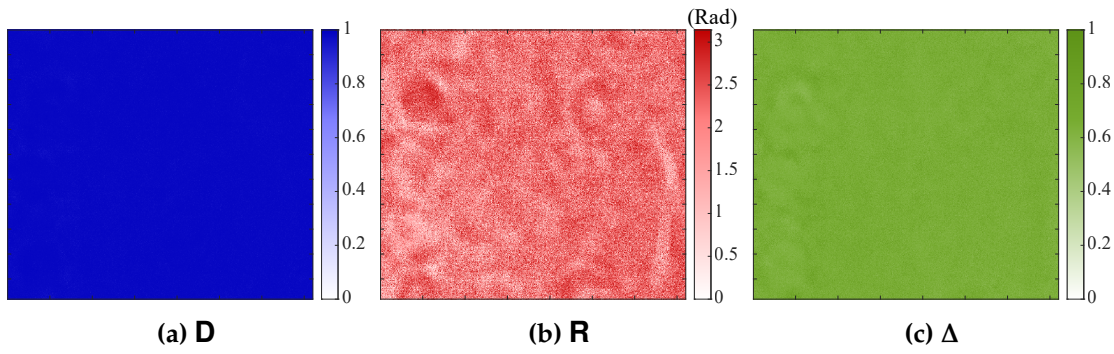


Figura 5.10: Imágenes de las propiedades ópticas de $\mathbf{M}_{c2}$. Diatenuación ($\mathbf{D}$), retardancia ($\mathbf{R}$) y depolarización (Δ).

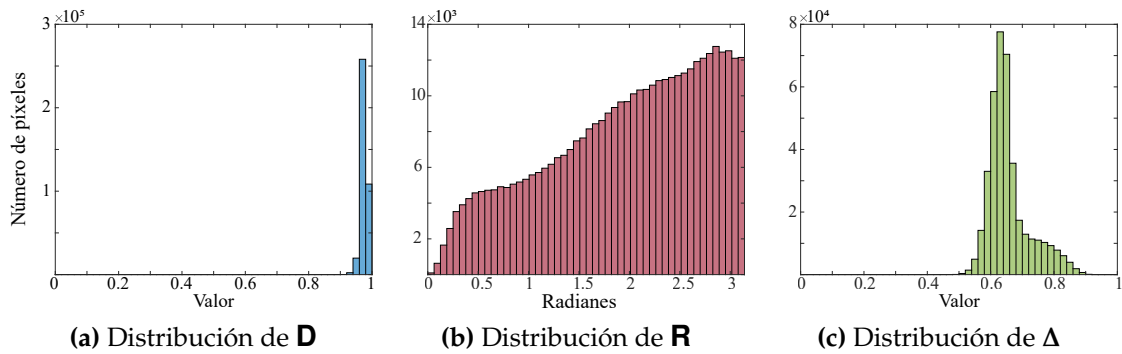


Figura 5.11: Histogramas de las imágenes de las propiedades ópticas de $\mathbf{M}_{c2}$. Diatenuación ($\mathbf{D}$), retardancia ($\mathbf{R}$) y depolarización (Δ).

CAPÍTULO 5. RESULTADOS

Retardador lineal de $\lambda/4$ a 30°

La cuarta muestra de referencia es una placa retardadora de $\lambda/4$ orientada con su eje rápido a 30° . Su matriz de Müller presentó mayor precisión y exactitud en sus propiedades ópticas que las de los polarizadores. Esto lo podemos observar en los tres histogramas de la figura 5.14. El valor medio en las 3 propiedades está alrededor del valor esperado (consultar tabla 5.3) y tienen muy poca dispersión entre pixeles.

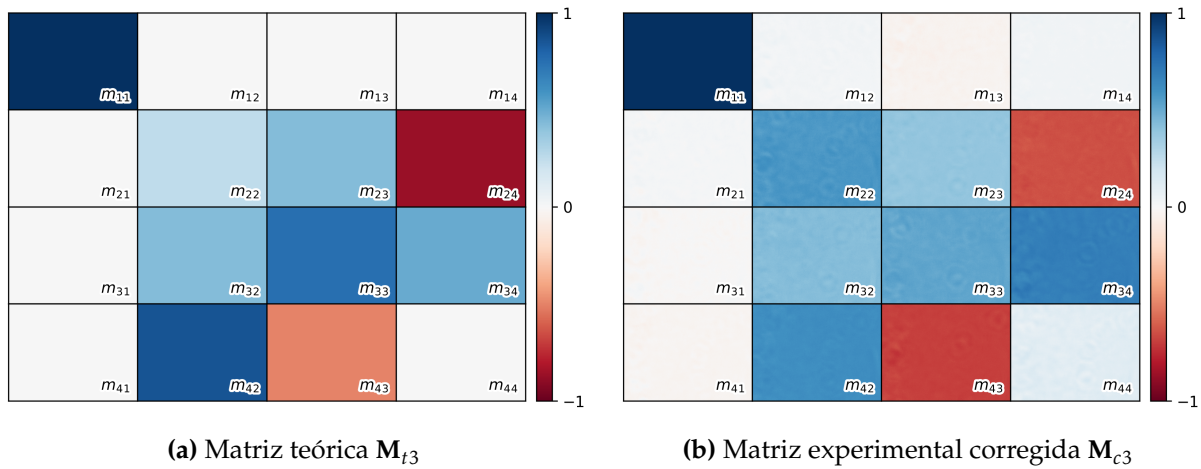


Figura 5.12: Visualización de las matrices de Müller del retardador lineal $\lambda/4$ a 30° .

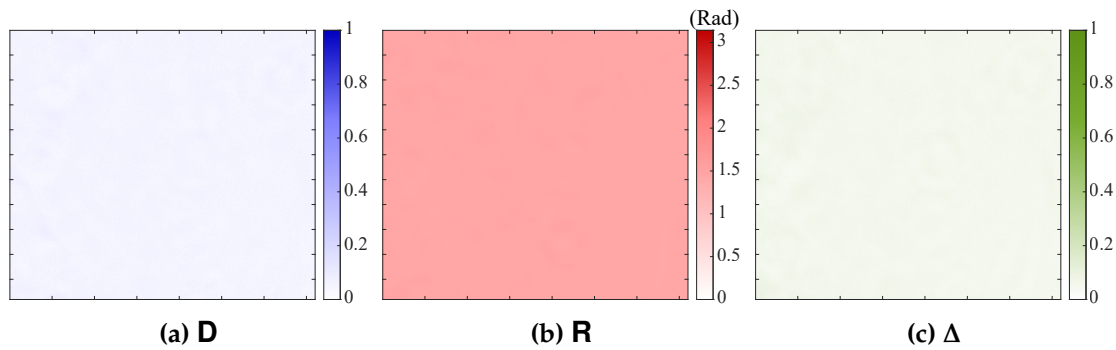


Figura 5.13: Imágenes de las propiedades ópticas de $\mathbf{M}_{c3}$. Diatenuación ($\mathbf{D}$), retardancia ($\mathbf{R}$) y depolarización (Δ).

5.2. EXPERIMENTOS CON MUESTRAS BIOLÓGICAS

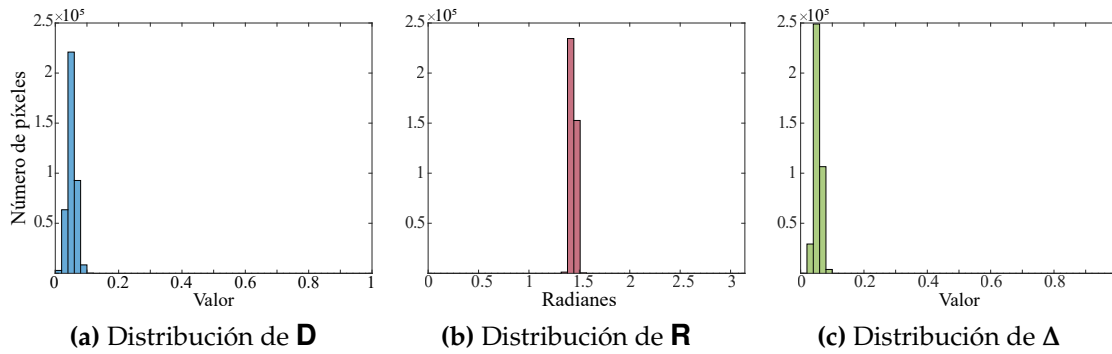


Figura 5.14: Histogramas de las imágenes de las propiedades ópticas de M_{c3} . Diatenuación (**D**), retardancia (**R**) y depolarización (Δ).

5.2. Experimentos con muestras biológicas

En esta sección se describirán a detalle los 3 experimentos que se realizaron, las consideraciones en el protocolo de adquisición y producción de resultados. Se incluyen las representaciones gráficas de los resultados para cada experimento y la discusión de su interpretación.

5.2.1. Experimento 1. Cortes de músculo

Como primer acercamiento con muestras biológicas utilizamos dos cortes de músculo humano sin tinción, un corte contiene músculo sano; el otro tiene músculo necrosado. Ambas muestras son cortes de $4[\mu\text{m}]$ de grosor de tejido de musculo estriado, incluido en parafina. Las patologías son de especial interés en el campo del procesamiento de imágenes ya que ofrecen una detección temprana de la enfermedad con un método no invasivo.

Otros trabajos han reportado resultados favorables al usar esta técnica polarimétrica para identificar patologías tisulares [26–29].

CAPÍTULO 5. RESULTADOS

Para comparar los alcances de la técnica con una microscopía tradicional, se tomaron fotografías de las muestras con un microscopio Nomarski (Figura 5.15).

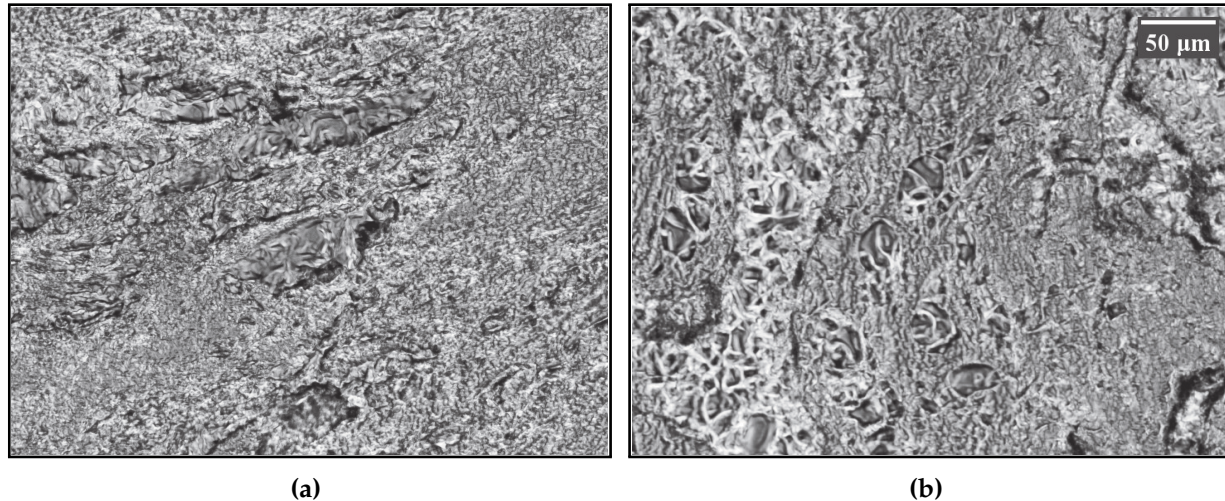


Figura 5.15: Fotografías de los cortes histológicos bajo microscopía de Nomarski. Ambas fotografías tienen la misma escala. a) Musculo en condición sana. b) Músculo con necrosis.

Aunque es posible diferenciar las dos condiciones del tejido con microscopía tradicional, bajo el polarímetro, las diferencias entre ambas condiciones se evidencian más. La diferencia más notable entre las dos muestras de tejido es el tamaño del patrón que se percibe en las imágenes. Las propiedades ópticas revelan una textura densa en la muestra sana, con un patrón de detalles más pequeños en comparación con la imagen de la necrosis, donde se observan huecos negros de gran tamaño. Estos huecos corresponden a los espacios que solían ocupar las células, que al morir, se encogieron y separaron, formando este patrón de mayor tamaño. Además que la diatenuación, retardancia y depolarización nos dejan ver información complementaria acerca de la estructura y organización del tejido (Figura 5.16).

