Universidad de la República
Facultad de Ingeniería

Microscopía de Matrices de Mueller para Diagnóstico Temprano en Muestras Completas de Tejidos Biológicos

Tesis presentada a la Facultad de Ingeniería de la Universidad de la República por

Roman Demczylo y Diego Silva Piedra

en cumplimiento parcial de los requerimientos para la obtención de los títulos de Ingeniero Físico Matemático para Roman Demczylo e Ingeniero Electricista para Diego Silva.

Tutores

Dr. Ariel Fernández ..... Universidad de la República (IFFI)
Dr. Federico Lecumberry ..... Universidad de la República (IIE)

Tribunal

Dr. Álvaro Gómez ..... Universidad de la República (IIE)
Dr. Lorenzo Lenci ..... Universidad de la República (IFFI)
Dr. Pablo Monzón ..... Universidad de la República (IIE)

Montevideo
martes 21 mayo, 2024

Microscopía de Matrices de Mueller para Diagnóstico Temprano en Muestras Completas de Tejidos Biológicos, Roman Demczylo y Diego Silva Piedra.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 169 páginas.
Compilada el martes 21 mayo, 2024.
`http://iie.fing.edu.uy/`

si vives así me dijo
el farol que parpadeaba
entre abandonar la noche
y dormir con la mañana

bailando en la cuerda floja
a pura puerta cerrada
censando gatos y envidos
acólito de barajas

si vives así me dijo
el farol o la ventana
un cartel preso en un muro
una paloma dañada
el batallón de una verja
un aroma de gencianas

no vas a reconocerte
cuando te encuentres la cara

pero yo no le hice caso
porque el aire me sobraba

Washington Benavides

Esta página ha sido intencionalmente dejada en blanco.

Reconocimientos

Al Grupo de Óptica Aplicada del Instituto de Física, principalmente a Ariel Fernández, por guiarnos; y a Alejandro Silva, por el asesoramiento con el Hardware. A Federico Lecumberry, por sumarse a la idea y guiarnos. A la comisión de carrera de Ingeniería Físico-Matemática, principalmente a Pablo Monzón, por aceptar la propuesta y facilitar los trámites. A Antonio Sáez, por el recorte y diseño de piezas mecánicas. A Bruno Schuty y Leonel Malacrida, por el asesoramiento y facilitar las muestras de tejido de piel humana. A nuestros familiares y compañeros por el apoyo.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

La polarización es una propiedad fundamental de la luz que puede ser modelada a través de cuatro números reales conocidos como parámetros de Stokes. Cuando la luz interactúa con un objeto, su estado de polarización se transforma a través de una función de transferencia conocida como matriz de Mueller. Esta matriz describe de forma completa las características polarimétricas del objeto, las cuales tienen asociación directa con sus propiedades físicas, principalmente de naturaleza mecánica. En los últimos años, surgió la imaginería polarimétrica de Mueller: una técnica capaz de medir las matrices de Mueller sobre un campo de visión acotado, permitiendo visualizar las características físicas en un mapa bidimensional.

Este trabajo presenta el diseño e implementación de un sistema de microscopía de matrices de Mueller para el análisis de tejidos biológicos de muestra completa. El sistema combina la adquisición de imágenes de alta resolución con el cálculo de matrices de Mueller para obtener información sobre la estructura y las propiedades físicas del tejido. Asimismo, se incorpora un algoritmo de *stitching* que permite la fusión de las capturas del microscopio para lograr caracterizar una muestra de tejido entera. La utilización de técnicas de aprendizaje automático puede habilitar la identificación automática de características relevantes y la clasificación de diferentes tipos de tejido y patologías. Se introduce un estudio de aprendizaje automático sobre las imágenes capturadas por el sistema de microscopía y sus características resultantes del cálculo de matrices de Mueller. Se exploran modelos de distinta complejidad, aplicándolos directamente sobre las imágenes o sobre vectores de características calculados a partir de las mismas, y se compara su desempeño en el conjunto de prueba.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de Contenidos

Esta página ha sido intencionalmente dejada en blanco.

Introducción

En muchos métodos de imaginería, los efectos de la polarización [1] de la luz suelen ser ignorados debido a la usual estructura aleatoria de los tejidos, lo que resulta en una rápida despolarización de la luz cuando se propaga en los mismos. Sin embargo, en ciertos tejidos biológicos, el grado de polarización (DoP) de la luz transmitida o reflejada es medible incluso para muestras de espesor considerable [2]. La información de la estructura del tejido puede extraerse a través de los cambios de la polarización de la luz, pudiendo ser de mucha utilidad para el diagnóstico temprano de diversas patologías, como por ejemplo cáncer [3].

Un avance significativo ha sido la incorporación del cálculo de matrices de Mueller, una herramienta que ha generado atención debido a sus ventajas para distinguir cambios en la estructura del tejido de forma no invasiva [4]. Este método es la forma completa de caracterizar las propiedades polarimétricas del tejido biológico. Existen varios métodos de análisis de matrices de Mueller para derivar grupos de parámetros con claras asociaciones a las propiedades físicas de los tejidos, de los cuales se destaca la descomposición de Lu-Chipman [5].

A efectos de cuantificar el estado de polarización (SoP) de la luz, la forma completa de hacerlo es a través de cuatro números reales denominados parámetros de Stokes. En imaginería polarimétrica, una forma adecuada de medir los parámetros de Stokes es mediante las técnicas de división del plano focal (DoFP) [6]. Debido a los avances en nanotecnología, ha sido posible la incorporación de polarizadores lineales de micro-grilla metálica integrados directamente sobre el sensor de una cámara. En un superpíxel (conjunto de cuatro píxeles), cada píxel cuenta con un polarizador orientado con ángulo de polarización (AoP) distinto. Se hace necesario introducir un *de-mosaicing* de la máscara de polarización para poder extraer, en cada canal de color, cada SoP [7].

Tradicionalmente, el diagnóstico de enfermedades a partir de imágenes obtenidas por microscopía demanda tiempo y la presencia de un especialista que analice el caso. La posibilidad de capturar imágenes digitales de un microscopio y someterlas a distintos procesamientos, nos permite contribuir al diagnóstico médico. Uno de los avances más significativos es la introducción de escáneres de muestras completas (*Whole-Slide Scanner*) y la imaginería de muestra completa (WSI). Estos escáneres permiten digitalizar una muestra entera, generando imágenes de alta resolución que pueden ser visualizadas, analizadas y compartidas. Sin embargo, el diagnóstico convencional se ha basado principalmente en la observación visual de tejidos teñidos y en la experiencia del patólogo para identificar la lesión [8].

En este contexto, el proceso de adquisición de imágenes de alta resolución a partir de muestras completas de tejido biológico plantea desafíos significativos debido a la necesidad de unir y combinar múltiples imágenes para formar una representación coherente y completa del tejido. El proceso de unión de imágenes, conocido como *Image Stitching* [9], es la herramienta por excelencia para lograrlo. El desarrollo de algoritmos de *stitching* eficientes y precisos es fundamental para lograr una captura de imágenes completa y de alta calidad en el área de microscopía, especialmente en circunstancias donde los equipos de alta gama como los escáneres mencionados no están disponibles. En microscopía, un buen algoritmo de *stitching* debe ser capaz de sobrellevar las posibles variaciones de las condiciones fotométricas y geométricas entre imágenes capturadas a través del microscopio, resultando en una imagen final nítida y sin distorsiones cromáticas ni geométricas.

Si bien sigue prevaleciendo la observación de expertos como método de identificación de posibles enfermedades en las muestras de tejido, en los últimos años y de la mano con el rápido desarrollo de tecnologías relativas a la inteligencia artificial, se han buscado maneras de auxiliar y de imitar esta capacidad humana. El Aprendizaje Automático (*Machine Learning*) [10] juega un papel cada vez más importante en la mejora y automatización de los procesos de análisis de imágenes médicas. Los algoritmos de aprendizaje automático pueden utilizarse para identificar automáticamente características relevantes en las imágenes, clasificar diferentes tipos de tejido y patologías, y ayudar en la interpretación de los resultados. Integrar técnicas de aprendizaje automático en el proceso de captura y análisis de imágenes de microscopía puede permitir una evaluación más rápida y precisa de los tejidos, así como la detección temprana de enfermedades y anomalías. Más en concreto, combinar la caracterización específica de los tejidos que brindan las matrices de Mueller con aprendizaje automático representa un frente innovador y prometedor a tal efecto [11].

Este trabajo se centra en el diseño de un sistema de microscopía de matrices de Mueller para el análisis de tejidos biológicos a muestra completa (*Mueller Whole-Slide Imaging*). El objetivo es capturar de manera automática todas las imágenes requeridas para construir una matriz de Mueller de toda la muestra. De esta manera, se establece un sistema multimodal que permite analizar el tejido no solo a través de su transmitancia, sino también mediante sus propiedades polarimétricas. En particular, los algoritmos de aprendizaje automático se ven beneficiados por una mayor cantidad de canales de información, lo que provee un reconocimiento de características más robusto.

En el Capítulo 1, describiremos los parámetros de Stokes y las matrices de Mueller. Mostraremos cómo se calculan y cómo extraer sus características físicas utilizando la descomposición de Lu-Chipman.

En el Capítulo 2, presentaremos el diseño del microscopio. Mostraremos las piezas fabricadas mediante impresión 3D, el sistema motorizado, la arquitectura digital y los elementos ópticos.

En el Capítulo 3, mostraremos los algoritmos para calcular la matriz de Mueller, cómo hicimos el escaneo de muestra completa y el diseño de la interfaz gráfica de usuario (GUI) para controlar el sistema de forma íntegra.

En el Capítulo 4, detallaremos el algoritmo empleado para hacer *stitching* y cómo lo adaptamos para realizar el *stitching* de matrices. Expondremos la calidad del algoritmo y cómo medimos la cantidad de fotos necesarias para fusionar un área de tejido dada.

Finalmente, el Capítulo 5 estará dedicado al diagnóstico de tejido de piel humana. Analizamos cuatro algoritmos de reconocimiento de patrones: *Support Vector Machine* (SVM), *Random Forest* (RF), *Multi-Layer Perceptron* (MLP) y *Convolutional Neural Network* (CNN). Entrenamos cada algoritmo con muestras clasificadas como melanoma y nevo, y presentaremos los resultados obtenidos.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Polarización y Óptica Electromagnética

La óptica electromagnética es la rama de la física que estudia la luz y la radiación próxima al espectro visible mediante la electrodinámica clásica. Si bien este enfoque es bastante preciso, existen fenómenos físicos relacionados con la radiación que deben ser abordados únicamente desde el punto de vista de la electrodinámica cuántica. En el límite de las bajas frecuencias (ondas de radio o menos), la teoría electromagnética clásica constituye un modelo bastante adecuado para tratar las ondas electromagnéticas. Cuando tratamos frecuencias en el espectro visible, incluyendo infrarrojo y ultravioleta, hablamos de óptica clásica [12].

En el año 1865 se formularon las ecuaciones de Maxwell, estableciendo así el vínculo entre la óptica y la teoría electromagnética:

$$\boldsymbol{\nabla} \cdot \boldsymbol{E} = \frac{\rho}{\epsilon_0} \,, \tag{1.1}$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{B} = 0 \,, \tag{1.2}$$

$$\boldsymbol{\nabla} \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} \,, \tag{1.3}$$

$$\boldsymbol{\nabla} \times \boldsymbol{B} = \boldsymbol{J} + \epsilon_0 \mu_0 \frac{\partial \boldsymbol{E}}{\partial t} \,. \tag{1.4}$$

Ecuaciones conocidas como Ley de Gauss eléctrica, Ley de Gauss magnética, Ley de Faraday y Ley de Ampere-Maxwell, respectivamente. Los campos $\boldsymbol{E}$ y $\boldsymbol{B}$ son conocidos como campo eléctrico y campo magnético. Los campos ρ y $\boldsymbol{J}$ son los términos fuentes, conocidos como densidad de carga eléctrica y densidad de corriente eléctrica. La constante $\epsilon_0 = 8{,}85 \times 10^{-12}\, C^2 N^{-1} m^{-2}$ es la permitividad eléctrica del vacío y la constante $\mu_0 = 4\pi \times 10^{-7}\, TmA^{-1}$ es la permeabilidad magnética del vacío.