5.2. EXPERIMENTOS CON MUESTRAS BIOLÓGICAS

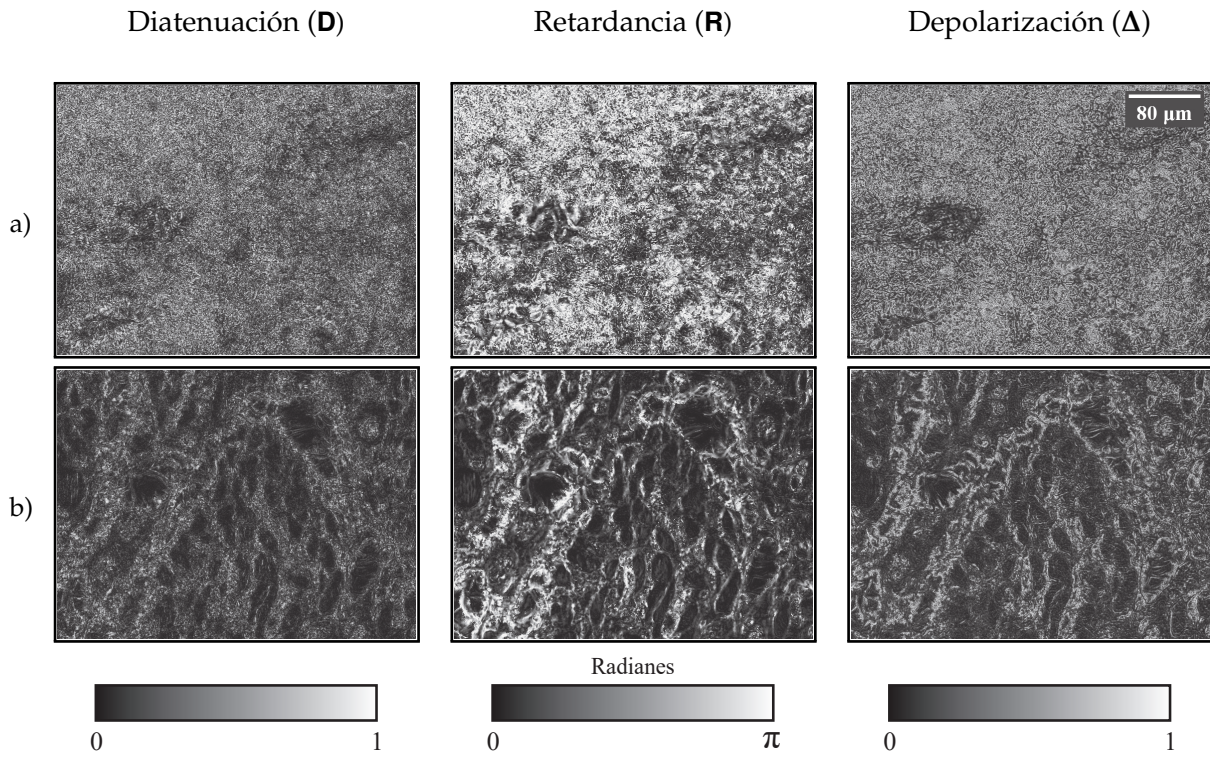


Figura 5.16: Imágenes de las propiedades ópticas de las muestras. La escala espacial es la misma para todas las fotografías. La correspondencia en valor numérico del color de cada pixel se indica en la barra inferior de cada columna. a) Muestra sana. b) Muestra con necrosis.

Recordemos que las imágenes de las propiedades ópticas provienen de un cálculo computacional de punto flotante de doble precisión (64 bits), de este resultado se obtienen las representaciones gráficas redondeando los valores para ajustarse a un formato de 8-bits. Los valores indicados en la barra de escala de las figuras 5.16, 5.17 y 5.18 corresponden a la magnitud física que representan: de 0 a 1 para la diatenuación y la depolarización, que son adimensionales, y de 0 a π para la retardancia, cuyas unidades son radianes. Siguiendo la misma convención de la sección anterior, asignamos: azul para la diatenuación, rojo para la retardancia y verde para la depolarización (Figura 5.17). Al sobreponer las tres propiedades ópticas notamos que cada una destaca distintos componentes tisulares (figura 5.18).

CAPÍTULO 5. RESULTADOS

Estas imágenes compuestas nos brindan una manera rápida y sencilla de analizar toda la información polarimétrica, aunque la precisión está limitada a 256 valores (8 bits por canal). Para el análisis formal de los datos se emplearon matrices numéricas que contienen los datos originales con una precisión ocho veces mayor (64 bits por canal).

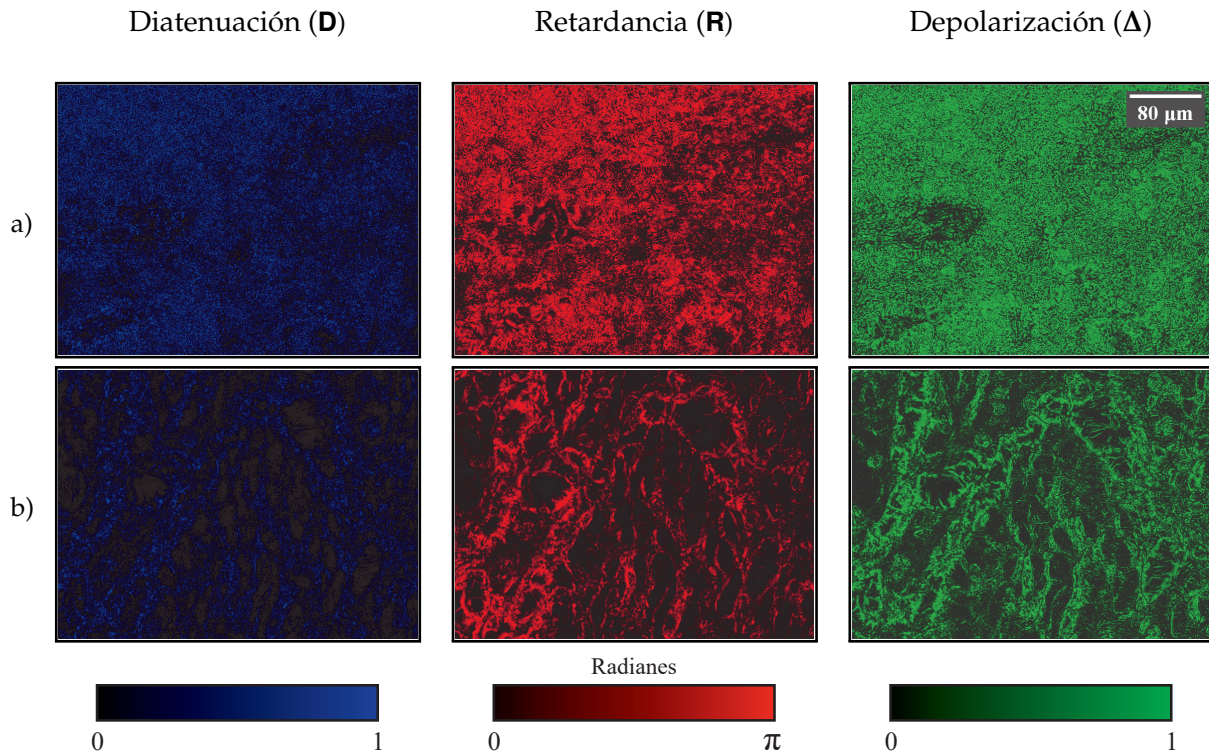
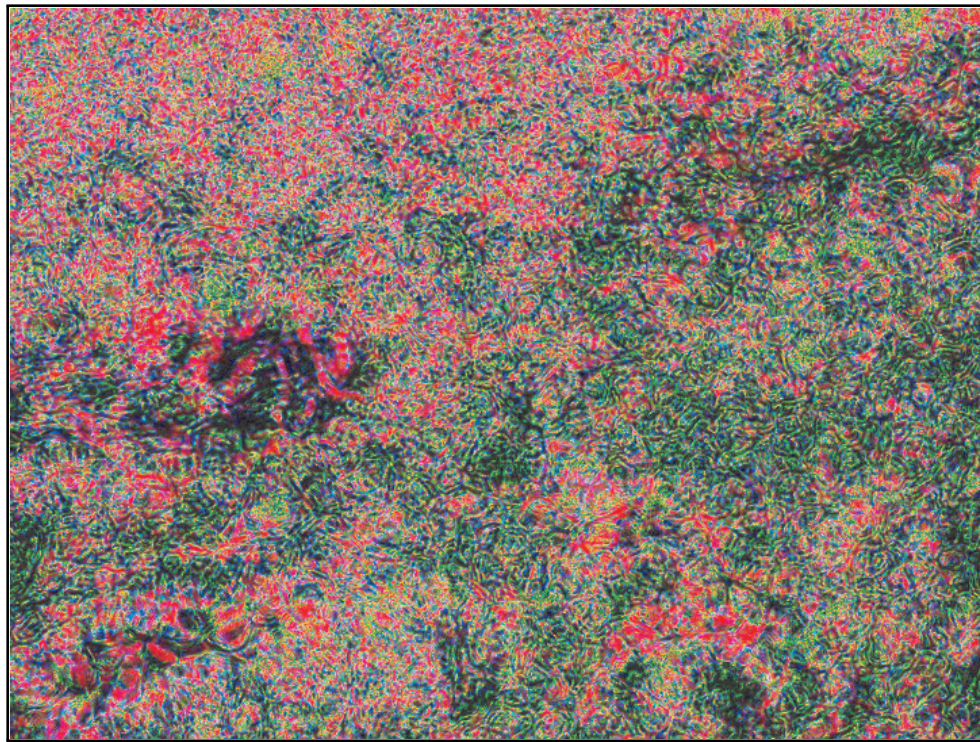
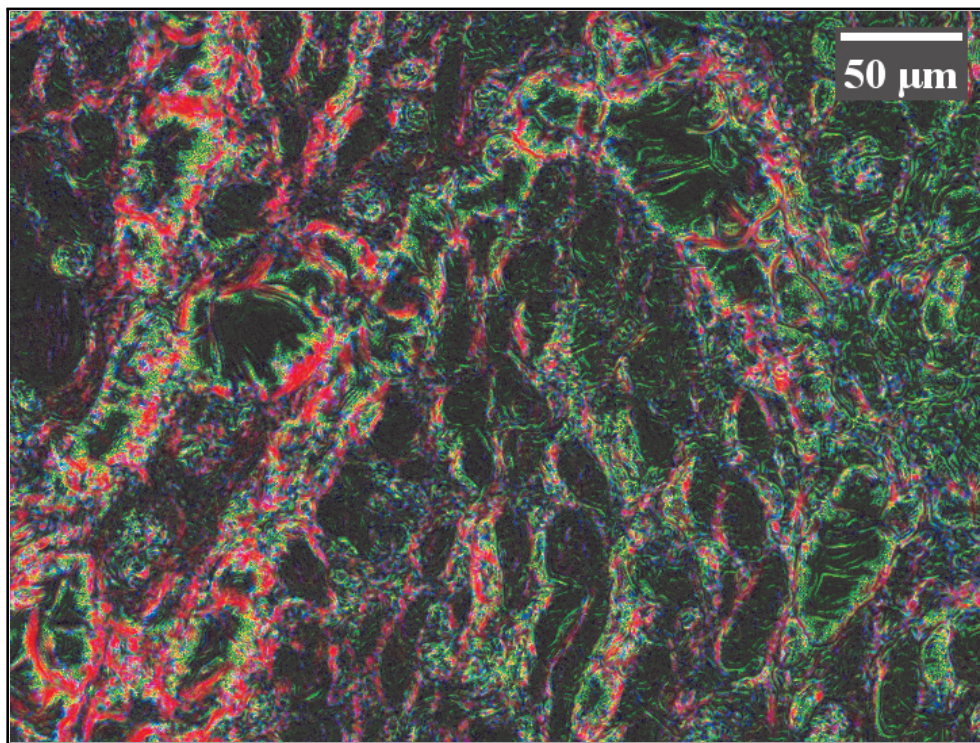


Figura 5.17: Imágenes coloreadas de las propiedades ópticas de las muestras. La escala espacial es la misma para todas las fotografías. La correspondencia en valor numérico del color de cada pixel se indica en la barra inferior de cada columna. a) Muestra sana. b) Muestra con necrosis.

5.2. EXPERIMENTOS CON MUESTRAS BIOLÓGICAS



(a)



(b)

Figura 5.18: Imágenes compuestas en RGB. Ambas imágenes tienen la misma escala. Canal Rojo (Retardancia), canal verde (depolarización) y canal azul (diatenuación). a) Muestra sana. b) Muestra con necrosis.

CAPÍTULO 5. RESULTADOS

5.2.2. Experimento 2. Neuronas fijadas pre y post-estimulación

El segundo experimento consistió en medir las propiedades ópticas de dos grupos de neuronas. Unas fijadas en estado de reposo (grupo control) y las otras después de ser estimuladas eléctricamente para alterar la organización del citoesqueleto.

Empleamos neuronas de la sanguijuela *Hirudo verbana*, la cuál fue anestesiada por inmersión en etanol al 8%. Se diseccionaron los ganglios en grupos de cuatro y fueron conservados en medio de cultivo *Leibovitz L-15* suplementado con 6 [mg/ml] de glucosa, 0.1 [mg/ml] de gentamicina y 2% de suero fetal bovino inactivado por calor. Posteriormente, se abrió con pinzas la capsula de los ganglios para exponer el soma y se incubaron durante 1 hora en solución de 2 [mg/ml] de colagenasa/dispasa. Después del tratamiento enzimático las neuronas fueron extraídas una por una por succión a través de una pipeta de vidrio y se les realizaron repetidos lavados en *L-15* para esterilizarlas. Las células individuales fueron sembradas en platos de cultivo con fondo de vidrio previamente recubierto con concanavalina-A. Los experimentos se realizaron después de 7-10 días en cultivo, con una temperatura ambiente de 18°.

La estimulación se realizó induciendo la despolarización de la membrana celular con solución Ringer con alto potasio (40mM) y posteriormente se fijaron con paraformaldehído (PFA) al 4%. A las neuronas del grupo control se les añadió una solución de Ringer con concentración normal de potasio, que no induce estimulación. Ambas soluciones tienen la misma osmolaridad (concentración total de las sustancias), la composición de estas se muestra en la Tabla 5.4.

Una vez producida la estimulación, lo cual ocurre aproximadamente 5 minutos después de añadir la solución correspondiente, se fijaron ambos grupos de neuronas con PFA para conservar su estado fisiológico y poder registrar sus propiedades ópticas en el polarímetro.

Las imágenes de las propiedades ópticas en blanco y negro del grupo control y el grupo estimulado se muestran en las Figuras 5.19 y 5.20

5.2. EXPERIMENTOS CON MUESTRAS BIOLÓGICAS

Sustancia	Concentración	
	Solución de Ringer (grupo control)	Solución con alto potasio (grupo estimulado)
Cloruro de Sodio (NaCl)	115 [mM]	79 [mM]
Cloruro de Calcio dihidrato (CaCl_2)	1.8 [mM]	1.8 [mM]
Cloruro de Potasio (KCl)	4 [mM]	40 [mM]
TRIS Maleate	10 [mM]	10 [mM]
Glucosa	11 [mM]	11 [mM]

Tabla 5.4: Composición de las soluciones utilizadas en el grupo control y el grupo estimulado.

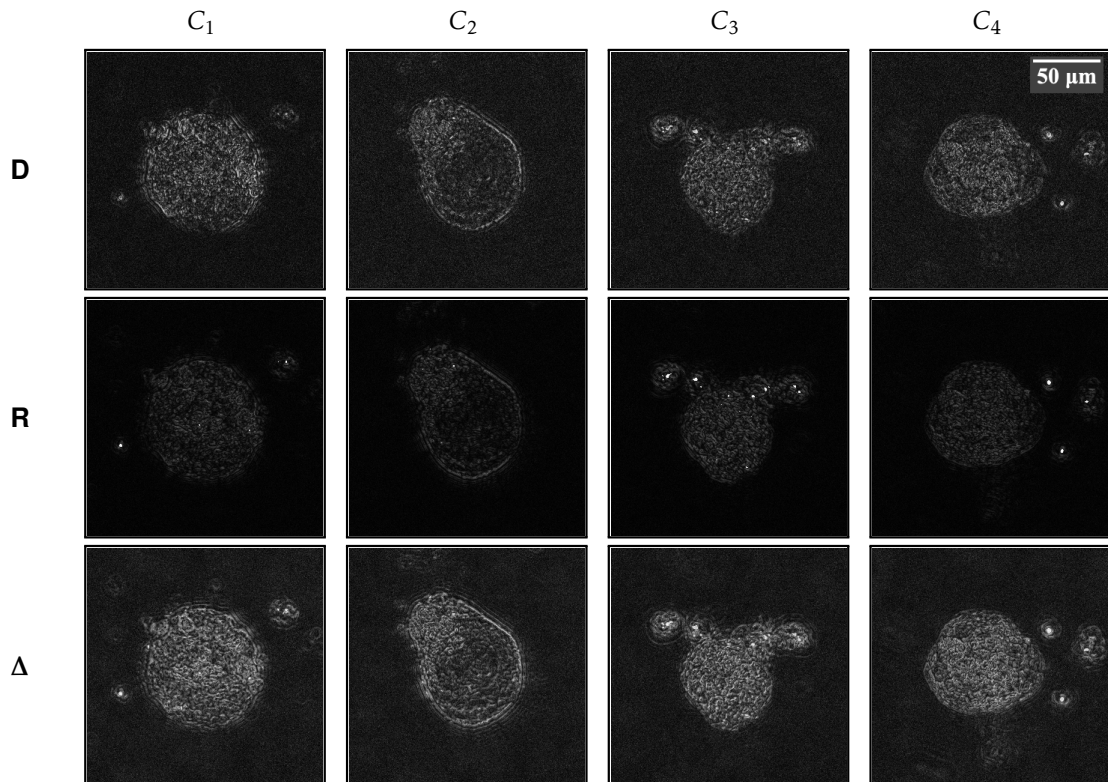


Figura 5.19: Imágenes de las propiedades ópticas de las neuronas del grupo control (C_{1-4}). La escala espacial es la misma para todas las fotografías. Diatenuación (**D**), retardancia (**R**) y depolarización (Δ)

CAPÍTULO 5. RESULTADOS

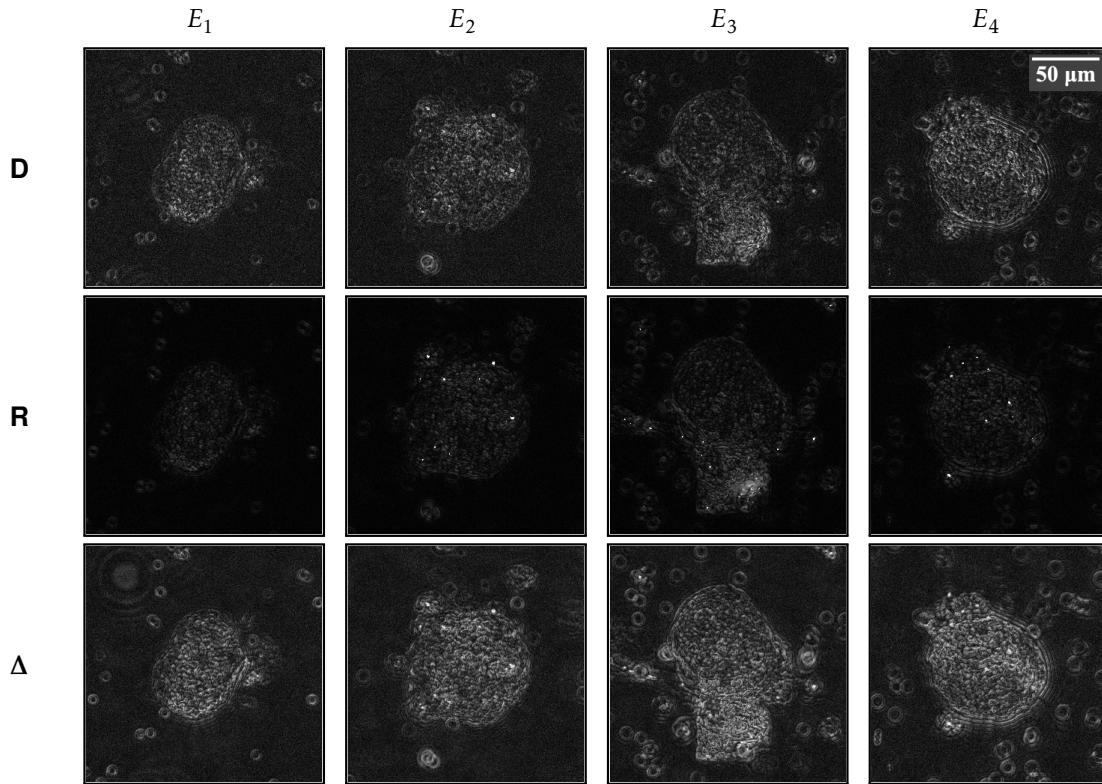


Figura 5.20: Imágenes de las propiedades ópticas de las neuronas del grupo estimulado (E_{1-4}). La escala espacial es la misma para todas las fotografías. Diatenuación (**D**), retardancia (**R**) y depolarización (Δ)

Las imágenes pseudocoloradas para diatenuación, retardancia y depolarización, se muestran en las Figuras 5.21 y 5.22. Para una comparación a primer vistazo entre los dos grupos, se formaron imágenes compuestas con los canales RGB. La Figura 5.23, muestra toda la información óptica que extrae el polarímetro en una sola imagen.

Los resultados de este experimento no parecen señalar una diferencia clara entre el grupo de neuronas estimuladas en comparación con las que permanecieron en reposo (control). Esto puede deberse a varios factores, entre los más probables se encuentra la baja resolución espacial que tienen las fotografías, se usó un objetivo con magnificación de 10x que encuadra a las neuronas en un área de 100x100 píxeles aproximadamente, lo cual no permite resolver adecuadamente las propiedades ópticas de cada región. Otro punto crítico se encuentra en la naturaleza anisotrópica de la muestra biológica, ya que

5.2. EXPERIMENTOS CON MUESTRAS BIOLÓGICAS

al conformarse de diversas estructuras, sus propiedades ópticas no son homogéneas y dependen de la sección longitudinal que cruzó la luz, dando como resultado un conjunto de propiedades ópticas acumuladas de diversos componentes de la célula. Una característica que observamos en ambos grupos, fue que en la periferia de la neurona se presenta mayor diatenuación y menor depolarización en comparación con el interior de esta, la cuál es posiblemente la corteza de actina, una estructura compuesta de filamentos que se acomodan en haces (filamentos paralelos entre sí) o formando tramas (filamentos reticulados).

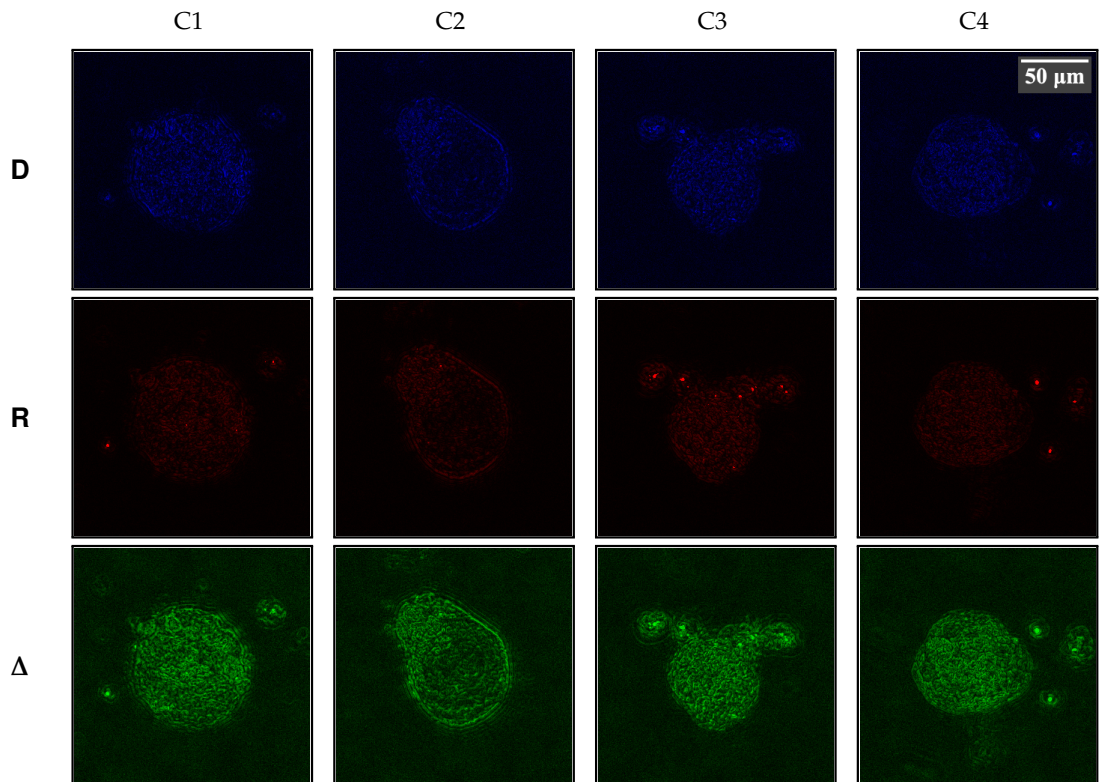


Figura 5.21: Imágenes pseudocoloreadas de las propiedades ópticas de las neuronas control. La escala espacial es la misma para todas las fotografías. Diatenuación (**D**), retardancia (**R**) y depolarización (**Δ**)