1.1. Ondas Electromagnéticas

Una onda electromagnética es una perturbación del medio que propaga energía en forma de radiación a través de una componente eléctrica y otra magnética. Utilizando las ecuaciones de Maxwell, si suponemos que no hay fuentes libres y el medio es lineal, isótropo y homogéneo, se deduce (ver Apéndice A.1):

$$\boldsymbol{\nabla}^2 \boldsymbol{E} - \frac{n^2}{c_0^2} \frac{\partial^2 \boldsymbol{E}}{\partial^2 t} = 0, \tag{1.5}$$

$$\boldsymbol{\nabla}^2 \boldsymbol{B} - \frac{n^2}{c_0^2} \frac{\partial^2 \boldsymbol{B}}{\partial^2 t} = 0. \tag{1.6}$$

Estas ecuaciones son las que modelan el comportamiento de la onda. La velocidad de propagación de la onda electromagnética, $c_0 = 1/\sqrt{\epsilon_0 \mu_0}$, es la velocidad de la luz en el vacío. En este sentido, consideramos que todo haz de luz se comporta como una onda electromagnética. Por las propiedades de este tipo de medios (ver ejemplos en la Tabla 1.1), la velocidad de la luz decae como $c = c_0/n$, siendo n el índice de refracción del medio. Además, en la mayoría de los problemas alcanza con analizar únicamente el comportamiento de la componente eléctrica.

Medio	Índice de Refracción
Aire	1.00
Agua	1.33
Vidrio (BK7)	1.52
Aceite de inmersión	1.52

Tabla 1.1: Índice de refracción en el espectro visible.

Supongamos que la onda se propaga en un medio con las características mencionadas anteriormente. En un sistema de coordenadas cartesiano, la ecuación de ondas vectorial (1.5) se simplifica en tres ecuaciones de onda escalares:

$$\boldsymbol{\nabla}^2 E_i(\boldsymbol{r}, t) - \frac{1}{c^2} \frac{\partial^2 E_i(\boldsymbol{r}, t)}{\partial t^2} = 0, \quad i = x, y, z. \tag{1.7}$$

Una onda es plana si los puntos del espacio donde la fase es constante representan un plano. Además, es monocromática si su variación temporal es sinusoidal. Matemáticamente, las ondas monocromáticas representan las componentes principales de Fourier de cualquier onda más compleja. Consideremos como solución de (1.7) una onda plana, monocromática, de frecuencia $\omega = 2\pi\nu$ y que se propaga en el eje generado por el vector $\boldsymbol{k}$. La forma de la onda se puede expresar, en cada coordenada y sin pérdida de generalidad, en notación fasorial (ver Apéndice A.2):

$$E_i(z, t) = \hat{E_{0i}} e^{i(\omega t - kz + \delta_i)}, \quad i = x, y, z, \tag{1.8}$$

siendo $\omega = kc$ la relación de dispersión y $\boldsymbol{k} = (0, 0, k)^T$ un vector de onda elegido sin pérdida de generalidad. Si estamos lo suficientemente lejos de la fuente, tomando la divergencia del campo eléctrico y utilizando Ley de Gauss:

$$\boldsymbol{\nabla} \cdot \boldsymbol{E} = \boldsymbol{k} \cdot \hat{\boldsymbol{E}}_{\boldsymbol{0}} = 0. \tag{1.9}$$

Entonces, el campo eléctrico oscila transversalmente a su dirección de propagación y $E_{0z} = 0$. Este resultado es importante porque permite definir la polarización de la onda. Podemos expresar finalmente el campo eléctrico en dos dimensiones:

$$\begin{cases} E_x(z, t) = E_{0x} \cos(\omega t - kz + \delta_x), \\ E_y(z, t) = E_{0y} \cos(\omega t - kz + \delta_y), \end{cases} \tag{1.10}$$

siendo E_0 la amplitud, ω la frecuencia, k el vector de onda y δ la fase. El par ordenado $\boldsymbol{J} = (E_{0x}, E_{0y} e^{j(\delta_y - \delta_x)})^T$ es conocido como vector de Jones y define el SoP de una onda monocromática cualquiera.

1.2. Parámetros de Stokes

Tenemos dos limitaciones en el modelo de la sección anterior. Primero, las componentes transversales del campo eléctrico de la luz visible oscilan en un período de tiempo del orden de $1fs$. Este período de tiempo es muy corto para ser medido por un instrumento de medida. Segundo, solo aplica a ondas monocromáticas, por lo que no puede ser utilizado para caracterizar, por ejemplo, la luz estructurada emitida por un LED (*Light Emission Diode*).

En el año 1852, George Gabriel Stokes (1819–1903) descubrió que la polarización de la luz podía caracterizarse a través de cuatro cantidades reales, conocidas finalmente como parámetros de Stokes. Estos se basan en la intensidad media, una cantidad medible y capaz de caracterizar la polarización de cualquier haz de luz.

Uno de los mayores aportes de Stokes fue el modelado de la luz completamente despolarizada. Definió experimentalmente la luz despolarizada como aquella cuya intensidad no es afectada por la presencia de un polarizador rotado a cualquier ángulo y un retardador de cualquier valor de retardancia. Este resultado fue imprescindible para la verificación de las ecuaciones de Maxwell en el espectro visible.

1.2.1. Luz Monocromática

Las ondas monocromáticas constituyen un modelo teórico elemental que funciona como base para entender problemas más complejos. Este tipo de onda se caracteriza por oscilar a una única frecuencia y tener una polarización bien definida. Si consideramos un campo eléctrico $\boldsymbol{E}$ que cumple las ecuaciones (1.10), sus parámetros de Stokes s_0, s_1, s_2 y s_3 se calculan (ver Apéndice A.3):

$$s_0 = E_{0x}^2 + E_{0y}^2 \,, \tag{1.11}$$

$$s_1 = E_{0x}^2 - E_{0y}^2 \,, \tag{1.12}$$

$$s_2 = 2E_{0x}E_{0y}\cos(\delta) \,, \tag{1.13}$$

$$s_3 = 2E_{0x}E_{0y}\sin(\delta) \,. \tag{1.14}$$

En este caso, la función $\boldsymbol{E}(z_0, t)$ con z_0 constante describe una curva conocida como elipse de polarización. El parámetro $\psi \in [0, \pi]$ es el Ángulo de Polarización (AoP) de la onda y mide el ángulo de rotación de la elipse. El AoP se relaciona con los parámetros de Stokes de la siguiente forma:

$$\tan(2\psi) = \frac{2E_{0_x}E_{0_y}\cos(\delta)}{E_{0_x}^2 - E_{0_y}^2} = \frac{s_2}{s_1}. \tag{1.15}$$

El parámetro $\chi \in [0, \pi/2]$ es el ángulo de elipticidad y mide el nivel de compresión de la elipse. Además, verifica la ecuación:

$$\sin(2\chi) = \frac{2E_{0x}E_{0y}\sin(\delta)}{E_{0x}^2 + E_{0y}^2} = \frac{s_3}{s_0}. \tag{1.16}$$

Si $\chi \in \{0, \pi/2\}$, entonces $s_3 = \sin(2\chi)s_0 = 0$ y la onda está **linealmente polarizada**. Si, en cambio, $\chi = \pi/4$, entonces $s_3 = s_0$ y la onda está circularmente polarizada. En cualquier otro caso la onda está elípticamente polarizada, como se puede apreciar en la Figura 1.1. Para más detalles nos referimos al Apéndice A.4.

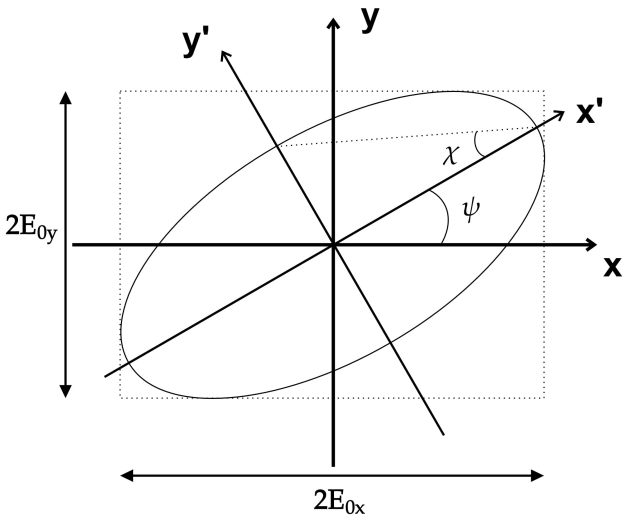

Figura 1.1: Elipse de polarización para una onda monocromática. Realizado en Mathcha [13].

1.2.2. Luz Natural

La luz natural consiste de una mezcla infinita de ondas monocromáticas simples oscilando a distintas frecuencias. Uno de los ejemplos más importantes es la luz solar. A diferencia de la luz monocromática, la luz natural no tiene por qué tener una polarización bien definida. En este caso, los parámetros de Stokes se expresan como el promedio de las N contribuciones monocromáticas cuando $N \to \infty$:

$$S_0 = \frac{1}{N}\sum_{i=1}^{N} s_0^{(i)}, \;\; S_1 = \frac{1}{N}\sum_{i=1}^{N} s_1^{(i)}, \;\; S_2 = \frac{1}{N}\sum_{i=1}^{N} s_2^{(i)}, \;\; S_3 = \frac{1}{N}\sum_{i=1}^{N} s_3^{(i)}. \tag{1.17}$$

Definimos entonces el vector de Stokes [1]:

$$\boldsymbol{S} = \begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix}. \tag{1.18}$$

La cantidad S_0 mide la intensidad media y representa la luz que vemos a simple vista. Las cantidades S_1, S_2 y S_3 miden cómo está polarizada la luz y son invisibles para el ojo humano. Para medir qué tan polarizado está un rayo de luz se utiliza una cantidad denominada **Grado de Polarización** (DoP):

$$\boxed{DoP = \frac{\sqrt{S_1^2 + S_2^2 + S_3^2}}{S_0}.} \tag{1.19}$$

El DoP es una cantidad real entre 0 y 1 que representa el porcentaje de la luz que está polarizada. En particular, si $DoP = 0$ la luz está completamente despolarizada y si $DoP = 1$ está completamente polarizada. En general, la luz está

[1] Matemáticamente no es un vector, pero se suele llamar así.

parcialmente polarizada y el vector de Stokes se puede escribir como una combinación lineal y convexa de un estado completamente polarizado y otro despolarizado:

$$\underbrace{\begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix}}_{\substack{\text{Parcialmente} \\ \text{Polarizado}}} = DoP \times \underbrace{\begin{pmatrix} S_0 \\ \frac{S_1}{DoP} \\ \frac{S_2}{DoP} \\ \frac{S_3}{DoP} \end{pmatrix}}_{\substack{\text{Completamente} \\ \text{Polarizado}}} + (1 - DoP) \times \underbrace{\begin{pmatrix} S_0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{\substack{\text{Completamente} \\ \text{Despolarizado}}} . \quad (1.20)$$

Si $S_1 > 0$ o $S_2 > 0$, el **Ángulo de Polarización** (1.15) de la luz natural es [2]:

$$AoP = \frac{1}{2} \arctan 2 \left(\frac{\frac{S_2}{DoP}}{\frac{S_1}{DoP}} \right) = \frac{1}{2} \arctan 2 \left(\frac{S_2}{S_1} \right). \quad (1.21)$$

1.3. Matrices de Mueller

Hemos descrito las propiedades fundamentales de la luz y cómo caracterizar su polarización a través del vector de Stokes. Ahora, nos concentraremos en las propiedades de los elementos capaces de cambiar la polarización de la luz.

En el año 1943, Hans Mueller (1900 - 1965) introdujo el uso de los parámetros de Stokes en la resolución de problemas ópticos. Como el estado de la luz antes y después de atravesar un dispositivo óptico se representa con vectores de Stokes $\boldsymbol{S}$ y $\boldsymbol{S}'$ respectivamente, entonces, suponiendo un modelo lineal, el dispositivo puede ser representado por una matriz cuadrada real $\boldsymbol{M}$, denominada matriz de Mueller:

$$\boldsymbol{S}' = \begin{pmatrix} S'_0 \\ S'_1 \\ S'_2 \\ S'_3 \end{pmatrix} = \begin{pmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{pmatrix} \begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix} = \boldsymbol{M}\boldsymbol{S}. \quad (1.22)$$

El caso de estudio más simple es el medio isotrópico. En este tipo de materiales, solo existe transmitancia y la polarización de la luz no cambia. Si definimos T a la **transmitancia**, la matriz de Mueller del medio queda:

$$\boldsymbol{M} = \begin{pmatrix} T & 0 & 0 & 0 \\ 0 & T & 0 & 0 \\ 0 & 0 & T & 0 \\ 0 & 0 & 0 & T \end{pmatrix}. \quad (1.23)$$

Si, en cambio, el medio es anisotrópico, existen cuatro formas elementales de cambiar el SoP de un haz de luz: diatenuación lineal (polarización lineal), retardancia lineal (birrefringencia), retardancia circular (rotación) y despolarización.