CAPÍTULO 5. RESULTADOS

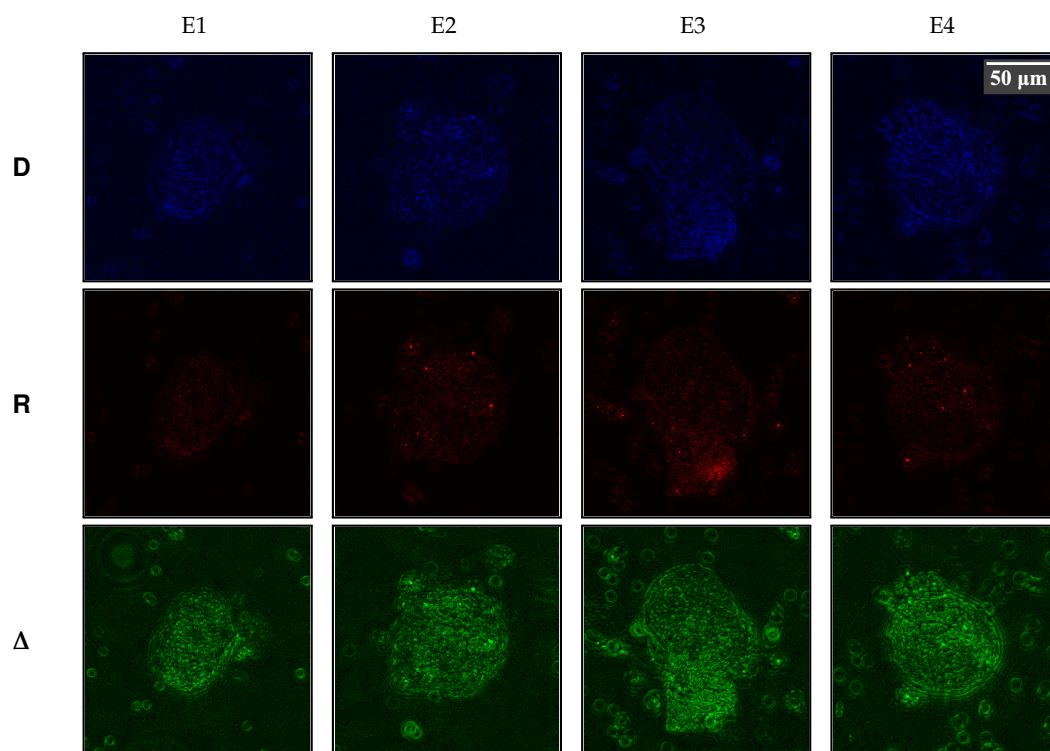


Figura 5.22: Imágenes pseudocoloreadas de las propiedades ópticas de las neuronas estimuladas. La escala espacial es la misma para todas las fotografías. Diatenuación (**D**), retardancia (**R**) y depolarización (Δ)

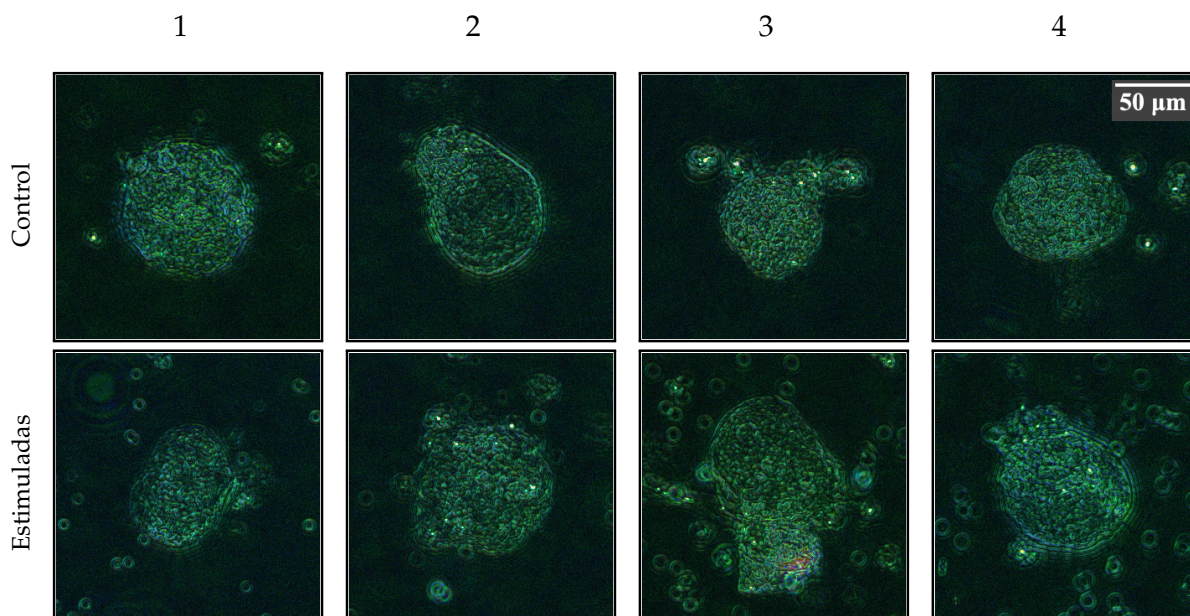


Figura 5.23: Imágenes RGB compuestas por las propiedades ópticas de las neuronas control vs estimuladas. La escala espacial es la misma para todas las fotografías.

5.2.3. Experimento 3. Neuronas vivas

El tercer experimento consistió en registrar las propiedades ópticas de una neurona durante cambios de voltaje transmembranal y la generación de impulsos eléctricos como respuesta a un estímulo de corriente senoidal de 10[Hz] y una amplitud pico a pico de 2[nA] durante 600[ms]. Se usó como muestra neuronas de la sanguijuela *Hirudo* vivas aisladas en cultivo primario. Se midió su potencial de reposo de -50[mV] para confirmar que se encontraban en condiciones viables.

Como el método de medición de las propiedades ópticas (DDR) depende de varias rotaciones mecánicas, el tiempo de adquisición constituye un *cuello de botella* que no podemos eliminar, ya que se deben tomar fotografías en 36 posiciones distintas para obtener un solo registro del polarímetro. Esto se traduce en que la duración total de la medición dura aproximadamente 2.5 minutos.

Las neuronas producen impulsos eléctricos con duración del orden de los milisegundos, por lo cual necesitamos una frecuencia de muestreo lo suficientemente alta para observar estos cambios rápidos. Se propuso adaptar el protocolo de adquisición de la siguiente manera:

- Se inyecta una señal de corriente compuesta por: 0[nA] constantes durante 200[ms] como basal, una senoidal de ± 1 [nA] de amplitud durante 600[ms] que provoca el estímulo eléctrico y al final 0[nA] durante 200[ms] como basal. Se introduce la punta de un electrodo para inyectar la corriente de forma intracelular en la neurona.
- Se registra el voltaje de la célula durante la estimulación a través del mismo electrodo en una configuración de fijación de corriente (current-clamp)
- Se adquieren fotografías durante un segundo a 520 fps en cada una de las 36 posiciones de los retardadores rotadores del polarímetro.

CAPÍTULO 5. RESULTADOS

- Las respuestas de la célula (voltaje y propiedades ópticas) se superponen temporalmente para calcular la diatenuación, retardancia y depolarización a lo largo del tiempo.

En el experimento ideal, las 36 repeticiones de la estimulación deberían resultar en una respuesta de la célula idéntica cada vez, pero como se observa en la figura 5.24 no ocurre así. Esto induce error en el cálculo de las tres propiedades ópticas, que aunque parecen mostrar cambios en el inicio y al final de la señal senoidal de estimulación indicados con franjas rojas verticales, no reportan claramente la rápida dinámica del potencial de acción.

El experimento en la Figura 5.24 se repitió con resultados similares a los mostrados aquí. La figura muestra datos de un experimento representativo y en el que las 36 respuestas de voltaje de la célula fueran lo más parecidas entre sí.

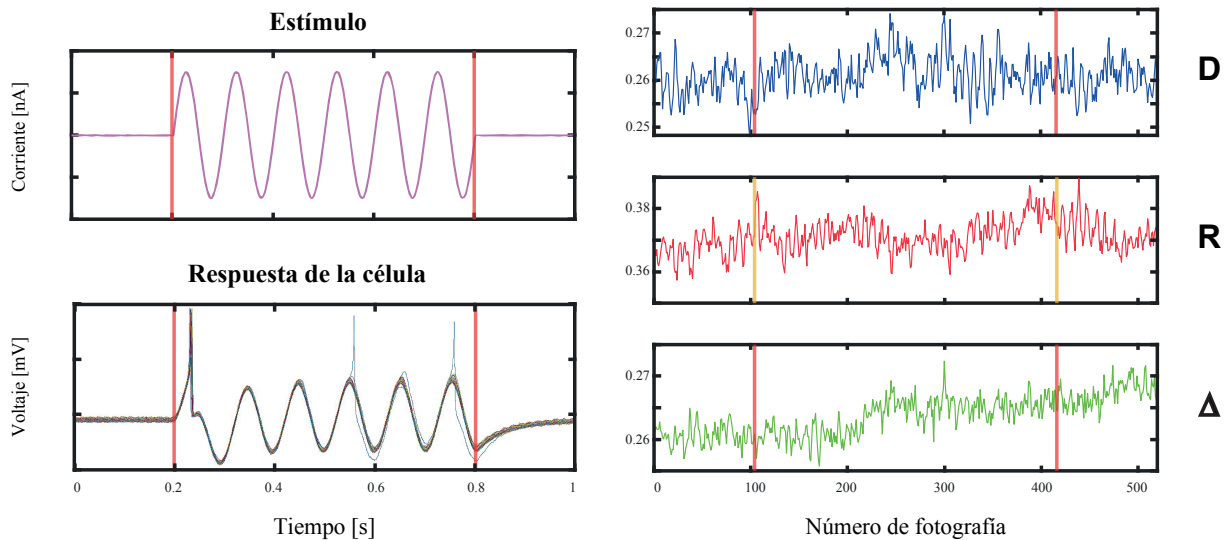


Figura 5.24: El recuadro superior izquierdo muestra la señal de corriente estimulante. El recuadro inferior izquierdo superpone la respuesta acumulada de voltaje en la célula (36 registros). Los recuadros de la derecha muestran las secuencias temporales de 1 [s] de las propiedades ópticas. Se adquirió con una velocidad de 520 fps.

5.2. EXPERIMENTOS CON MUESTRAS BIOLÓGICAS

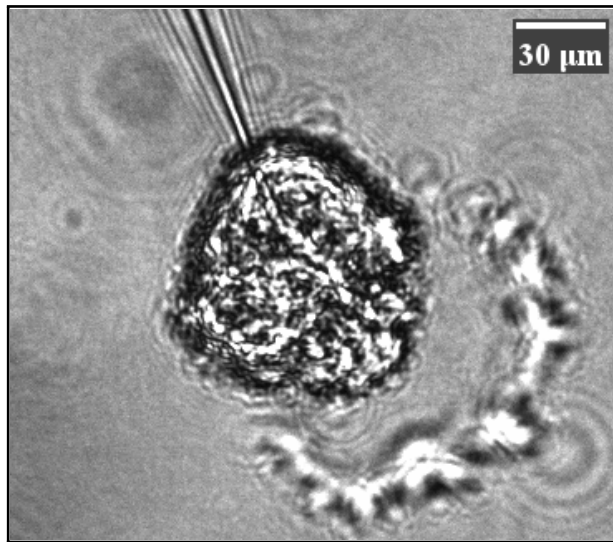


Figura 5.25: Fotografía de la neurona con el electrodo intracelular utilizado para inyectar corriente y medir voltaje.