En la Figura 1.2 se aprecia un rayo de luz con vector de Stokes $\boldsymbol{S}$ incidiendo sobre un elemento polarizador con matriz de Mueller $\boldsymbol{M}$. Al atravesar el medio, el vector de Stokes de entrada se transforma en un nuevo vector de Stokes $\boldsymbol{S}'$.

[2] La función arctan 2 es el arcotangente definido por ramas y se puede encontrar en `https://numpy.org/doc/stable/reference/generated/numpy.arctan2.html`

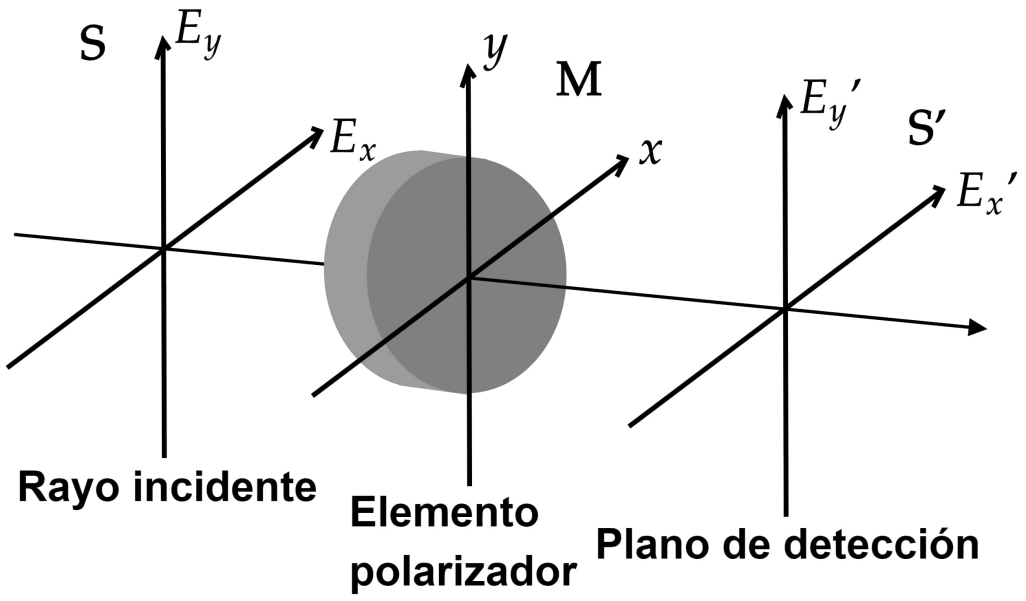

Figura 1.2: Haz de luz atravesando un elemento polarizador. Realizado en Mathcha.

1.3.1. Matriz de Rotación

El rotador de ángulo θ es un elemento que rota un ángulo θ las componentes principales del campo eléctrico. Una de las formas más comunes de hacerlo es a través de la presencia de moléculas quirales presentes en ciertos materiales, fenómeno conocido como actividad óptica o **retardancia circular**. Además, la matriz de Mueller del rotador es particularmente útil para hallar las matrices de Mueller de los demás elementos polarizadores rotados. La matriz de Mueller del rotador con retardancia circular θ se puede calcular (ver Apéndice A.5.1):

$$\boldsymbol{M}_{Rot}(2\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(2\theta) & -\sin(2\theta) & 0 \\ 0 & \sin(2\theta) & \cos(2\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{1.24}$$

1.3.2. Matriz de Polarización Lineal

El polarizador o diatenuador atenúa las componentes ortogonales del campo eléctrico de forma diferenciada. Según su uso, se le llama generador si crea luz polarizada o analizador si está inmediatamente antes del dispositivo de detección. Generalmente, funcionan mediante la absorción de la energía electromagnética en una componente privilegiada, propiedad denominada dicroísmo. Existen dos tipos de polarizador: lineal y circular. El polarizador lineal atenúa las componentes lineales del campo, a diferencia del circular que atenúa las componentes circulares.

Los parámetros que definen al polarizador lineal son los factores de atenuación de las componentes ortogonales del campo. Si definimos p_x y p_y a los factores de

atenuación en los ejes $\hat{x}$ e $\hat{y}$ respectivamente, entonces la **diatenuación lineal** es una cantidad entre 0 y 1 definida (ver Apéndice A.5.2):

$$D_L = \frac{p_x^2 - p_y^2}{p_x^2 + p_y^2}, \;\; p_x > p_y. \tag{1.25}$$

Un caso importante es el polarizador lineal ideal con eje de polarización en el eje $\hat{x}$. Como la polarización es ideal, $p_y = 0$ y la diatenuación lineal es unitaria, entonces la matriz de Mueller se calcula:

$$\boldsymbol{M}_H = \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{1.26}$$

Si queremos rotar un elemento polarizador con matriz de Mueller $\boldsymbol{M}$ un ángulo θ respecto al eje $\hat{x}$, debemos computar el siguiente cálculo (ver Apéndice A.5.1):

$$\boldsymbol{M}_{D,\theta} = \boldsymbol{M}_{Rot}(2\theta)\boldsymbol{M}\boldsymbol{M}_{Rot}(-2\theta). \tag{1.27}$$

Haciendo cuentas:

$$\boldsymbol{M}_{D,\theta} = \frac{1}{2}\begin{pmatrix} 1 & \cos(2\theta) & \sin(2\theta) & 0 \\ \cos(2\theta) & \cos^2(2\theta) & \sin(2\theta)\cos(2\theta) & 0 \\ \sin(2\theta) & sin(2\theta)\cos(2\theta) & \sin^2(2\theta) & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{1.28}$$

El caso más general de un diatenuador puede definirse a través del vector de diatenuación $\boldsymbol{D} = \big(d_1, d_2, d_3\big)^T$. El mismo se define de forma tal que el vector de Stokes $\boldsymbol{S} = \big(1, \boldsymbol{D}\big)^T$ sea vector propio de la matriz de Mueller del diatenuador. Además, la diatenuación D y el ángulo de diatenuación θ se calculan:

$$D = \|\boldsymbol{D}\|, \;\; \theta = \frac{1}{2}\arctan 2\left(\frac{d_2}{d_1}\right). \tag{1.29}$$

La forma normalizada más general de la matriz de Mueller del diatenuador es:

$$\boldsymbol{M}_D = \frac{1}{2}\begin{pmatrix} 1 & \boldsymbol{D}^T \\ \boldsymbol{D} & \alpha\boldsymbol{I}_{3\times 3} + (1-\alpha)(\hat{\boldsymbol{D}} \otimes \hat{\boldsymbol{D}}) \end{pmatrix}, \tag{1.30}$$

siendo $\alpha = \sqrt{1 - D^2}$.

Un experimento importante es cuando un rayo de luz completamente despolarizado incide sobre un polarizador lineal e ideal con ángulo de diatenuación θ. El SoP del rayo transmitido $\boldsymbol{S}'$ se puede calcular de la siguiente forma:

$$\boldsymbol{S}'_\theta = \boldsymbol{M}_{D,\theta}\begin{pmatrix} S_0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \frac{S_0}{2}\begin{pmatrix} 1 \\ \cos(2\theta) \\ \sin(2\theta) \\ 0 \end{pmatrix}. \tag{1.31}$$

Entonces, el vector de Stokes $\boldsymbol{S}'_\theta$ representa todos los posibles estados de polarización lineal, siendo θ el ángulo de polarización de la onda.

1.3.3. Matriz de Retardo Lineal

El retardador lineal introduce un retraso ϕ entre las componentes ortogonales del campo eléctrico. Específicamente, la componente del eje lento, $\hat{e_2}$, se retrasa un ángulo ϕ respecto a la componente del eje rápido, $\hat{e_1}$.

Su origen se debe a la dependencia del índice de refracción respecto al ángulo de polarización de la onda incidente. Este fenómeno fue observado por primera vez en cristales (birrefringencia natural), pero también aparece en materiales tensionados como el plástico (birrefringencia artificial).

Si denominamos n_x y n_y a los índices de refracción en los ejes $\hat{e_1}$ y $\hat{e_2}$ respectivamente, entonces $n_y > n_x$ y el ángulo ϕ denominado **retardancia lineal** se calcula:

$$\phi = \frac{2\pi(n_y - n_x)z}{\lambda}, \tag{1.32}$$

siendo z el espesor del retardador y λ la longitud de onda.

La matriz de Mueller del retardador lineal con desfasaje ϕ y eje rápido coincidente con el eje horizontal, $\hat{e_1} = \hat{x}$, se calcula (ver Apéndice A.5.3):

$$\boldsymbol{M}_R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\phi) & \sin(\phi) \\ 0 & 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix}. \tag{1.33}$$

En general, podemos escribir:

$$\boldsymbol{M}_R = \begin{pmatrix} 1 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{m}_R \end{pmatrix}, \tag{1.34}$$

siendo el sub-bloque $\boldsymbol{m}_R$ una matriz unitaria.

Un ejemplo importante es el retardador lineal de cuarta longitud de onda ($\phi = 90°$) con eje rápido en $\hat{x}$:

$$\boldsymbol{M}_{QWP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{pmatrix}. \tag{1.35}$$

1.3.4. Matriz de Despolarización

El despolarizador transforma la intensidad polarizada en despolarizada, principalmente por medio del fenómeno de *scattering*. El *scattering* o esparcimiento es el fenómeno físico que altera la dirección de la onda cuando atraviesa un medio localmente irregular. En el caso de los tejidos biológicos, esto se debe a su estructura usualmente aleatoria, lo que resulta en una pérdida de energía polarizada.

El despolarizador anisotrópico no ideal con sus ejes principales en $\hat{x}$ e $\hat{y}$ tiene la matriz de Mueller:

$$\boldsymbol{M}_\Delta = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \Delta_1 & 0 & 0 \\ 0 & 0 & \Delta_2 & 0 \\ 0 & 0 & 0 & \Delta_3 \end{pmatrix}, \tag{1.36}$$

siendo Δ_1, Δ_2 y Δ_3 tres parámetros reales entre 0 y 1 asociados a una propiedad del material conocida como **despolarizancia**. Si en particular se cumple, $\Delta_1 = \Delta_2 = \Delta_3 = \Delta$, hablamos de un despolarizador isotrópico. Si además, $\Delta = 0$, se trata de un despolarizador ideal.

Puesto que una matriz de Mueller tiene dieciséis grados de libertad, de lo cuales siete representan fenómenos físicos que no despolarizan (tres para retardancia, tres para diatenuación y uno para transmitancia), quedan nueve grados para modelar la despolarización. La forma más general de una matriz de despolarización es:

$$\boldsymbol{M}_\Delta = \begin{pmatrix} 1 & \boldsymbol{0}^T \\ \boldsymbol{P}_\Delta & \boldsymbol{m}_\Delta \end{pmatrix}, \tag{1.37}$$

siendo $\boldsymbol{P}_\Delta$ un vector y $\boldsymbol{m}_\Delta$ una matriz simétrica.

1.4. Polarimetría Lineal

La rama de la ciencia que mide la polarización de la luz se llama polarimetría. Cuando utilizamos polarizadores lineales y retardadores lineales, podemos calcular los cuatro parámetros de Stokes, procedimiento denominado polarimetría completa o *full Stokes*. En caso contrario hablamos de polarimetría incompleta.

Este tipo de técnicas recae sobre el uso de fotodetectores modernos, sensores basados en semiconductores. Estos dispositivos son capaces de medir la intensidad de campo eléctrico incidente promediada en un período de tiempo T, comúnmente llamado tiempo de exposición.