CAPÍTULO 5. RESULTADOS

Capítulo 6

Conclusiones

El polarímetro de Müller implementado en este trabajo tiene aplicaciones en la neurociencia, medicina y todas aquellas áreas que requieran analizar muestras biológicas con propiedades polarimétricas.

El polarímetro identificó diferencias significativas entre tejido muscular sano y necrótico. Un estudio patológico comparando técnicas histológicas permitirá identificar estructuras y posiblemente moléculas que polarizan la luz con propiedades diagnósticas. Una limitación del equipo actual es su resolución temporal, debido a que el método empleado requiere la adquisición de 36 imágenes consecutivas.

El trabajo en neuronas fijadas nos enfrentó a detectar cambios en la estructura intracelular, que a pesar de haber sido detectados con técnicas inmunológicas no se lograron resolver con polarimetría, ya que, las propiedades ópticas que reflejan los resultados son el producto de multitud de elementos que interactúan con la polarización de la luz, sin posibilidad de analizarlos por separado. Es posible que el estudio usando un objetivo de mayor amplificación y apertura numérica mejore la sensibilidad polarimétrica y resolución óptica del instrumento.

CAPÍTULO 6. CONCLUSIONES

El problema se complejizó aún más al tratar de detectar en tiempo real la dinámica temporal rápida de la actividad eléctrica neuronal, acumulando los errores anteriores y sumando el que se produce debido al movimiento de la muestra viva.

Un área de oportunidad que tiene el polarímetro, en la que puede trabajarse en primera instancia, está en el proceso de caracterización con muestras de referencia (calibración), que tratamos de resolver junto con Isaac Chavez y Dilia Aguirre en [19]. Matemáticamente, las matrices de Müller de los elementos ópticos como placas retardadoras y polarizadores se encuentran en el límite de lo físicamente realizable, lo cual vuelve complejo encontrar una solución computacionalmente, o siquiera tener certeza de que exista una solución viable que compense los errores sistemáticos producidos por faltas de alineación del instrumento o defectos de fabricación en los elementos ópticos. Teniendo en cuenta que este proceso debería refinar los datos y llevarlos a una solución físicamente realizable, no se puede asegurar la fiabilidad de los datos de muestras desconocidas, pues en las pruebas con el mismo conjunto de muestras de referencia de calibración, algunos resultados, como la depolarización, se alejan del valor esperado.

El trabajo a futuro ha de enfocarse en: probar distintos métodos de calibración para escoger el que dé mejores resultados experimentales, la construcción de una configuración en reflexión y la cuantificación de la relación señal ruido (SNR) para determinar su sensibilidad y compararlo con la configuración actual en transmisión, con la intención de que estos cambios de potencial tan pequeños (en el orden de los milivolts) que se producen en las neuronas sean percibidos por el equipo.

El polarímetro tiene un principio de funcionamiento que, adaptándolo según el tipo de muestra, está en posibilidad de: identificar estructuras celulares con propiedades polarimétricas, diferenciar los tejidos sanos y con patologías y detectar actividad eléctrica neuronal.

Apéndice A

Código fuente

SNAP-GUI.py

```
1 import os.path
2 from Clase_exp import Experimento
3 import tkinter as tk
4 from tkinter import simpledialog, messagebox
5 import sys
6 import scipy as sio
7
8 ## Programa principal del polarímetro
9 class InterfazExperimento:
10     def __init__(self, ventana):
11         self.exp = None
12         self.ventana = ventana
13         self.ventana.title("Polarímetro. En operación")
14
15         self.boton_nuevo_exp = tk.Button(ventana,
16             text="Nuevo_Experimento", command=self.newExp)
```

APÉNDICE A. CÓDIGO FUENTE

```
17         self.boton_nuevo_exp.pack(pady=10)
18
19         self.boton_salir = tk.Button(ventana, text="Salir",
20                                     command=self.ventana.destroy)
21         self.boton_salir.pack(pady=10)
22
23         # Variables para los checkboxes
24         self.var_exp = tk.IntVar()
25         self.var_calib = tk.IntVar()
26         self.var_filt = tk.IntVar()
27         self.var_png = tk.IntVar()
28
29
30     def newExp(self):
31         # Ocultar botones anteriores
32         self.boton_nuevo_exp.pack_forget()
33         self.boton_salir.pack_forget()
34
35         # Instancia un nuevo experimento
36         self.exp = Experimento()
37
38         # Mostrar nuevos botones
39         self.boton_snap = tk.Button(self.ventana, text="Snap",
40                                     command=self.ejecutar_snap)
41         self.boton_snap.pack(pady=10)
42
43
44     def ejecutar_snap(self):
```

```

45         # Redirigir la entrada y salida estándar
46         sys.stdin = simpledialog.askstring("Input", "Introduzca
47         un_nombre_para_el_experimento_o_presione_enter_para
48         continuar._'EXIT'_para_salir...")
49         sys.stdout = tk.Text(self.ventana)
50         sys.stdout.pack()
51
52         # Llamar al método que requiere input() y print()
53         self.start()
54
55         # Mostrar checkboxes después de ejecutar 'snap'
56         self.checkbox_exp = tk.Checkbutton(self.ventana,
57         text="Guardar_matriz_de_Müller",
58         variable=self.var_exp)
59         self.checkbox_exp.pack()
60
61         self.checkbox_calib = tk.Checkbutton(self.ventana,
62         text="Guardar_matriz_calibrada",
63         variable=self.var_calib)
64         self.checkbox_calib.pack()
65
66         self.checkbox_filt = tk.Checkbutton(self.ventana,
67         text="Guardar_matriz_calibrada_y_filtrada",
68         variable=self.var_filt)
69         self.checkbox_filt.pack()
70
71         self.checkbox_png = tk.Checkbutton(self.ventana,
72         text="Generar_imagen_PNG", variable=self.var_png)

```

APÉNDICE A. CÓDIGO FUENTE

```
73         self.checkbox_png.pack()
74
75         # Botón para computar los calculos seleccionados
76         self.boton_procesar = tk.Button(self.ventana,
77             text="Procesar", command=self.procesar)
78         self.boton_procesar.pack(pady=10)
79
80     def procesar(self):
81         if (self.var_exp.get()!=1 and self.var_calib.get()!=1
82             and self.var_filt.get()!=1
83             and self.var_png.get()!=1):
84             self.volver_al_menu_inicial()
85         else:
86             M = self.exp.calcMueller()
87
88             if self.var_exp == 1:
89                 sio.savemat(os.path.join('Matrices_Exp',
90                     self.nom+'.mat'), {self.nom + "_exp": M})
91
92             if self.var_calib == 1:
93                 M = self.exp.calibMueller(M)
94                 sio.savemat(os.path.join('Matrices_Cal',
95                     self.nom + '.mat'), {self.nom + "_cal": M})
96
97             if self.var_filt == 1:
98                 M = self.exp.filtMueller(M)
99                 sio.savemat(os.path.join('Matrices_Filt',
100                     self.nom + '.mat'), {self.nom + "_filt": M})
```



```

101
102         if self.var_png == 1:
103             filename = os.path.join('PNGs', self.nom+'.png')
104             self.exp.plotMueller(filename, M)
105
106         self.volver_al_menu_inicial()
107
108
109     def volver_al_menu_inicial(self):
110         # Ocultar elementos de la nueva interfaz
111         self.boton_snap.pack_forget()
112         self.checkbox_exp.pack_forget()
113         self.checkbox_calib.pack_forget()
114         self.checkbox_filt.pack_forget()
115         self.checkbox_png.pack_forget()
116
117         # Mostrar botones originales
118         self.boton_nuevo_exp.pack(pady=10)
119         self.boton_salir.pack(pady=10)
120
121
122     # Crear la ventana principal
123     ventana_principal = tk.Tk()
124
125     # Crear la interfaz
126     interfaz = InterfazExperimento(ventana_principal)
127
128     # Ejecutar el bucle principal de la interfaz gráfica

```

APÉNDICE A. CÓDIGO FUENTE

```
129 ventana_principal.mainloop()
```

Clase_exp.py

```
1 import os
2 import time
3 import numpy as np
4 import cv2
5 import scipy.io as sio
6 import matplotlib.pyplot as plt
7 import matplotlib
8 from PIL import Image
9 from mpl_toolkits.axes_grid1 import ImageGrid
10 from nncam import NncamCamera
11 import serial
12 import sys
13 from math import sqrt
14
15
16
17 class Experimento:
18     def __init__(self):
19         # Atributo: Nombre del experimento
20         self.nom = ''
21
```

```

22     matplotlib.use('Agg') # Backend para renderización de
23                             # imágenes PNG. Soluciona
24                             # compatibilidad con Linux
25
26     # Creación de la estructura de directorios
27     os.makedirs('Matrices_Exp', exist_ok=True)
28     os.makedirs('Matrices_Cal', exist_ok=True)
29     os.makedirs('Matrices_Filt', exist_ok=True)
30     os.makedirs('PNGs', exist_ok=True)
31
32     # Asigna el puerto serial al ArdNano. Controlador CH340
33     # con problema de incompatibilidad con 'brltty' en Linux
34     try:
35         self.arduino = serial.Serial('/dev/ttyCH341USB0',
36                                     9600)
37     except:
38         self.arduino = serial.Serial('/dev/ttyUSB0', 9600)
39
40     # Instancia de la cámara para el experimento
41     self.cam = NncamCamera()
42     self.cam.open()
43
44     # Configuración de la captura de la cámara
45     self.exp = 1.5
46     self.gain = 2.5
47     self.cam.set_auto_exposure(False)
48     self.cam.set_exposure_time(int(self.exp * 1000))
49     self.cam.set_a_gain(int(self.gain * 100))

```

APÉNDICE A. CÓDIGO FUENTE

```
50         self.cam.set_rtime(2)
51
52
53     def calcMueller(self):
54         print("Cargando_Imágenes...")
55         cont = 0
56         imlist = []
57         degrees_light = []
58         for deg_l in range(0, 176, 5):
59             cont += 1
60
61             # Aquí se leen las imágenes
62             ruta = 'Imagenes/' + self.nom + '/'
63             filename = f"{cont}_{deg_l:02d}.png"
64             filename_src = ruta + filename
65
66             img = cv2.imread(filename_src, -1)
67             if img is None:
68                 print("Warning:_Image_file_" + filename_src
69                       + "_cannot_be_opened")
70                 continue
71
72             imlist.append(img)
73             degrees_light.append(deg_l)
74
75             # convierte la lista de imgs en un numpy array
76             # y los ángulos a radianes
77             images = np.array(imlist)
```

```

78     images = np.moveaxis(images, 0, -1)
79     radians_light = np.array(degrees_light) * np.pi / 180.0
80
81     print("Calculando la matriz de Müller...")
82     print(f"images.shape: {images.shape}")
83
84     height, width, N = images.shape
85
86
87     #-----#
88     #----- Dual-rotating retarder polarimeter -----#
89     #-----#
90
91     # Matrices del método: "x" (matriz de senos y cosenos)
92     # y "a"(vector de coeficientes de la serie)
93
94     x = np.array(
95         [np.ones_like(radians_light),
96          np.cos(2 * radians_light),
97          np.cos(4 * radians_light),
98          np.cos(6 * radians_light),
99          np.cos(8 * radians_light),
100         np.cos(10 * radians_light),
101         np.cos(12 * radians_light),
102         np.cos(14 * radians_light),
103         np.cos(16 * radians_light),
104         np.cos(18 * radians_light),
105         np.cos(20 * radians_light),