La mayoría de las técnicas que se utilizan en polarimetría recaen sobre dos familias: Polarimetría por División del Tiempo (DoTP) y Polarimetría por División del Plano Focal (DoFP). En las técnicas DoTP se registran los datos secuencialmente espaciados en el tiempo. En cambio, en las técnicas DoFP se toman los datos de forma instantánea utilizando un arreglo de polarizadores.

1.4.1. Cálculo de Stokes

En las secciones anteriores mostramos que cualquier SoP de la luz se puede representar de forma completa con un vector de Stokes. Sin embargo, hasta ahora carecemos de una metodología capaz de calcularlo de forma práctica. Dado un rayo de luz, Stokes propone el uso de un polarizador ideal justo antes del plano de detección, denominado analizador. El polarizador se usa para obtener la intensidad media de la luz polarizada en un conjunto de configuraciones particulares.

Supongamos que queremos medir un haz de luz cuyo SoP se caracteriza por el vector de Stokes $\boldsymbol{S} = (S_0, S_1, S_2, S_3)^T$. Consideremos $\boldsymbol{M}_{D,\theta}$, la matriz de Mueller de un polarizador lineal ideal con ángulo de diatenuación θ (1.28). El SoP de la luz transmitida, $\boldsymbol{S'}$, se modela mediante el producto: $\boldsymbol{S'} = \boldsymbol{MS}$. Como la cuarta columna de $\boldsymbol{M}_{D,\theta}$ está llena de ceros, la componente S_3 del vector de Stokes carece de interés. Si quisiéramos hallar S_3, sería necesario agregar elementos de retardo lineal al sistema de detección. Para facilitar el problema, nos concentraremos en

los tres primeros parámetros de Stokes: S_0, S_1 y S_2. La intensidad total media de la luz transmitida es la componente S_0':

$$S_0' = I(\theta) = \frac{1}{2}[S_0 + \cos(2\theta)S_1 + \sin(2\theta)S_2]. \tag{1.38}$$

Notamos que si el haz está completamente despolarizado, $S_1 = S_2 = 0$ y la intensidad I no depende de θ, tal como observó Stokes. Por otro lado, si evaluamos la intensidad en cuatro ángulos particulares:

$$I(0^\circ) = I_0 = \frac{S_0 + S_1}{2}\,, \tag{1.39}$$

$$I(45^\circ) = I_{45} = \frac{S_0 + S_2}{2}\,, \tag{1.40}$$

$$I(90^\circ) = I_{90} = \frac{S_0 - S_1}{2}\,, \tag{1.41}$$

$$I(135^\circ) = I_{135} = \frac{S_0 - S_2}{2}\,. \tag{1.42}$$

Las cantidades I_0,I_{45},I_{90} y I_{135} son medibles experimentalmente y constituyen la única información que contamos para medir la polarización de la luz. Para inferir un cálculo de los parámetros de Stokes en función de las intensidades, debemos agruparlas en un vector $\boldsymbol{I} = (I_0, I_{45}, I_{90}, I_{135})^T$. Podemos escribir matricialmente:

$$\boldsymbol{I} = \begin{pmatrix} I_0 \\ I_{45} \\ I_{90} \\ I_{135} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix}\begin{pmatrix} S_0 \\ S_1 \\ S_2 \end{pmatrix} = \boldsymbol{A}_{ideal}\boldsymbol{S}. \tag{1.43}$$

Invertimos la relación a través de la pseudoinversa[3]:

$$\boldsymbol{S} = \boldsymbol{A}^{+}_{ideal}\boldsymbol{I} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}\boldsymbol{I}. \tag{1.44}$$

Finalmente, los parámetros de Stokes S_0, S_1 y S_2 se calculan:

$$\boxed{\begin{aligned} S_0 &= \frac{1}{2}[I_0 + I_{45} + I_{90} + I_{135}], \\ S_1 &= I_0 - I_{90}, \\ S_2 &= I_{45} - I_{135}. \end{aligned}} \tag{1.45}$$

Para medir qué tan linealmente polarizado está el haz de luz se utiliza el **Grado de Polarización Lineal** (DoLP). Esta es una versión alternativa del DoP que no utiliza S_3, definida como:

$$\boxed{DoLP = \frac{\sqrt{S_1^2 + S_2^2}}{S_0}.} \tag{1.46}$$

[3]Operación definida como: $A^+ = (A^T A)^{-1}A^T$, solo si $A^T A$ es invertible.

En particular, $DoLP = 1$ si el haz está linealmente polarizado y $DoLP = 0$ si el haz está completamente despolarizado o circularmente polarizado. En la Figura 1.3 podemos apreciar el haz de luz cuyo SoP $\boldsymbol{S}$ se quiere analizar, el polarizador lineal con matriz de Mueller $\boldsymbol{M}_{D,\theta}$ y el vector de intensidades $\boldsymbol{I}$.

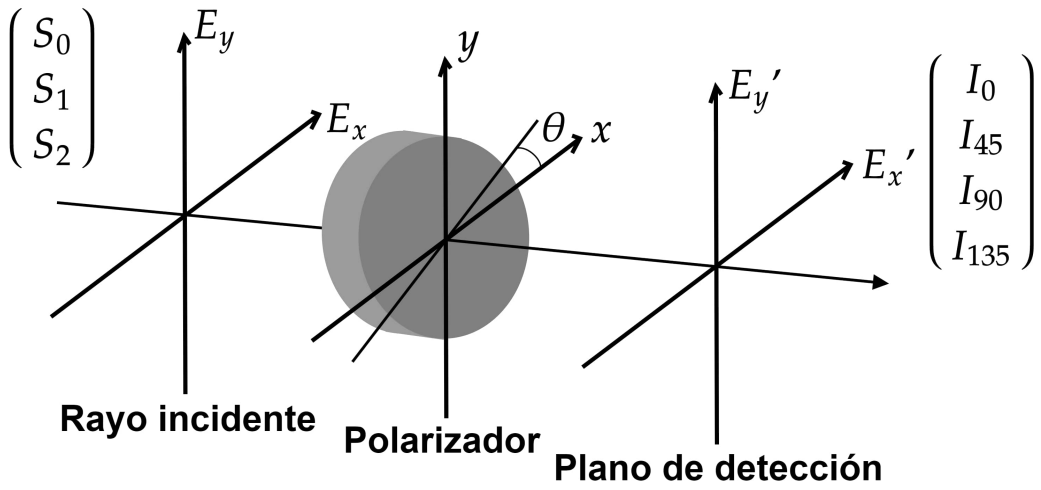

Figura 1.3: Polarimetría lineal de Stokes. El haz de luz con vector de Stokes $\boldsymbol{S} = (S_0, S_1, S_2)^T$ se polariza con cuatro ángulos de polarización $0°$, $45°$, $90°$ y $135°$. En cada caso se mide la intensidad total media en el plano de detección y se agrupa la información en un vector de intensidades $\boldsymbol{I} = (I_0, I_{45}, I_{90}, I_{135})^T$. Realizado en Mathcha.

1.4.2. Cálculo de Mueller

La matriz de Mueller es un arreglo bidimensional de 16 componentes reales que describe de forma completa las propiedades polarimétricas de cualquier material en reflexión o transmisión. La polarimetría de Mueller es completa si medimos todas las componentes e incompleta en caso contrario. Para ser completa, es necesario un sistema de detección de polarimetría de Stokes completa y un generador constituido por un polarizador lineal y un retardador lineal. En general, se suelen utilizar mecanismos giratorios para modular la polarización de la luz de la forma deseada.

Nos enfocaremos en hacer polarimetría de Mueller incompleta para medir 9 de las 16 componentes de la matriz de Mueller de un elemento polarizador cualquiera. Para hacerlo, es necesario conocer cómo cambia un conjunto controlado de rayos de luz polarizados linealmente al atravesar el medio.

Consideremos un medio con matriz de Mueller desconocida. Como estamos trabajando solo con polarizadores lineales, podemos llegar a medir los parámetros de Stokes S_0', S_1' y S_2' de la luz transmitida. Como además la luz incidente está linealmente polarizada, entonces $S_3 = 0$ y es válida la siguiente relación matricial:

$$\boldsymbol{S}' = \begin{pmatrix} S_0' \\ S_1' \\ S_2' \end{pmatrix} = \begin{pmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{pmatrix} \begin{pmatrix} S_0 \\ S_1 \\ S_2 \end{pmatrix} = \boldsymbol{M}_{3\times 3}\boldsymbol{S}. \tag{1.47}$$

Consideremos n rayos de luz con polarización lineal que inciden sobre el medio. Los estados de polarización de cada rayo deben ser lo suficientemente distintos entre sí para conocer cómo el elemento responde a distintos ángulos de polarización. Una forma sencilla de hacerlo es girando el polarizador que usamos como generador. Los vectores de Stokes incidentes $\boldsymbol{S}^{(1)}$, $\boldsymbol{S}^{(2)}$ ... $\boldsymbol{S}^{(n)}$ se pueden agrupar en una matriz de Stokes de entrada $\boldsymbol{\mathcal{S}}_{in}$:

$$\boldsymbol{\mathcal{S}}_{in} = \begin{pmatrix} S_0^{(1)} & S_0^{(2)} & \cdots & S_0^{(n)} \\ S_1^{(1)} & S_1^{(2)} & \cdots & S_1^{(n)} \\ S_2^{(1)} & S_2^{(2)} & \cdots & S_2^{(n)} \end{pmatrix}. \tag{1.48}$$

La respuesta del medio a este conjunto de estados se puede agrupar en una nueva matriz de Stokes de salida, $\boldsymbol{\mathcal{S}}_{out}$:

$$\boldsymbol{\mathcal{S}}_{out} = \boldsymbol{M}_{3\times 3}\boldsymbol{\mathcal{S}}_{in} = \begin{pmatrix} S_0'^{(1)} & S_0'^{(2)} & \cdots & S_0'^{(n)} \\ S_1'^{(1)} & S_1'^{(2)} & \cdots & S_1'^{(n)} \\ S_2'^{(1)} & S_2'^{(2)} & \cdots & S_2'^{(n)} \end{pmatrix}. \tag{1.49}$$

Si $n \geq 3$ y la matriz de Stokes, $\boldsymbol{\mathcal{S}_{in}}$, es pseudo invertible, entonces podemos inferir la matriz de Mueller, $\boldsymbol{M}_{3\times 3}$, a través del siguiente cálculo matricial:

$$\boxed{\boldsymbol{M}_{3\times 3} = \boldsymbol{\mathcal{S}_{out}}\boldsymbol{\mathcal{S}_{in}}^{+}.} \tag{1.50}$$

En la Figura 1.4 podemos ver el esquema básico de polarimetría de matrices de Mueller 3×3 con $n = 3$. La matriz $\boldsymbol{\mathcal{S}}_{in}$ se modula y para medirla alcanza con quitar la muestra y medir la matriz de Stokes vista. La matriz $\boldsymbol{\mathcal{S}}_{out}$ también es medible y cambia en función de la muestra.

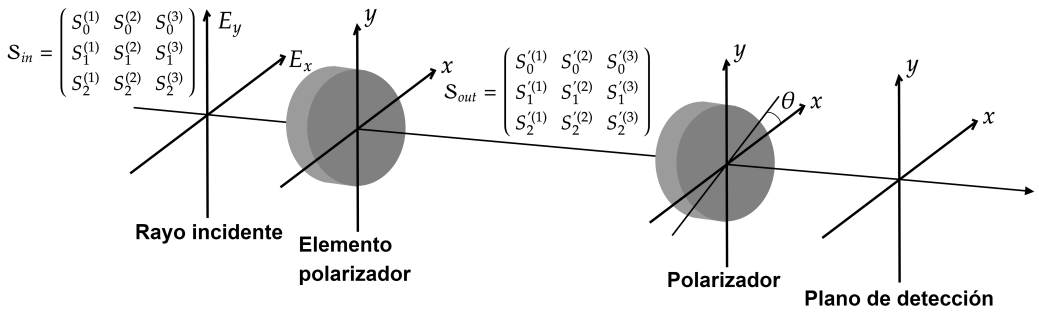

Figura 1.4: Polarimetría de Mueller 3×3. El elemento polarizado (medio) con matriz de Mueller $\boldsymbol{M}$ transforma la matriz de Stokes $\boldsymbol{\mathcal{S}_{in}}$ en una nueva matriz de Stokes $\boldsymbol{\mathcal{S}_{out}}$. Ambas matrices de Stokes se pueden medir utilizando polarimetría lineal de Stokes. Realizado en Mathcha.