```

APÉNDICE A. CÓDIGO FUENTE

```

106         np.cos(22 * radians_light),
107         np.cos(24 * radians_light),
108         np.sin(2 * radians_light),
109         np.sin(4 * radians_light),
110         np.sin(6 * radians_light),
111         np.sin(8 * radians_light),
112         np.sin(10 * radians_light),
113         np.sin(12 * radians_light),
114         np.sin(14 * radians_light),
115         np.sin(16 * radians_light),
116         np.sin(18 * radians_light),
117         np.sin(20 * radians_light),
118         np.sin(22 * radians_light),
119         np.sin(24 * radians_light)]).T
120
121     x_op = np.linalg.inv(x.T @ x) @ x.T # (25, N)
122
123     a = np.tensordot(x_op, images, axes=(1, -1))
124         # (25, height, width)
125
126     a = np.moveaxis(a, 0, -1) # (height, width, 25)
127
128     a = a.reshape(height * width, 25)
129
130     # Una vez teniendo el vector de coeficientes, llenamos
131     # la MM siguiendo las igualdades del artículo de Azzam.
132     m = np.zeros([height * width, 16])
133     m[0:,0] = a[0:,0] - a[0:,2] + a[0:,8] -a[0:,10]+a[0:,12]
```

```

134     m[0:, 4] = 2 * a[0:, 2] - 2 * a[0:, 8] - 2 * a[0:, 12]
135     m[0:, 8] = 2 * a[0:, 14] - 2 * a[0:, 20] - 2 * a[0:, 24]
136     m[0:, 12] = a[0:, 13] - 2 * a[0:, 23]
137     m[0:, 1] = -2 * a[0:, 8] + 2 * a[0:, 10] - 2 * a[0:, 12]
138     m[0:, 5] = 4 * a[0:, 8] + 4 * a[0:, 12]
139     m[0:, 9] = -4 * a[0:, 20] + 4 * a[0:, 24]
140     m[0:, 13] = -4 * a[0:, 21]
141     m[0:, 2] = -2 * a[0:, 20] + 2 * a[0:, 22] - 2 * a[0:, 24]
142     m[0:, 6] = 4 * a[0:, 20] + 4 * a[0:, 24]
143     m[0:, 10] = 4 * a[0:, 8] - 4 * a[0:, 12]
144     m[0:, 14] = 4 * a[0:, 9]
145     m[0:, 3] = 2 * a[0:, 15] - a[0:, 17]
146     m[0:, 7] = -4 * a[0:, 15]
147     m[0:, 11] = -4 * a[0:, 3]
148     m[0:, 15] = -2 * a[0:, 4]
149
150     m = m.reshape(height, width, 16)
151     print("Matriz_lista.")
152     print(f"M.shape_:_{m.shape}")
153     return m
154
155 def plotMueller(filename, img_mueller, vabsmax=None,
156                 cmap="RdBu", add_title=True):
157     print("Renderizando_Imagen_PNG...")
158     print(f"Nombre_del_archivo:_{filename}")
159
160     # Lee la dimensión de la matriz, 3x3 o 4x4
161     height, width, channel = img_mueller.shape

```

APÉNDICE A. CÓDIGO FUENTE

```

162         if channel == 9:
163             n = 3
164         elif channel == 16:
165             n = 4
166         else:
167             raise ValueError(
168                 f" 'img_mueller' _shape_should_be_(height , _width ,
169                 16):_({height} , _{width} , _{channel}) ")
170
171         # Agrega los titulos dentro de cada imagen
172         def add_inner_title(ax, title , loc , size=None,**kwargs):
173             from matplotlib.offsetbox import AnchoredText
174             from matplotlib.path_effects import withStroke
175             if size is None:
176                 size =dict(size=plt.rcParams[ 'legend.fontsize ' ])
177             at = AnchoredText(title , loc=loc , prop=size ,
178                               pad=0. , borderpad=0.5 ,
179                               frameon=False , **kwargs)
180             ax.add_artist(at)
181             at.txt._text.set_path_effects([ withStroke
182                 (foreground="w" , linewidth=3)])
183             return at
184
185         fig = plt.figure()
186
187         # Crea la rejilla (grid)
188         grid = ImageGrid(fig , 111,
189                         nrows_ncols=(n, n) ,

```



```

190         axes_pad=0.0,
191         share_all=True,
192         cbar_location="right",
193         cbar_mode="single",
194         cbar_size="3%",
195         cbar_pad=0.10,
196     )
197
198     # Encuentra valores min y max de todas las matrices
199     vabsmax = np.max(np.abs(img_mueller)) if (vabsmax is
200                                         None) else vabsmax
201     vmax = vabsmax
202     vmin = -vabsmax
203
204     # Agrega las imagenes a la rejilla
205     for i, ax in enumerate(grid):
206         ax.set_xticks([])
207         ax.set_yticks([])
208
209         # Agrega el titulo
210         if add_title:
211             maintitle = "$m$_{0}$$_{1}$".format(i // n + 1,
212                                                i % n + 1)
213             # m{ }{ }
214             t = add_inner_title(ax, maintitle,
215                                loc='lower_right')
216
217     # Inserta la imagen en su cuadro respectivo

```

APÉNDICE A. CÓDIGO FUENTE

```

218         im = ax.imshow(img_mueller[:, :, i], vmin=vmin,
219                         vmax=vmax, cmap=cmap)
220
221     # Colorbar
222     cbar = ax.cax.colorbar(im, ticks=[vmin, 0, vmax])
223     cbar.solids.set_edgecolor("face")
224     ax.cax.toggle_label(True)
225
226     # Guarda la imagen
227     plt.savefig(filename, bbox_inches='tight', dpi=300)
228     plt.close()
229
230     return None
231
232 def filtMueller(M_exp):
233     print(" Filtrando la Matriz de Müller... ")
234     height, width, N = M_exp.shape
235
236     M_exp = M_exp.reshape(height * width, 16)
237
238     # Transformacion de M_exp a T_exp
239     T_exp = np.zeros((height * width, 16), dtype='complex_')
240
241     T_exp[0:, 0] = M_exp[0:, 0] + M_exp[0:, 5] +
242                     M_exp[0:, 10] + M_exp[0:, 15]
243     T_exp[0:, 1] = M_exp[0:, 1] + M_exp[0:, 4] -
244                     (1j * (M_exp[0:, 11] - M_exp[0:, 14]))
245     T_exp[0:, 2] = M_exp[0:, 2] + M_exp[0:, 8] +

```

```

246             (1j * (M_exp[0:, 7] - M_exp[0:, 13]))
247 T_exp[0:, 3] = M_exp[0:, 3] + M_exp[0:, 12] -
248             (1j * (M_exp[0:, 6] - M_exp[0:, 9]))
249 T_exp[0:, 4] = M_exp[0:, 1] + M_exp[0:, 4] +
250             (1j * (M_exp[0:, 11] - M_exp[0:, 14]))
251 T_exp[0:, 5] = M_exp[0:, 0] + M_exp[0:, 5] -
252             M_exp[0:, 10] - M_exp[0:, 15]
253 T_exp[0:, 6] = M_exp[0:, 6] + M_exp[0:, 9] +
254             (1j * (M_exp[0:, 3] - M_exp[0:, 12]))
255 T_exp[0:, 7] = M_exp[0:, 7] + M_exp[0:, 13] -
256             (1j * (M_exp[0:, 2] - M_exp[0:, 8]))
257 T_exp[0:, 8] = M_exp[0:, 2] + M_exp[0:, 8] -
258             (1j * (M_exp[0:, 7] - M_exp[0:, 13]))
259 T_exp[0:, 9] = M_exp[0:, 6] + M_exp[0:, 9] -
260             (1j * (M_exp[0:, 3] - M_exp[0:, 12]))
261 T_exp[0:, 10] = M_exp[0:, 0] - M_exp[0:, 5] +
262             M_exp[0:, 10] - M_exp[0:, 15]
263 T_exp[0:, 11] = M_exp[0:, 11] + M_exp[0:, 14] +
264             (1j * (M_exp[0:, 1] - M_exp[0:, 4]))
265 T_exp[0:, 12] = M_exp[0:, 3] + M_exp[0:, 12] +
266             (1j * (M_exp[0:, 6] - M_exp[0:, 9]))
267 T_exp[0:, 13] = M_exp[0:, 7] + M_exp[0:, 13] +
268             (1j * (M_exp[0:, 2] - M_exp[0:, 8]))
269 T_exp[0:, 14] = M_exp[0:, 11] + M_exp[0:, 14] -
270             (1j * (M_exp[0:, 1] - M_exp[0:, 4]))
271 T_exp[0:, 15] = M_exp[0:, 0] - M_exp[0:, 5] -
272             M_exp[0:, 10] + M_exp[0:, 15]
273

```

APÉNDICE A. CÓDIGO FUENTE

```

274     T_exp = (1 / 2) * T_exp
275
276     T_exp = T_exp.reshape(height * width, 4, 4)
277
278     # Vector de eigenvalores y matriz de eigenvectores
279     L4 = np.zeros((height * width, 4, 4))
280     eigVal, U4 = np.linalg.eigh(T_exp, UPLO='U')
281
282     # Ordenamiento de los eigenvalores y sus eigenvectores
283     U4[0:] = U4[0:, :, ::-1]
284
285     #-----#
286     #- FILTRADO DE LOS EIGENVALORES NEGATIVOS. eigV >= 0 -#
287     #-----#
288
289     eigVal[eigVal < 0] = 0
290
291     #-----#
292
293     # Matriz Lambda de eigenvalores
294     L4[0:, 0, 0] = eigVal[0:, 3]
295     L4[0:, 1, 1] = eigVal[0:, 2]
296     L4[0:, 2, 2] = eigVal[0:, 1]
297     L4[0:, 3, 3] = eigVal[0:, 0]
298
299     # Matriz de eigenvectores adjunta (transpuesta conj.)
300     U4_Adj = np.zeros((height * width, 4, 4),
301                        dtype='complex_')

```

```

302     U4_Adj = np.moveaxis(U4, 1, -1)
303     U4_Adj = np.conjugate(U4_Adj)
304
305     # Matriz de coherencia filtrada
306     T_filt = np.zeros([height * width, 4, 4],
307                       dtype='complex_')
308     T_filt[0:] = U4[0:] @ L4[0:] @ U4_Adj[0:]
309
310     T_filt = T_filt.reshape(height * width, 16)
311
312     # Transformacion de T_filt a M_filt
313     M_filt = np.zeros((height * width, 16), dtype='complex_')
314     M_filt[0:, 0] = T_filt[0:, 0] + T_filt[0:, 5] +
315                   T_filt[0:, 10] + T_filt[0:, 15]
316     M_filt[0:, 1] = T_filt[0:, 1] + T_filt[0:, 4] -
317                   (1j * (T_filt[0:, 11] - T_filt[0:, 14]))
318     M_filt[0:, 2] = T_filt[0:, 2] + T_filt[0:, 8] +
319                   (1j * (T_filt[0:, 7] - T_filt[0:, 13]))
320     M_filt[0:, 3] = T_filt[0:, 3] + T_filt[0:, 12] -
321                   (1j * (T_filt[0:, 6] - T_filt[0:, 9]))
322     M_filt[0:, 4] = T_filt[0:, 1] + T_filt[0:, 4] +
323                   (1j * (T_filt[0:, 11] - T_filt[0:, 14]))
324     M_filt[0:, 5] = T_filt[0:, 0] + T_filt[0:, 5] -
325                   T_filt[0:, 10] - T_filt[0:, 15]
326     M_filt[0:, 6] = T_filt[0:, 6] + T_filt[0:, 9] +
327                   (1j * (T_filt[0:, 3] - T_filt[0:, 12]))
328     M_filt[0:, 7] = T_filt[0:, 7] + T_filt[0:, 13] -
329                   (1j * (T_filt[0:, 2] - T_filt[0:, 8]))

```

APÉNDICE A. CÓDIGO FUENTE

```

330     M_filt[0:, 8] = T_filt[0:, 2] + T_filt[0:, 8] -
331         (1j * (T_filt[0:, 7] - T_filt[0:, 13]))
332     M_filt[0:, 9] = T_filt[0:, 6] + T_filt[0:, 9] -
333         (1j * (T_filt[0:, 3] - T_filt[0:, 12]))
334     M_filt[0:, 10] = T_filt[0:, 0] - T_filt[0:, 5] +
335         T_filt[0:, 10] - T_filt[0:, 15]
336     M_filt[0:, 11] = T_filt[0:, 11] + T_filt[0:, 14] +
337         (1j * (T_filt[0:, 1] - T_filt[0:, 4]))
338     M_filt[0:, 12] = T_filt[0:, 3] + T_filt[0:, 12] +
339         (1j * (T_filt[0:, 6] - T_filt[0:, 9]))
340     M_filt[0:, 13] = T_filt[0:, 7] + T_filt[0:, 13] +
341         (1j * (T_filt[0:, 2] - T_filt[0:, 8]))
342     M_filt[0:, 14] = T_filt[0:, 11] + T_filt[0:, 14] -
343         (1j * (T_filt[0:, 1] - T_filt[0:, 4]))
344     M_filt[0:, 15] = T_filt[0:, 0] - T_filt[0:, 5] -
345         T_filt[0:, 10] + T_filt[0:, 15]
346
347     M_filt = (1 / 2) * np.real(M_filt)
348     M_filt = M_filt.reshape(height, width, 16)
349
350     print("Filtrado_listo.")
351     print(f"M_filt.shape_{M_filt}")
352     return M_filt
353
354     def normMueller(self, m):
355         height, width, N = m.shape
356         m = m.reshape(height * width, 16)
357

```

```

358     norma = np.copy(m[0:, 0])
359     m[0:, 0] = m[0:, 0] / norma
360     m[0:, 1] = m[0:, 1] / norma
361     m[0:, 2] = m[0:, 2] / norma
362     m[0:, 3] = m[0:, 3] / norma
363     m[0:, 4] = m[0:, 4] / norma
364     m[0:, 5] = m[0:, 5] / norma
365     m[0:, 6] = m[0:, 6] / norma
366     m[0:, 7] = m[0:, 7] / norma
367     m[0:, 8] = m[0:, 8] / norma
368     m[0:, 9] = m[0:, 9] / norma
369     m[0:, 10] = m[0:, 10] / norma
370     m[0:, 11] = m[0:, 11] / norma
371     m[0:, 12] = m[0:, 12] / norma
372     m[0:, 13] = m[0:, 13] / norma
373     m[0:, 14] = m[0:, 14] / norma
374     m[0:, 15] = m[0:, 15] / norma
375
376     m = m.reshape(height, width, 16)
377     return m
378
379 def MMDecomp(self, M):
380     h, w, N = M.shape
381     M = M.reshape([h * w, N])
382     D = np.zeros([h * w])
383     D_v = np.zeros([h * w, 3])
384
385     # Vector de diatenuación

```

APÉNDICE A. CÓDIGO FUENTE

```

386     D_v[:, 0] = M[:, 1] / M[:, 0]
387     D_v[:, 1] = M[:, 2] / M[:, 0]
388     D_v[:, 2] = M[:, 3] / M[:, 0]
389
390     # Magnitud de la Diatenuación
391     D[:] = np.sqrt((M[:, 1] ** 2) + (M[:, 2] ** 2)
392                   + (M[:, 3] ** 2)) / M[:, 0]
393
394     # Matriz de Diatenuación 4x4
395     MD = np.zeros([h * w, 4, 4])
396
397     # Submatriz de diatenuación 3x3
398     mD = np.zeros([h * w, 3, 3])
399
400     # Matriz identidad 3x3
401     I = np.repeat([np.eye(3)], h * w, axis=0)
402
403     # Cálculo de la submatriz mD
404     mD[:] = (1 - D[:, np.newaxis, np.newaxis] ** 2) **
405             (1 / 2) * I[:] + ((1 - (1 - D[:, np.newaxis,
406             np.newaxis]** 2) ** (1 / 2)) / D[:, np.newaxis,
407             np.newaxis] ** 2) * (np.einsum('ij, ik->ijk',
408             D_v[:, D_v[:]))
409
410     # Primer row y primera col = Vector de diatenuación
411     MD[:, 0, 0] = 1
412     MD[:, 0, 1:] = D_v[:]
413     MD[:, 1:, 0] = D_v[:]

```



```

414     # Los elemetos restantes corresponden a la submatriz mD
415     MD[:, 1:, 1:] = mD[:]
416
417     M = M.reshape([h * w, 4, 4])
418
419     M_prima = np.zeros([h * w, 4, 4])
420     M_prima[:, :] = M[:, :] @ np.linalg.inv(MD[:, :])
421
422     m_prima = np.zeros([h * w, 3, 3])
423     m_prima[:, :] = M_prima[:, 1:, 1:]
424
425     m_dep_2 = np.zeros([h * w, 3, 3])
426     m_dep_2[:, :] = np.matmul(m_prima, np.transpose(m_prima,
427                                                         axes=(0, 2, 1)))
428
429     eig = np.linalg.eigvalsh(m_dep_2)
430     signo = np.sign(np.linalg.det(m_prima))
431
432     m_dep = np.zeros([h * w, 3, 3])
433     m_dep[:, :] = signo[:, np.newaxis, np.newaxis] *
434                     np.linalg.inv(m_dep_2[:, :] +
435                                     (np.sqrt(eig[:, 0] * eig[:, 1]) +
436                                     np.sqrt(eig[:, 1] * eig[:, 2]) +
437                                     np.sqrt(eig[:, 2] * eig[:, 0]))
438                                    [:, np.newaxis, np.newaxis] *
439                                     I[:, :]) @ ((np.sqrt(eig[:, 0]) +
440                                     np.sqrt(eig[:, 1]) + np.sqrt(eig[:, 2]))
441                                    [:, np.newaxis, np.newaxis] *

```

APÉNDICE A. CÓDIGO FUENTE

```

442         m_dep_2[:] + (np.sqrt(eig[:, 0] * eig[:, 1] *
443         eig[:, 2]))[:, np.newaxis, np.newaxis] * I[:]))
444
445     P = np.zeros([h * w, 3])
446     P[:] = (1 / M[:, 0, 0])[:, np.newaxis] * M[:, 1:, 0]
447
448     P_d = np.zeros([h * w, 3])
449     P_d[:] = (P[:] - np.einsum('ijk, ik->ij', M[:, 1:, 1:],
450         D_v[:])) / (1 - D[:, ** 2])[:, np.newaxis]
451
452     M_dep = np.zeros([h * w, 4, 4])
453     M_dep[:, 0, 0] = 1
454     M_dep[:, 1:, 0] = P_d[:]
455     M_dep[:, 1:, 1:] = m_dep[:]
456
457     trM_dep = np.sum(M_dep.diagonal(axis1=1, axis2=2),
458         axis=1)
459     Dep = 1 - ((np.abs(trM_dep - 1)) / 3)
460     Dep = Dep.reshape([h, w])
461
462     M_R = np.linalg.inv(M_dep) @ M_prima
463
464     trM_R = np.sum(M_R.diagonal(axis1=1, axis2=2), axis=1)
465     R = np.arccos((trM_R / 2) - 1)
466     R = R.reshape([h, w])
467
468     D = D.reshape([h, w])
469

```

```

470         return D, Dep, R
471
472     def fecha(self):          # Modifica el atributo self.nom
473         # Fecha Y hora actual
474         anio = time.strftime('%Y')
475         mes = time.strftime('%m')
476         dia = time.strftime('%d')
477         hora = time.strftime('%H')
478         minu = time.strftime('%M')
479
480         # Nombre predeterminado de cada toma (36 fotografías)
481         self.nom = 'F_' + anio + mes + dia + '_H_' + hora + minu
482         return None
483
484     def start(self):
485         mod = input("Introduzca un nombre para el experimento
486         o presione enter para continuar. 'EXIT'
487         para salir ... ")
488
489         if (mod == ''):
490             self.fecha()
491         elif (mod == 'EXIT'):
492             self.close()
493         else:
494             self.nom = mod
495
496         try:
497             os.makedirs('Imagenes/' + self.nom)

```

APÉNDICE A. CÓDIGO FUENTE

```
498         print(f"Exp._nuevo_creado_con_exito_{self.nom}")
499     except OSError as e:
500         r = input("Atención. El experimento ya existe ,
501                 ¿desea sobrescribir los datos? S/N")
502         if (r == 's' or r=='S'):
503             print(f"Experimento_{self.nom}_
504             _____Sobrescribiendo_datos.")
505         else:
506             print("Operación cancelada.")
507
508     return None
509
510     def close(self):
511         self.arduino.close()
512         self.cam.close()
513
514     def preview(self):
515         self.cam.set_auto_exposure(True)
516
517         while (True):
518             frame = np.array(self.cam.get_pil_image())
519             frame = cv2.rotate(frame, cv2.ROTATE_90_CLOCKWISE)
520             cv2.imshow('Frame', frame)
521
522             if cv2.waitKey(1) & 0xFF == ord('q'):
523                 cv2.destroyAllWindows()
524                 break
525
```

```

526         self.cam.set_auto_exposure(False)
527         self.cam.set_exposure_time(int(self.exp * 1000))
528         self.cam.set_a_gain(int(self.gain * 100))
529         self.cam.set_rtime(2)
530
531     def pil_img(self, data):
532         raw = data.view(np.uint8).reshape(data.shape + (-1,))
533         image = Image.fromarray(raw[:, :, 0], mode='L')
534         return image
535
536     def snapMueller(self):
537         os.makedirs(os.path.join('Imagenes', self.nom),
538                     exist_ok=True)
539
540         # Contador fotos
541         i = 0
542         print("Listo ,_inicie_la_rutina_de_'Thorlabs_Kinesis'")
543         while (i < 36):
544             cad = self.arduino.read(1)
545
546             if (cad == b'1'):
547                 print(f"Captura!_{i}")
548                 i = i + 1
549                 img = self.cam.get_image_data()
550                 img = self.pil_img(np.array(img))
551                 img.save('Imagenes/' +
552                         self.nom + f"/{i:02d}.png")
553                 img.save(os.path.join('Imagenes', self.nom,

```

APÉNDICE A. CÓDIGO FUENTE

```
554                                     f"{i:02d}.png"))
555             else:
556                 pass
557     return None
```

nncam.py

```
1  #-----#
2  # Este código ha sido adaptado del ejemplo incluido en el SDK
3  # proveído por el fabricante de la cámara AmScope, así como
4  # el código core_nncam.py y las bibliotecas necesarias para
5  # Windows y Linux pueden descargarse de:
6  # https://eu.amscope.com/pages/software-downloads
7  #-----#
8  import os
9  import ctypes
10 from PIL import Image
11 from io import StringIO
12 from numpy import zeros, uint8, uint32
13 from core_nncam import \
14 lib, Nncam_EVENT_IMAGE, Nncam_EVENT_STILLIMAGE, success, HNncam
15
```

```

16
17 class NncamCamera(object):
18     _data = None
19     _frame_fn = None
20     _temptint_cb = None
21     _save_path = None
22
23     resolution = None
24     size = None
25
26     def __init__(self, resolution=None, bits=32, size=None):
27         if resolution is None and size is None:
28             resolution = 2
29
30         if bits not in (32,):
31             raise ValueError('8_o_32')
32
33         if size is None:
34             self.resolution = resolution
35         else:
36             self.size = size
37
38         self.cam = self.get_camera()
39         self.bits = bits
40
41     def save(self, p, extension='JPEG', *args, **kw):
42         image = self.get_pil_image()
43         image.save(p, extension, *args, **kw)

```

APÉNDICE A. CÓDIGO FUENTE

```
44
45     def save_jpeg(self , p, quality=100):
46         im = self.get_pil_image()
47         im.save(p, 'JPEG', quality=quality)
48
49     def save_tiff(self , p):
50         im = self.get_pil_image()
51         im.save(p, 'TIFF')
52
53     def get_jpeg_data(self , data=None, quality=75):
54         im = self.get_pil_image(data)
55         s = StringIO()
56         im.save(s, 'JPEG', quality=quality)
57         s.seek(0, os.SEEK_END)
58         return s.getvalue()
59
60     def get_pil_image(self , data=None):
61         # im = self._data
62         if data is None:
63             data = self._data
64
65         raw = data.view(uint8).reshape(data.shape + (-1,))
66         bgr = raw[... , :3]
67         image = Image.fromarray(bgr, 'RGB')
68         b, g, r = image.split()
69         return Image.merge('RGB', (r, g, b))
70
71
```



```

72     def get_image_data(self, *args, **kw):
73         d = self._data.copy()
74         return d
75
76     def close(self):
77         if self.cam:
78             lib.Nncam_Close(self.cam)
79
80     def open(self):
81         if self.resolution:
82             self.set_esize(self.resolution)
83         else:
84             self.set_size(*self.size)
85
86         args = self.get_size()
87         if not args:
88             return
89
90         h, w = args[1].value, args[0].value
91
92         shape = (h, w)
93         if self.bits == 8:
94             dtype = uint8
95         else:
96             dtype = uint32
97
98         self._data = zeros(shape, dtype=dtype)
99