1.5. Descomposición de Matrices de Mueller

Dada una matriz de Mueller experimental, nos gustaría poder separar la diatenuación, la retardancia y la despolarizancia. Una descomposición posible es la de

Lu-Chipman, la cual reescribe cualquier matriz de Mueller no singular [4] como un producto de tres matrices de Mueller elementales. Si $\boldsymbol{M}$ es una matriz de Mueller no singular, una posible descomposición tiene la siguiente forma:

$$\boldsymbol{M} = m_{00}\boldsymbol{M_\Delta}\boldsymbol{M_R}\boldsymbol{M_D}, \tag{1.51}$$

donde m_{00} es la transmitancia, $\boldsymbol{M_\Delta}$ la matriz de despolarizancia, $\boldsymbol{M_R}$ la matriz de retardancia y $\boldsymbol{M_D}$ la matriz de diatenuación (ver Apéndice A.6).

Cuando el medio tiene una única propiedad óptica, se puede interpretar la matriz de Mueller $\boldsymbol{M}$ como alguna de las matrices elementales vistas en la Sección 1.3. Si, en cambio, el medio tiene más de una propiedad, los elementos de la matriz se mezclan y la interpretación es más complicada [14]. En general nos referiremos a la Figura 1.5, donde podemos apreciar qué características se pueden extraer.

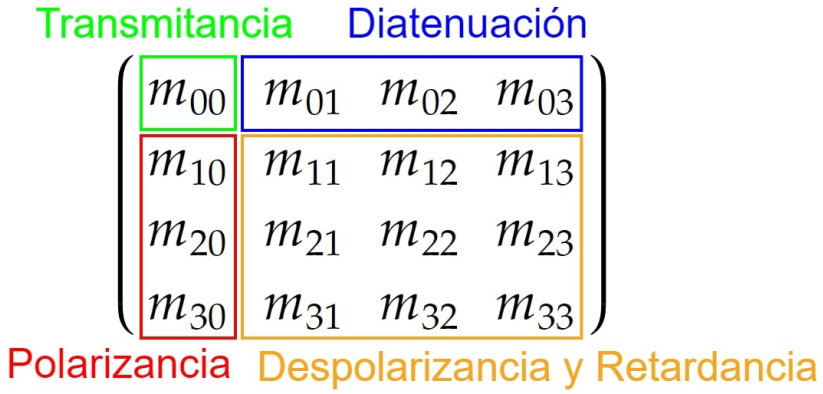

Figura 1.5: Interpretación de la matriz de Mueller. Realizado en Mathcha.

Podemos apreciar las características físicas [15] más importantes del medio: diatenuación lineal (1.54), polarizancia lineal (1.55), ángulo de polarización (1.56), poder de despolarizancia (1.57), retardancia lineal (1.53) y retardancia circular (1.52). Las fórmulas (1.55) y (1.56) son expresiones que respectivamente miden cuánto se polariza la luz y en qué dirección lo hace. La fórmula (1.57) es una cantidad real entre 0 y 1 que mide cuánto despolariza la luz. Las demás expresiones ya fueron presentadas en la Sección 1.2.

$$R_C = \arctan 2\left(\frac{\boldsymbol{M}_{R,21} - \boldsymbol{M}_{R,12}}{\boldsymbol{M}_{R,11} + \boldsymbol{M}_{R,22}}\right), \tag{1.52}$$

$$R_L = cos^{-1}\left(\sqrt{(\boldsymbol{M}_{R,11} + \boldsymbol{M}_{R,22})^2 + (\boldsymbol{M}_{R,21} - \boldsymbol{M}_{R,12})^2} - 1\right), \tag{1.53}$$

[4] Diatenuación no unitaria y parámetros de despolarizancia no nulos.

$$D_L = \frac{\sqrt{m_{01}^2 + m_{02}^2}}{m_{00}}, \quad (1.54)$$

$$P_L = \frac{\sqrt{m_{10}^2 + m_{20}^2}}{m_{00}}, \quad (1.55)$$

$$AoP = \frac{1}{2} \arctan 2 \left(\frac{m_{02}}{m_{01}} \right), \quad (1.56)$$

$$PoD = 1 - \frac{Tr(\boldsymbol{M}_\Delta) - 1}{3}. \quad (1.57)$$

1.6. Imaginería Polarimétrica

Hasta ahora hemos cubierto la teoría necesaria para obtener toda la información polarimétrica lineal de un haz de luz. A este tipo de análisis se lo conoce más comúnmente como elipsometría lineal. Si lo que queremos es hacer un análisis espacial de la luz que emite un objeto, lo que necesitamos hacer es usar técnicas de imaginería polarimétrica.

Para hacer imaginería debemos medir una cantidad física de la luz sobre un arreglo bidimensional de puntos en el plano objeto. Las medidas físicas más importantes son intensidad, frecuencia y polarización. Las cámaras convencionales miden en un número de rangos de canales entre 1 (monocromática) y 3 (color), los sistemas multiespectrales miden en el orden de 10 canales espectrales y los hiperespectrales pueden medir 300 o más.

Generalmente, la polarización varía lentamente en función de la longitud de onda, lo que nos provee de información que tiende a estar descorrelacionada con las medidas espectrales del sistema. Así como el espectro frecuencial nos da información sobre la composición del material, la polarización nos brinda información sobre su estructura, cohesión, elasticidad y aspereza.

La polarimetría se basa en la manipulación de un haz de luz con varios ángulos de polarización distintos. Entonces, para hacer imaginería polarimétrica debemos registrar un número secuencial o simultáneo de imágenes y procesarlas digitalmente. Existen diversos tipos de técnicas, cada una con sus respectivas ventajas y desventajas, de las cuales nos concentraremos en una de las más importantes: División del Plano Focal (DoFP).

1.6.1. División del Plano Focal

Los avances en la nanotecnología permitieron integrar elementos ópticos polarizadores directamente sobre el chip de las cámaras. Generalmente, los sistemas DoFP usan polarizadores lineales y trabajan prácticamente en cualquier región del espectro óptico. La mayoría de los sistemas tienen entrelazados una matriz de superpíxeles, un arreglo de cuatro píxeles polarizados en cuatro ángulos de polarización particulares: 0°, 45°, 90° y 135°. A partir de ahora a cualquier arreglo de píxeles lo denominaremos FPA (Focal Plane Array).

En la Figura 1.6 podemos observar el superpíxel de un sistema DoFP. Su principal ventaja es la capacidad de capturar las componentes lineales de Stokes en

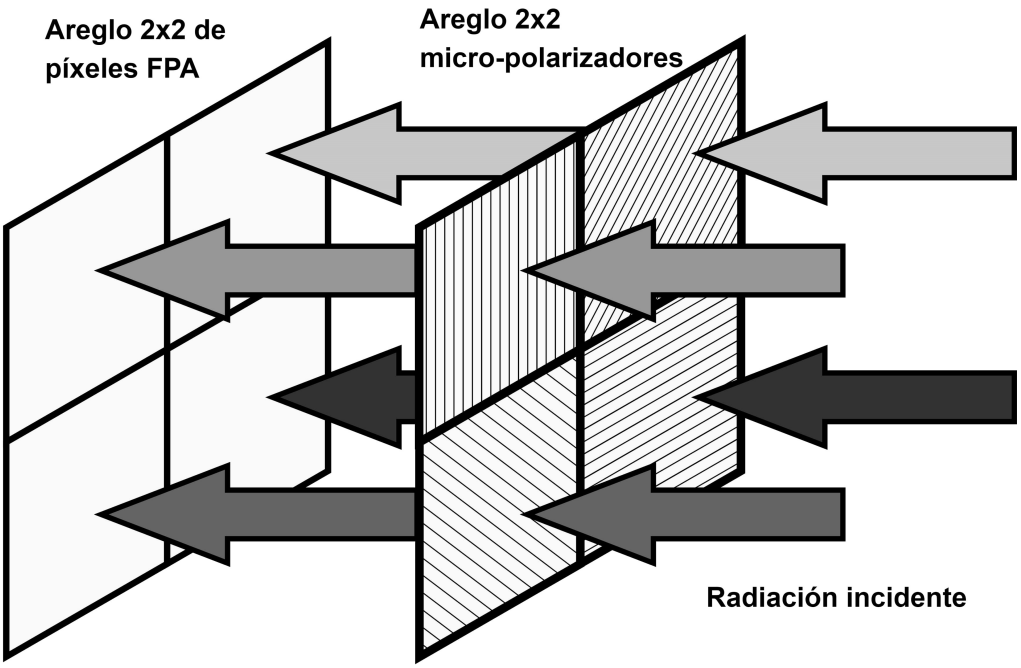

Figura 1.6: División del Plano Focal. Realizado en Mathcha.

tiempo real y en cada píxel de la imagen. Sin embargo, el sistema tiene por construcción un error de registro de un píxel, el cual puede ser mitigado naturalmente desenfocando un poco el sistema.

1.6.2. Calibración de Parámetros de Stokes

Cada sistema de polarimetría tiene sus propios problemas de alineación y calibración. Si se utiliza un FPA, la respuesta individual de cada detector debe ser conocida. Una corrección de la no uniformidad de los filtros es generalmente necesaria. Otros problemas comunes en polarimetría son el alineamiento de los ejes ópticos y las aberraciones polarizadas.

Supongamos un sistema DoFP con un arreglo como el de la Figura 1.6. Los cuatro polarizadores se pueden describir a través de sus vectores de diatenuación, uno por cada ángulo de polarización: $\boldsymbol{D}^{(0)}$, $\boldsymbol{D}^{(45)}$, $\boldsymbol{D}^{(90)}$ y $\boldsymbol{D}^{(135)}$.

Si un haz con vector de Stokes $\boldsymbol{S}$ incide sobre el FPA, tenemos entonces las siguientes intensidades observadas:

$$I_0 = S_0 + \boldsymbol{D}_1^{(0)} S_1 + \boldsymbol{D}_2^{(0)} S_2, \tag{1.58}$$

$$I_{45} = S_0 + \boldsymbol{D}_1^{(45)} S_1 + \boldsymbol{D}_2^{(45)} S_2, \tag{1.59}$$

$$I_{90} = S_0 + \boldsymbol{D}_1^{(90)} S_1 + \boldsymbol{D}_2^{(90)} S_2, \tag{1.60}$$

$$I_{135} = S_0 + \boldsymbol{D}_1^{(135)} S_1 + \boldsymbol{D}_2^{(135)} S_2. \tag{1.61}$$

Matricialmente:

$$\boldsymbol{I} = \begin{pmatrix} I_0 \\ I_{45} \\ I_{90} \\ I_{135} \end{pmatrix} = \begin{pmatrix} 1 & \boldsymbol{D}_1^{(0)} & \boldsymbol{D}_2^{(0)} \\ 1 & \boldsymbol{D}_1^{(45)} & \boldsymbol{D}_2^{(45)} \\ 1 & \boldsymbol{D}_1^{(90)} & \boldsymbol{D}_2^{(90)} \\ 1 & \boldsymbol{D}_1^{(135)} & \boldsymbol{D}_2^{(135)} \end{pmatrix} \begin{pmatrix} S_0 \\ S_1 \\ S_2 \end{pmatrix} = \boldsymbol{A}\boldsymbol{S}. \tag{1.62}$$

La matriz $\boldsymbol{A}$ es la matriz de instrumentación del sistema. Si pseudoinvertimos la relación, podemos encontrar una relación entre los parámetros de Stokes y el vector de intensidades:

$$\boldsymbol{S} = \boldsymbol{A}^{+}\boldsymbol{I}. \tag{1.63}$$

Una forma de estimar la matriz $\boldsymbol{A}$ experimentalmente es utilizando un polarizador ideal colocado justo en frente del sensor. Girando el polarizador en un conjunto de ángulos de polarización distintos $\theta \in \{\theta_1, \theta_2, \ldots, \theta_n\}$, generamos n vectores de Stokes $\boldsymbol{S}^{(1)}$, $\boldsymbol{S}^{(2)}$ $\ldots$ $\boldsymbol{S}^{(n)}$, donde:

$$\boldsymbol{S}^{(i)} = \begin{pmatrix} 1 \\ \cos(\theta_i) \\ \sin(\theta_i) \end{pmatrix}. \tag{1.64}$$

Podemos medir con el sensor los n observables $\boldsymbol{I}^{(1)}$, $\boldsymbol{I}^{(2)}$ $\ldots$ $\boldsymbol{I}^{(n)}$ e inferir la matriz de instrumentación a través de la pseudoinversa:

$$\boldsymbol{A} = \begin{pmatrix} I_0^{(1)} & I_0^{(2)} & \cdots & I_0^{(n)} \\ I_{45}^{(1)} & I_{45}^{(2)} & \cdots & I_{45}^{(n)} \\ I_{90}^{(1)} & I_{90}^{2} & \cdots & I_{90}^{(n)} \\ I_{135}^{(1)} & I_{135}^{2} & \cdots & I_{35}^{(n)} \end{pmatrix} \begin{pmatrix} S_0^{(1)} & S_0^{(2)} & \cdots & S_0^{(n)} \\ S_1^{(1)} & S_1^{(2)} & \cdots & S_1^{(n)} \\ S_2^{(1)} & S_2^{(2)} & \cdots & S_2^{(n)} \end{pmatrix}^{+}. \tag{1.65}$$

Capítulo 2

Desarrollo del Microscopio: *Hardware*

Diseñamos un sistema óptico-mecánico capaz de obtener características polarimétricas de una muestra completa utilizando una cámara polarizada, un polarizador y una platina. Tanto el polarizador como la platina están motorizados para moverse según las coordenadas $\{X, Y, \theta\}$ y son controlados por una Raspberry Pi. Las coordenadas X e Y son espaciales y miden la posición de la muestra tomada. La coordenada θ es angular y mide el ángulo de polarización de la luz incidente. La Raspberry Pi y la cámara son controladas por la computadora (PC) y se coordinan para mover los motores y capturar las imágenes. Ver Figura 2.1.

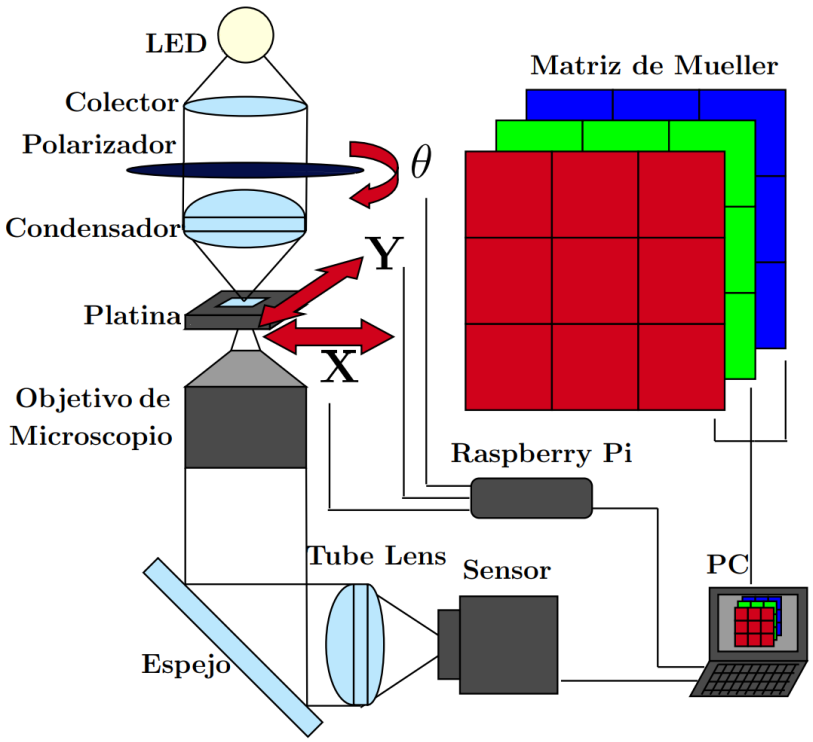

Figura 2.1: Esquema del microscopio. Realizado en Mathcha.

2.1. Diseño e Impresión

La impresión 3D se ha vuelto popular en los últimos años por su versatilidad, calidad y bajo costo. Consiste en la impresión aditiva de un modelo digital utilizando plástico fundido, principalmente ácido poliláctico (PLA). A continuación presentaremos las impresiones de cada parte del sistema y describiremos cómo funcionan. La mayoría de las piezas se imprimieron en una impresora 3D Ultimaker 3. Los programas utilizados para el diseño de las piezas fueron Blender y Fusion360.

2.1.1. Iluminador

Consiste de una fuente LED blanca con brillo regulable, una película polarizadora y dos lentes: Colector (CO) y Condensador (CL). Como CO se eligió una lente convencional de 40 mm de distancia focal y lo colocamos de forma tal que el LED quede en el punto focal de la lente. Esta lente tiene el rol de capturar la mayor cantidad de luz posible del LED y transmitirlos paralelos al eje óptico (colimados). Como CL elegimos un doblete acromático de 25 mm de distancia focal, el cual concentra la luz sobre la muestra mejor que una lente convencional. Se coloca justo después del polarizador, a cualquier distancia del mismo. Ver Figura 2.2.

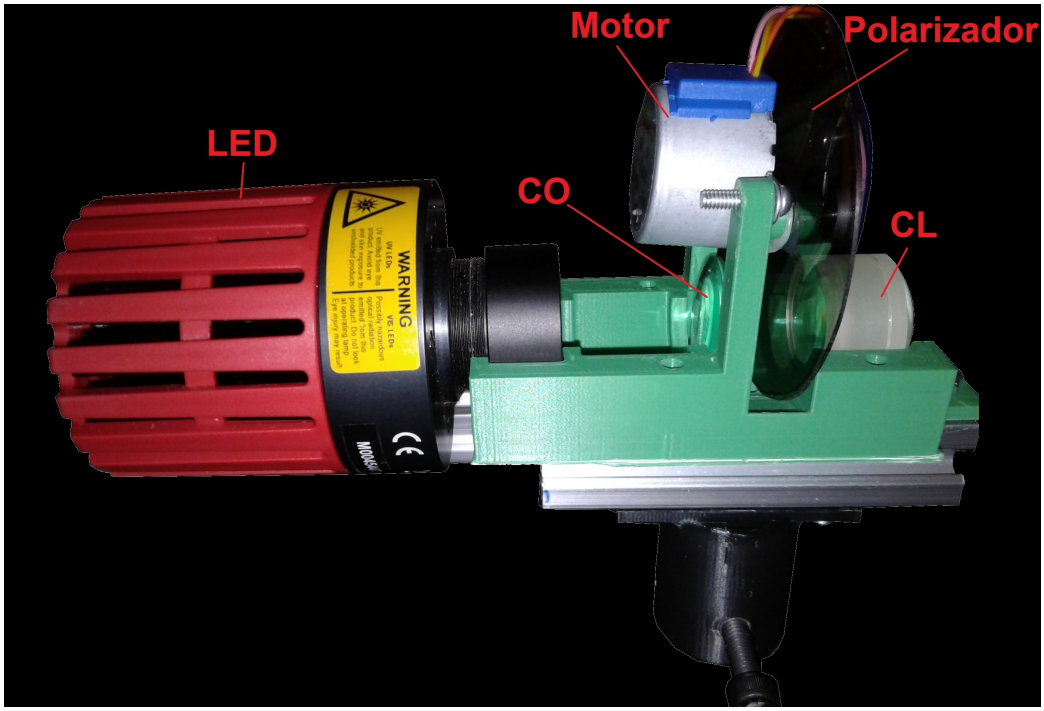

Figura 2.2: Iluminador con polarizador motorizado.

El polarizador debe girar automáticamente para generar distintos estados de polarización e iluminar al tejido. Se atornilla en el iluminador un motor de paso modelo 28BYJ-48. Al motor le unimos un polarizador de plástico circular de 6 cm

de diámetro centrado en el eje. El iluminador entero se atornilla en un trozo de perfil metálico de aluminio de $20 \times 20 \times 80$ (mm).

El perfil se atornilla a un soporte de plástico negro que lo ayuda a afirmarse a un par de postes metálicos unidos por una abrazadera de 90°. El sistema de iluminación se afirma con una base magnética a la mesa metálica, una vez posicionado. Para verificar su funcionamiento, colocamos un trozo de papel y corroboramos la formación de una mancha de luz encima del Objetivo de Microscopio (MO).

2.1.2. Porta espejo y Portaobjetivo

Consiste de una caja hueca de 5 cm de ancho y 2 mm de espesor. En su interior tiene un escalón a 45° para soportar un espejo y un soporte para una lente conocida como *Tube Lens* (TL). El espejo está diseñado para no cambiar el (SoP) de la luz. Como TL elegimos un doblete acromático de 50 mm de diámetro y 200 mm de distancia focal. También cuenta con cuatro agujeros que se atornillan al portaobjetivo a través de cuatro tornillos $M3$. Finalmente, se afirma la caja a la mesa metálica con dos tornillos $M5$. Ver Figura 2.3.

Además, armamos una pieza cilíndrica en donde se enrosca un objetivo de microscopio (MO) de la marca Olympus UPLFLN $\times 10$. Se imprimió en dos partes y adentro colocamos un anillo de cobre diseñado en el Taller de Mecánica del Instituto de Física (IFFI). El hilo de montaje del Objetivo tiene un paso del tipo 36 TPI (*Thread Per Inch*) y un diámetro de 20,32 mm. Ver Figura 2.4.

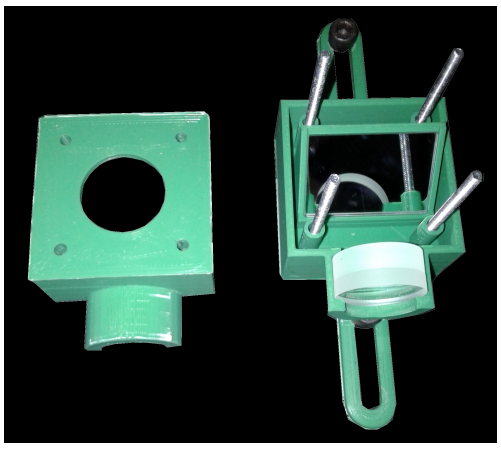

Figura 2.3: Porta espejo.

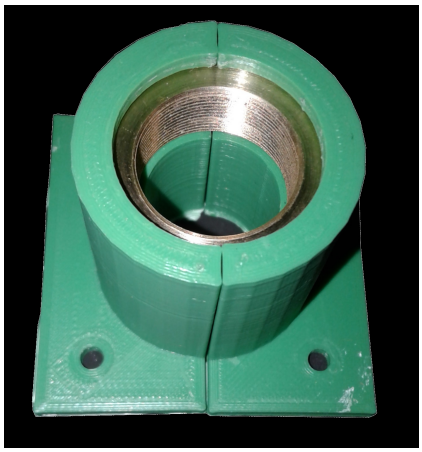

Figura 2.4: Portaobjetivo.

2.1.3. Trípode para Cámara

Mantiene la cámara polarizada a una altura determinada. Utiliza tres tornillos $M3$ para afirmarse a la cámara y un tornillo $M5$ para afirmarse a la mesa metálica. A la cámara le instalamos un tubo de extensión con montura de tipo C. El mismo se conecta finalmente al resto del sistema, previniendo la interferencia de luz externa. Ver Figura 2.5.

2.1.4. Portamuestras

Soporta el portaobjetos (lámina de vidrio en donde va la muestra), diseñado para aquellos que miden $27 \times 75 \times 1$ (mm). Cuenta con cuatro agujeros de 4 mm de diámetro en donde pasan dos varillas metálicas que sirven como guías. Además, tiene un agujero de $8 \times 4 \times 12$ (mm) para colocar una tuerca que convierte el movimiento circular del motor Y en movimiento lineal. Para conectar el motor Y con el portamuestras utilizamos un acople de $5 \times 5 \times 25$ (mm) y una barra hilada de 5 mm de diámetro, 35 mm de largo y 1 mm de *pitch*. Ver Figura 2.6.

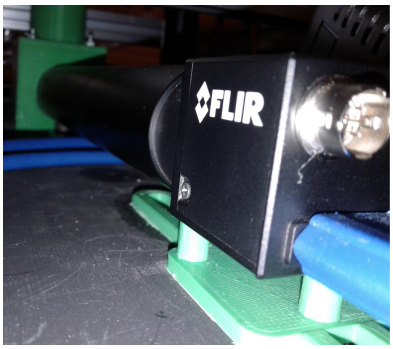

Figura 2.5: Porta cámara.