```

APÉNDICE A. CÓDIGO FUENTE

```
100         bits = ctypes.c_int(self.bits)
101
102     def get_frame(nEvent, ctx):
103         if nEvent == Nncam_EVENT_IMAGE:
104             w, h = ctypes.c_uint(), ctypes.c_uint()
105
106             lib.Nncam_PullImage(self.cam,
107                                ctypes.c_void_p(
108                                    self._data.ctypes.data),
109                                bits,
110                                ctypes.byref(w),
111                                ctypes.byref(h))
112
113         elif nEvent == Nncam_EVENT_STILLIMAGE:
114             w, h = self.get_size()
115             h, w = h.value, w.value
116
117             still = zeros((h, w), dtype=uint32)
118             lib.Nncam_PullStillImage(self.cam,
119                                     ctypes.c_void_p(
120                                         still.ctypes.data),
121                                     bits, None, None)
122             self._do_save(still)
123
124         callback = ctypes.CFUNCTYPE(None, ctypes.c_uint,
125                                     ctypes.c_void_p)
126         self._frame_fn = callback(get_frame)
127
```

```

128         result = lib.Nncam_StartPullModeWithCallback(self.cam,
129                                                         self._frame_fn)
130
131         return success(result)
132
133     def _do_save(self, im):
134         image = self.get_pil_image(im)
135         image.save(self._save_path)
136
137     def _lib_func(self, func, *args, **kw):
138         ff = getattr(lib, 'Nncam_{}'.format(func))
139         result = ff(self.cam, *args, **kw)
140         return success(result)
141
142     def _lib_get_func(self, func):
143         v = ctypes.c_int()
144         if self._lib_func('get_{}'.format(func),
145                             ctypes.byref(v)):
146             return v.value
147
148     # setters
149     def set_gamma(self, v):
150         self._lib_func('put_Gamma', ctypes.c_int(v))
151
152     def set_contrast(self, v):
153         self._lib_func('put_Contrast', ctypes.c_int(v))
154
155     def set_brightness(self, v):

```

APÉNDICE A. CÓDIGO FUENTE

```
156         self._lib_func('put_Brightness', ctypes.c_int(v))
157
158     def set_saturation(self, v):
159         self._lib_func('put_Saturation', ctypes.c_int(v))
160
161     def set_hue(self, v):
162         self._lib_func('put_Hue', ctypes.c_int(v))
163
164     def set_exposure_time(self, v):
165         self._lib_func('put_ExpoTime', ctypes.c_ulong(v))
166
167     def set_a_gain(self, v):
168         self._lib_func('put_ExpoAGain', ctypes.c_ushort(v))
169
170
171     # getters
172     def get_gamma(self):
173         return self._lib_get_func('Gamma')
174
175     def get_contrast(self):
176         return self._lib_get_func('Contrast')
177
178     def get_brightness(self):
179         return self._lib_get_func('Brightness')
180
181     def get_saturation(self):
182         return self._lib_get_func('Saturation')
183
```

```

184     def get_hue(self):
185         return self._lib_get_func('Hue')
186
187     def get_exposure_time(self):
188         return self._lib_get_func('ExpoTime')
189
190     def get_a_gain(self):
191         return self._lib_get_func('ExpoAGain')
192
193
194     def do_awb(self, callback=None):
195
196         def temptint_cb(temp, tint):
197             if callback:
198                 callback((temp, tint))
199
200         callback = ctypes.CFUNCTYPE(None, ctypes.c_uint,
201                                     ctypes.c_void_p)
202         self._temptint_cb = callback(temptint_cb)
203
204         return self._lib_func('AwbOnePush', self._temptint_cb)
205
206     def set_temperature_tint(self, temp, tint):
207         lib.Nncam_put_TempTint(self.cam, temp, tint)
208
209     def get_temperature_tint(self):
210         temp = ctypes.c_int()
211         tint = ctypes.c_int()

```

APÉNDICE A. CÓDIGO FUENTE

```
212         if self._lib_func('get_TempTint', ctypes.byref(temp),
213                             ctypes.byref(tint)):
214             return temp.value, tint.value
215
216     def get_auto_exposure(self):
217         expo_enabled = ctypes.c_bool()
218         result = lib.Nncam_get_AutoExpoEnable(self.cam,
219                                                ctypes.byref(expo_enabled))
220         if success(result):
221             return expo_enabled.value
222
223     def set_auto_exposure(self, v):
224         expo_enabled = ctypes.c_bool()
225         expo_enabled.value = v
226         lib.Nncam_put_AutoExpoEnable(self.cam, v)
227
228     #Antiparpadeo
229     def set_hz(self, v):
230         lib.Nncam_put_HZ(self.cam, v)
231
232     def get_hz(self):
233         hz = ctypes.c_int()
234         result = lib.Nncam_get_HZ(self.cam, ctypes.byref(hz))
235         if success(result):
236             return hz.value
237
238     # REAL TIME
239     def set_rtime(self, v):
```

```

240         lib.Nncam_put_RealTime(self.cam, v)
241
242     def get_rtime(self):
243         v = ctypes.c_int()
244         result = lib.Nncam_get_RealTime(self.cam,
245                                         ctypes.byref(v))
246         if success(result):
247             return v.value
248
249     # ROTATE
250     def set_Rot(self, value):
251         lib.Nncam_put_Option(self.cam, 0x18, value)
252
253     def get_Rot(self):
254         v = ctypes.c_int()
255         result = lib.Nncam_get_Option(self.cam, 0x18,
256                                         ctypes.byref(v))
257         if success(result):
258             return v.value
259         else:
260             return str(result)
261
262
263     # Frame Speed
264     def set_speed(self, v):
265         lib.Nncam_put_Speed(self.cam, v)
266
267     def get_speed(self):

```

APÉNDICE A. CÓDIGO FUENTE

```
268         v = ctypes.c_ushort()
269         result = lib.Nncam_get_Speed(self.cam, ctypes.byref(v))
270         if success(result):
271             return v.value
272
273     def get_maxSpeed(self):
274         v = ctypes.c_ushort()
275         result = lib.Nncam_get_MaxSpeed(self.cam, ctypes.byref(v))
276         if success(result):
277             return v.value
278
279     def get_camera(self, cid=None):
280         func = lib.Nncam_Open
281         func.restype = ctypes.POINTER(HNncam)
282         cam = func(cid)
283         return cam
284
285     def get_serial(self):
286         sn = ctypes.create_string_buffer(32)
287         result = lib.Nncam_get_SerialNumber(self.cam, sn)
288         if success(result):
289             sn = sn.value
290             return sn
291
292     def get_firmware_version(self):
293         fw = ctypes.create_string_buffer(16)
294         result = lib.Nncam_get_FwVersion(self.cam, fw)
295         if success(result):
```



```

296         return fw.value
297
298     def get_hardware_version(self):
299         hw = ctypes.create_string_buffer(16)
300         result = lib.Nncam_get_HwVersion(self.cam, hw)
301         if success(result):
302             return hw.value
303
304     def get_size(self):
305         w, h = ctypes.c_long(), ctypes.c_long()
306
307         result = lib.Nncam_get_Size(self.cam, ctypes.byref(w),
308                                     ctypes.byref(h))
309         if success(result):
310             return w, h
311
312     def get_esize(self):
313         res = ctypes.c_long()
314         result = lib.Nncam_get_eSize(self.cam, ctypes.byref(res))
315         if success(result):
316             return res
317
318     def set_esize(self, nres):
319         lib.Nncam_put_eSize(self.cam, ctypes.c_ulong(nres))
320
321     def set_size(self, w, h):
322         self._lib_func('put_Size', ctypes.c_long(w),
323                        ctypes.c_long(h))

```

APÉNDICE A. CÓDIGO FUENTE

Referencias

- [1] A. L. Hodgkin and A. F. Huxley, "Action potentials recorded from inside a nerve fibre," *Nature*, vol. 144, pp. 710–711, 1939.
- [2] K. S. Cole and H. J. Curtis, "Electric impedance of the squid giant axon during activity," *The Journal of General Physiology*, vol. 22, pp. 649 – 670, 1939.
- [3] A. L. Hodgkin, A. F. Huxley, and B. Katz, "Measurement of current-voltage relations in the membrane of the giant axon of loligo," *The Journal of Physiology*, vol. 116, 1952.
- [4] D. K. Hill and R. D. Keynes, "Opacity changes in stimulated nerve," *The Journal of Physiology*, vol. 108, 1949.
- [5] L. B. Cohen, R. D. Keynes, and B. Hille, "Light scattering and birefringence changes during nerve activity," *Nature*, vol. 218, pp. 438–441, 1968.
- [6] L. B. Cohen, B. Hille, and R. D. Keynes, "Changes in axon birefringence during the action potential," *The Journal of Physiology*, vol. 211, 1970.
- [7] R. Stepnoski, A. Laporta, F. Raccaia-Behling, G. Blonder, R. E. Slusher, and D. Kleinfeld, "Noninvasive detection of changes in membrane potential in cultured neurons by light scattering,," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 88, pp. 9382 – 9386, 1991.

REFERENCIAS

- [8] K. Carter, J. S. George, and D. M. Rector, "Simultaneous birefringence and scattered light measurements reveal anatomical features in isolated crustacean nerve," *Journal of Neuroscience Methods*, vol. 135, pp. 9–16, 2004.
- [9] A. H. Badreddine, K. J. Schoener, and I. J. Bigio, "Elucidating the temporal dynamics of optical birefringence changes in crustacean nerves.," *Biomedical optics express*, vol. 6 10, pp. 4165–78, 2015.
- [10] A. H. Badreddine, T. Jordan, and I. J. Bigio, "Real-time imaging of action potentials in nerves using changes in birefringence," *Biomedical Optics Express*, vol. 7, pp. 1966 – 1973, 2016.
- [11] R. M. A. Azzam, "Photopolarimetric measurement of the mueller matrix by fourier analysis of a single detected signal," *Opt. Lett.*, vol. 2, pp. 148–150, Jun 1978.
- [12] D. Goldstein, *Polarized Light*. New York: Marcel Dekker, second ed., 2003.
- [13] M. Bass, *Handbook of Optics, Vol. II*. sponsored by the Optical Society of America, second ed., 1995.
- [14] I. Montes, "Diseño y construcción de un polarímetro de stokes para obtención de imágenes," tesis de maestría, Universidad Nacional Autónoma de México, 2018.
- [15] D. Arago and A. Fresnel, "Mémoire sur l'action que les rayons de lumière polarisés exercent les uns sur les autres," *Annales de Chimie et de Physique*, vol. 17, pp. 101–112, 1821.
- [16] G. G. Stokes, *On the Composition and Resolution of Streams of Polarized Light from different Sources*, p. 233–258. Cambridge Library Collection - Mathematics, Cambridge University Press, 2009.
- [17] G. G. Stokes, *On the Change of Refrangibility of Light*, p. 267–414. Cambridge Library Collection - Mathematics, Cambridge University Press, 2009.

- [18] D. H. Goldstein, "Mueller matrix dual-rotating retarder polarimeter," *Appl. Opt.*, vol. 31, pp. 6676–6683, Nov 1992.
- [19] A. D. Chavez Ramírez Isaac, "Caracterización de un polarímetro de mueller para la detección de actividad eléctrica neuronal," tesis de licenciatura, Universidad Nacional Autónoma de México, en preparación.
- [20] S. R. Cloude, "Conditions For The Physical Realisability Of Matrix Operators In Polarimetry," in *Polarization Considerations for Optical Systems II* (R. A. Chipman, ed.), vol. 1166 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 177–185, Jan. 1990.
- [21] J. J. Gil, "Characteristic properties of mueller matrices," *J. Opt. Soc. Am. A*, vol. 17, pp. 328–334, Feb 2000.
- [22] J. J. Gil, "On optimal filtering of measured mueller matrices," *Appl. Opt.*, vol. 55, pp. 5449–5455, Jul 2016.
- [23] S. Cloude, *Polarisation: Applications in Remote Sensing*. OUP Oxford, 2009.
- [24] S.-Y. Lu and R. A. Chipman, "Interpretation of mueller matrices based on polar decomposition," *J. Opt. Soc. Am. A*, vol. 13, pp. 1106–1113, May 1996.
- [25] E. Compain, S. Poirier, and B. Drevillon, "General and self-consistent method for the calibration of polarization modulators, polarimeters, and mueller-matrix ellipsometers," *Appl. Opt.*, vol. 38, pp. 3490–3502, Jun 1999.
- [26] S. L. Jacques, J. C. Ramella-Roman, and K. L. M.D., "Imaging skin pathology with polarized light," *Journal of Biomedical Optics*, vol. 7, no. 3, pp. 329 – 340, 2002.
- [27] E. Du, H. He, N. Zeng, M. Sun, Y. Guo, J. Wu, S. Liu, and H. Ma, "Mueller matrix polarimetry for differentiating characteristic features of cancerous tissues," *Journal of Biomedical Optics*, vol. 19, no. 7, p. 076013, 2014.

REFERENCIAS

- [28] S. Alali and I. A. Vitkin, "Polarized light imaging in biomedicine: emerging Mueller matrix methodologies for bulk tissue assessment," *Journal of Biomedical Optics*, vol. 20, no. 6, p. 061104, 2015.
- [29] I. Ahmad, A. Khaliq, M. Iqbal, and S. Khan, "Mueller matrix polarimetry for characterization of skin tissue samples: A review," *Photodiagnosis and Photodynamic Therapy*, vol. 30, p. 101708, 2020.