Figura 2.6: Portamuestras.

2.1.5. Carcasas Nema 17

Posicionan los motores X e Y, cuyo diseño original fue descargado en thingiverse.com [16] y acondicionado en Blender. El motor X se vincula con el motor Y a través de un acople impreso en plástico con una barra hilada en su interior de 6 mm de diámetro, 80 mm de largo y 1 mm de *pitch*. Ver Figura 2.7.

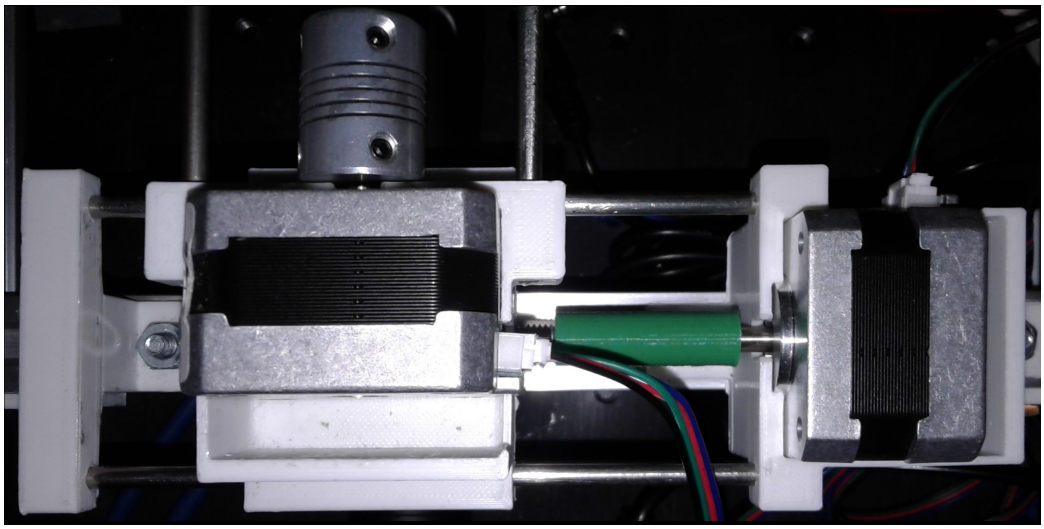

Figura 2.7: Motores y sus carcasas.

2.1.6. Carcasa Raspberry-Drivers

Utilizamos diseños preexistentes disponibles gratuitamente en internet, modificándolos en Blender para que se ajusten a nuestros modelos y necesidades específicas:

- Carcasa Raspberry: Modelo de printables.com [17].
- Carcasa Doble Driver L298N: Modelo de printables.com [18].
- Carcasa Driver ULN2003: Modelo de thingiverse.com [19].

Las modificaciones realizadas fueron ajustes de tamaño basándose en las mediciones hechas con calibre de cada elemento. Además, añadimos enganches o soportes para realizar el montaje entre las tres piezas. Ver Figuras 2.8 y 2.9.

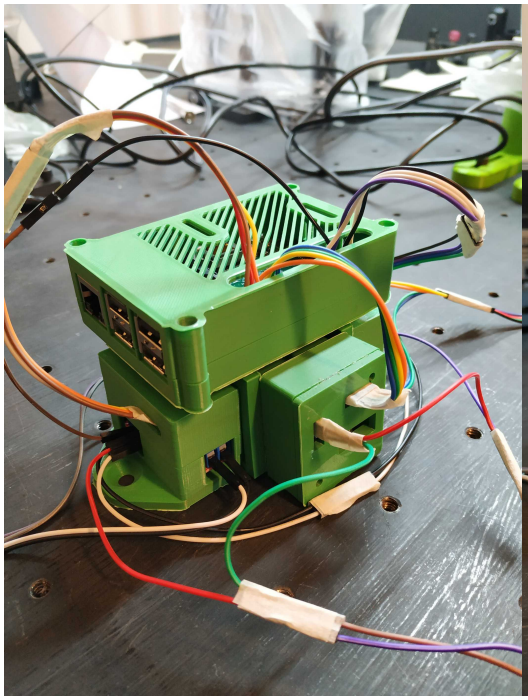

Figura 2.8: Perspectiva Isométrica.

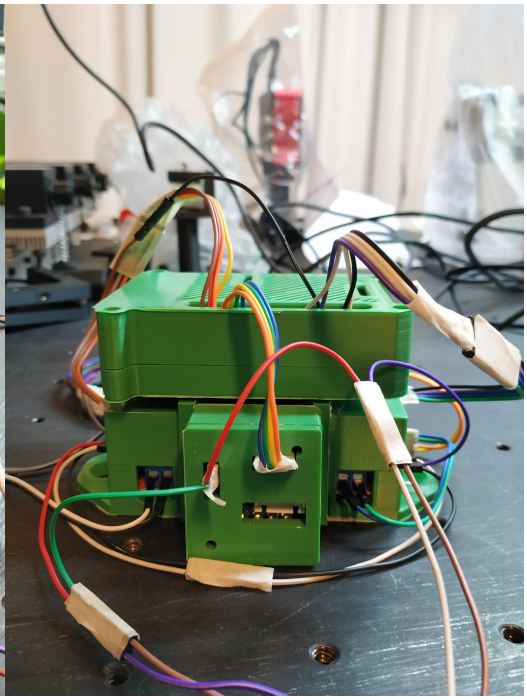

Figura 2.9: Perspectiva Frontal.

2.1.7. Montaje Completo del *Setup*

Renderizar es el proceso que convierte una escena digital tridimensional en una imagen bidimensional. En Blender [20], existen distintas técnicas de renderizado (Eevee, Cycles, etc.) basados en fenómenos físicos, principalmente de origen óptico. El resultado final depende de las luces, el tipo de materiales y la cámara. Para terminar esta sección, presentamos el renderizado que realizamos en la Figura 2.10 y una foto del sistema completo armado en la Figura 2.11.

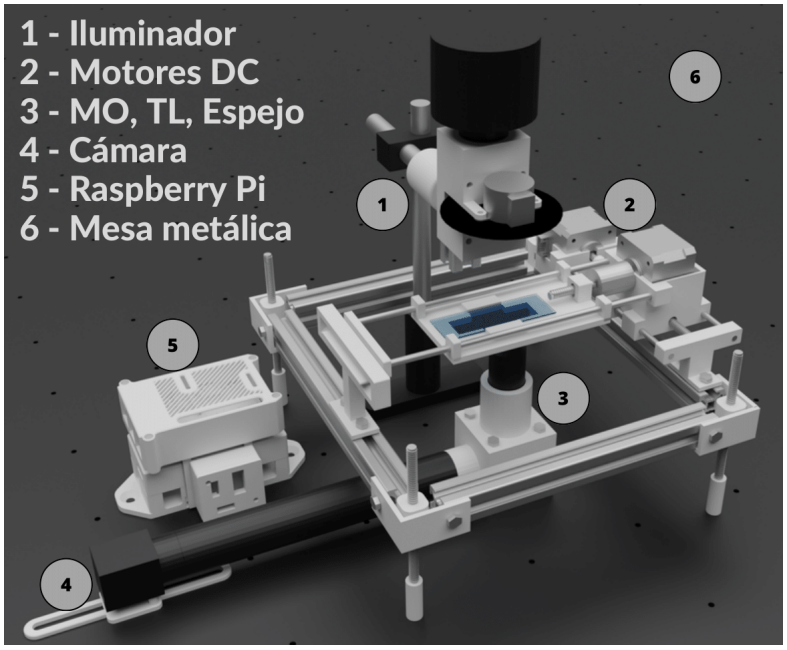

Figura 2.10: Renderizado del sistema [21][22][23].

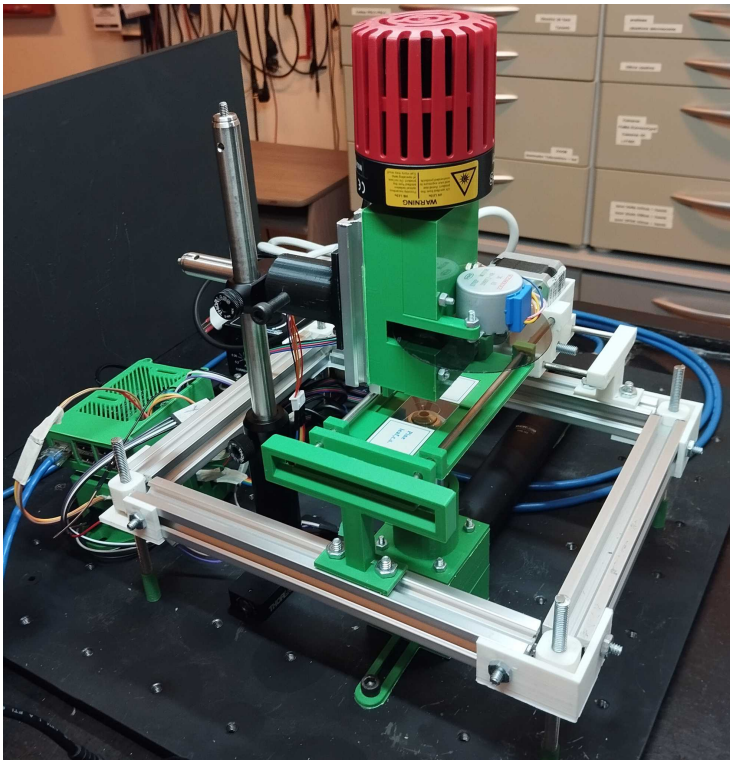

Figura 2.11: *Setup* del sistema.

2.2. Raspberry Pi

Raspberry Pi es un tipo de computadora de placa simple (SBC) que se ha vuelto muy popular dado su versatilidad para integrar diversos componentes electrónicos y desarrollar una amplia variedad de tareas. Su característica distintiva son las filas de pines de entrada/salida de propósito general (GPIO) situadas a un lado de la placa, permitiendo una interfaz física entre la placa y el mundo exterior.

Para este trabajo, elegimos una Raspberry Pi 2 porque disponíamos de una en el laboratorio. La misma está equipada con una CPU ARM Cortex-A7 de 900 MHz, una memoria RAM de 1 GB y una GPU VideoCore IV, proporcionando las especificaciones adecuadas para controlar los tres motores.

2.2.1. Sistema Operativo (OS)

Para utilizar la Raspberry Pi se requiere proveerle una unidad de memoria con un sistema operativo (booteable). Decidimos optar por una Micro SD Kingston de 8 GB en la que instalamos Raspberry Pi OS (anteriormente llamado Raspbian), una distribución de GNU/Linux basado en Debian para Raspberry Pi. Para esto, utilizamos la aplicación 'Raspberry Pi Imager', encargada de realizar el formateo, instalación y configuración inicial del sistema operativo.

Como vemos en la Figura 2.12, utilizamos un nombre de usuario acorde al título del trabajo, agregamos una contraseña y habilitamos el SSH con autenticación por contraseña para poder comunicarnos con la Raspberry Pi en una red local.

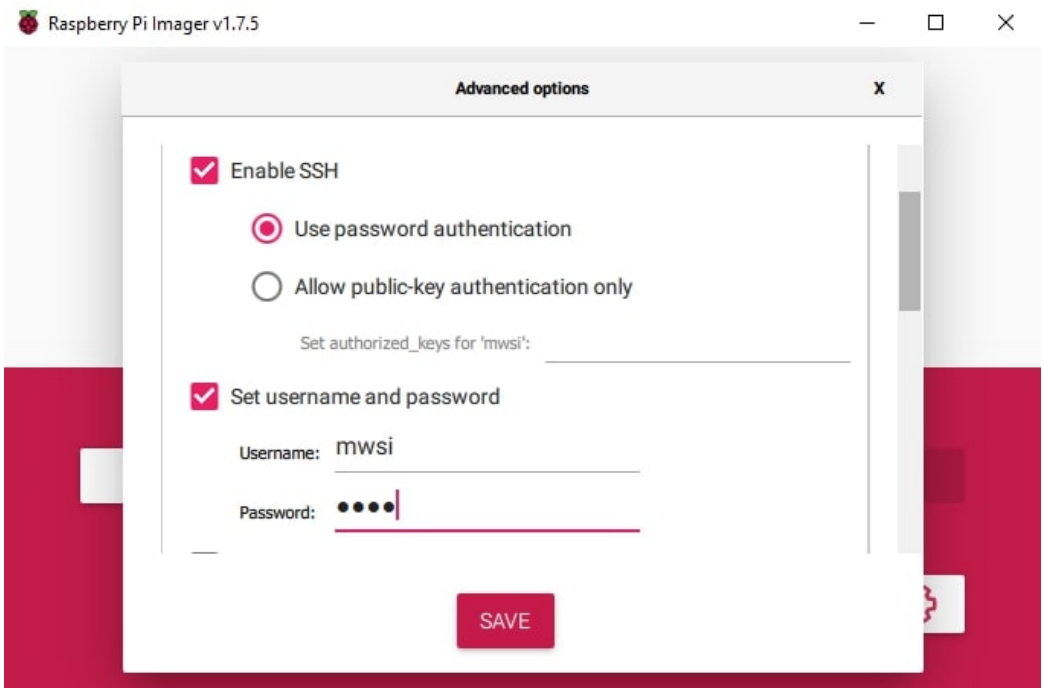

Figura 2.12: Instalación Raspbian.

2.2.2. Esquema de Conexiones: Motores, *Drivers* y Placa

El desplazamiento del portamuestras en el sistema de microscopía es plano, por lo que utilizamos un par de motores Stepper Nema 17 para controlarlo con suficiente precisión. Estos motores son bipolares, trabajan a 5 V, consumen 1,2 A y presentan un ángulo de paso de 1,8°. Esto significa que pueden dividir cada revolución en 200 pasos, y que, usando un tornillo con un *pitch* de 1 mm, podemos mover en cada paso $\frac{1mm}{360°} = 6\ \mu m$, distancia adecuada para recorrer un tejido microscópico. Para alimentarlos, tuvimos en cuenta que cada motor se acciona por vez y, por lo tanto, alcanza con el transformador elegido de 5 V y 3 A.

Para la rotación del polarizador utilizamos un motor 28BYJ-48, de menor calidad, ya que solo se requieren seis posiciones (correspondientes a 0°, 30°, 60°, 90° y 120°). Este motor es unipolar, tiene 32 pasos de tipo *full-step* por vuelta, funciona a 5 V y consume 55 mA. Además, cuenta con una reductora integrada de 64 : 1, por lo que divide cada paso en 64 más pequeños para mayor precisión. Por lo tanto, llega a los 2048 pasos *full-step* por vuelta de 0,7° cada uno. Esto nos deja con un motor con suficiente precisión para el trabajo, y que además puede ser alimentado directamente desde la Raspberry Pi por su bajo consumo. La elección tanto de los motores como de los tornillos usados puede cambiarse si se requieren movimientos de otras características (más o menos precisos, menor consumo, etc.)

Presentamos un diagrama que esclarece la conexión de estos elementos con la Raspberry Pi. Ver Figura 2.13. La placa se alimenta con un transformador USB de 5 V y 3 A. Los dos drivers L289N están alimentados por una fuente de 5 V en sus respectivos pines de alimentación (pin +12 V). Los pines de 5 V corresponden a la alimentación de la parte lógica del circuito del driver y están siendo alimentados mediante la entrada de 12 V por el *Jumper*, incluido en el driver con este fin.

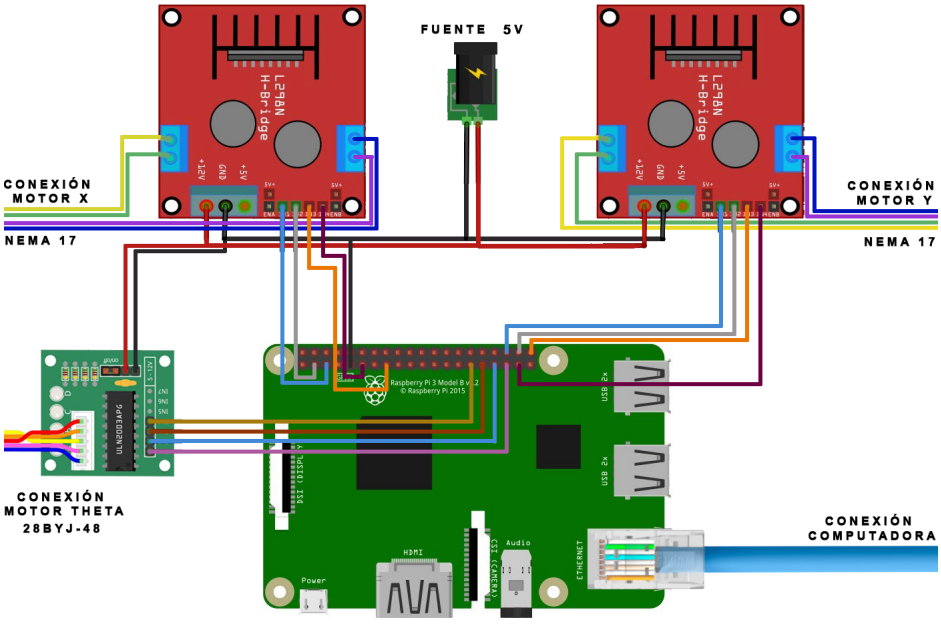

Figura 2.13: Esquema de conexión de la Raspberry Pi.

2.3. Computadora

Para la operación de la cámara y el *software* principal utilizamos una computadora. Actualmente, contamos con una notebook Gateway NV52L con una CPU AMD Radeon 7640G de 1,9 GHz, RAM de 8 GB y GPU de 512 MB integrados.

Para maximizar estos recursos, decidimos utilizar un sistema operativo Unix, específicamente Ubuntu 22.04. Esta computadora se conecta mediante Ethernet (cat5) a la Raspberry con el fin de realizar y coordinar la ejecución, mediante comandos enviados por red, del código que controla los motores conectados a la Raspberry. Esta separación de órdenes responde a la disponibilidad de recursos, la Raspberry no cuenta con suficiente poder computacional para llevar a cabo todo el proceso por su cuenta.

2.3.1. Cámara Polarizada

El dispositivo para capturar imágenes es una cámara a color polarizada, modelo BFS-U3-51S5PC-C. Consta de un *chip* Sony IMX250MYR conformado por una grilla rectangular de 2048×2448 píxeles cuadrados. Cada píxel tiene un ancho de 3,45 μm, una profundidad de color de 8 bits y sostiene encima un polarizador con un ángulo de polarización $\psi \in \{0°, 45°, 90°, 135°\}$, el cual define cada uno de los canales de polarización [24]. Esta cámara fue elegida porque cuenta con un sensor que trabaja en tres canales de color en el espectro visible y porque es uno de los pocos modelos disponibles en *Edmund Optics*.

La información de color se procesa a través del filtro de Bayer. Para obtener los canales de color en cada canal de polarización, es necesario un *de-mosaicing*. En este sentido podemos decir que la cámara tiene 12 canales, 4 por cada canal de polarización y 3 por cada color. En la Figura 2.14 podemos ver el diagrama de construcción del superpíxel y su respectivo filtrado de Bayer.

Para instalar la cámara en Ubuntu, primero descargamos el SDK correspondiente a nuestra versión de Ubuntu desde la página web de FLIR [24]. Luego, descomprimimos e instalamos el archivo .whl correspondiente al SDK:

```
tar -xvf archivo_sdk.tar.gz
pip install instalador_sdk.whl
```

A continuación, instalamos las librerías requeridas por la cámara:

```
apt-get install libavformat58 libswscale5
libavcodec58
```

Por último, proveemos a nuestro usuario con los permisos necesarios para operar la cámara, agregando al usuario de Ubuntu con el que trabajamos al grupo `flirimaging` que se crea automáticamente al instalar la cámara. Este grupo contiene por defecto los permisos para operar la cámara. Si no le agregamos usuarios, solo el superusuario podrá acceder a las cámaras.

```
usermod -a -G flirimaging nombre_usuario
```

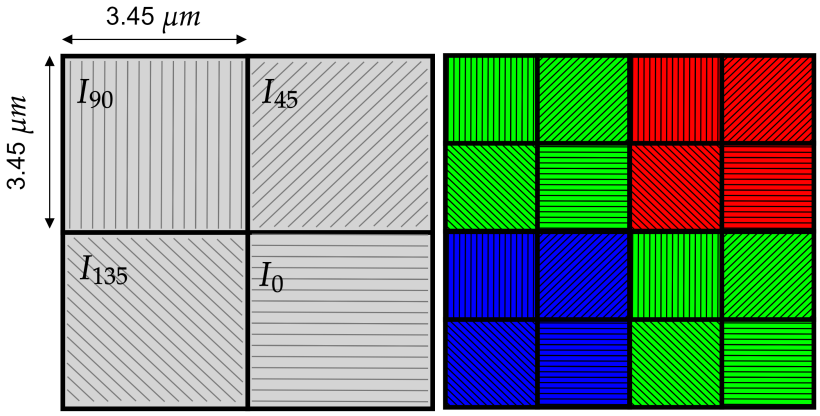

Figura 2.14: Diagrama de superpíxel (izquierda) y su respectivo filtrado de Bayer (derecha). Realizado en Mathcha.

2.3.2. Comunicación con Raspberry Pi

Los comandos a ejecutar en la Raspberry se envían mediante una conexión SSH vía Ethernet. Esto asegura una comunicación fluida y segura entre terminales. Para esto configuramos una red local cableada (aunque se podría usar wifi) que asegura la permanencia de la IP asignada a nuestra Raspberry. Esto se hace desde la herramienta de configuración de Ubuntu en la computadora, configurando las opciones de IPv4 con método "Solo enlace local". Hecho esto podemos buscar la IP de nuestra Raspberry y probar la conexión:

```
apt install net-tools openssh-server # packages
    necesarios
arp -a                               # obtener ip
    raspberry
ssh user_raspberry@ip_raspberry      # conectar a
    raspberry
```

En la Figura 2.15 vemos el resultado del protocolo de resolución de direcciones (ARP). En la última línea está la Raspberry Pi, su dirección IP y MAC.

```
root@mueller-NV52L:/home/mueller/Escritorio/MWSI_REPO# arp -a
? (172.16.134.221) en 3c:7a:aa:eb:cd:d4 [ether] en wlp2s0
_gateway (172.16.0.1) en 94:0c:6d:80:a6:ff [ether] en wlp2s0
DESKTOP-9KF1TDS.guest.fing.edu.uy (172.16.112.137) en 90:a4:de:40:92:8d [ether] en wlp2s0
DESKTOP-8FA79NT.guest.fing.edu.uy (172.16.105.131) en 10:63:c8:60:6e:6f [ether] en wlp2s0
DESKTOP-R0IT4U7.guest.fing.edu.uy (172.16.124.8) en 5c:87:9c:67:cb:86 [ether] en wlp2s0
LAPTOP-JGHGQF05.guest.fing.edu.uy (172.16.147.213) en 5c:3a:45:d9:3d:09 [ether] en wlp2s0
EPSONDCA5F9.guest.fing.edu.uy (172.16.112.60) en e0:bb:9e:dc:a5:f9 [ether] en wlp2s0
DESKTOP-MBKT62J.guest.fing.edu.uy (172.16.135.237) en 68:54:5a:d9:2d:58 [ether] en wlp2s0
salon102.guest.fing.edu.uy (172.16.127.79) en 18:c0:4d:fe:51:47 [ether] en wlp2s0
LAPTOP-MI2CNPT4.guest.fing.edu.uy (172.16.150.183) en 3c:91:80:b2:56:ef [ether] en wlp2s0
LAPTOP-TNIHV9C3.guest.fing.edu.uy (172.16.101.86) en cc:2f:71:3d:be:0e [ether] en wlp2s0
EPSON06EE40.guest.fing.edu.uy (172.16.100.211) en a4:d7:3c:06:ee:40 [ether] en wlp2s0
LYOS-323850.guest.fing.edu.uy (172.16.120.185) en 14:eb:b6:cd:d7:cc [ether] en wlp2s0
DESKTOP-FH1T73J.guest.fing.edu.uy (172.16.112.9) en d0:df:9a:cf:52:8d [ether] en wlp2s0
LAPTOP-83MUHLF3.guest.fing.edu.uy (172.16.136.31) en dc:fb:48:14:5e:fb [ether] en wlp2s0
DESKTOP-1KFHJOJ.guest.fing.edu.uy (172.16.137.2) en 40:23:43:5b:cf:3f [ether] en wlp2s0
DESKTOP-MC77C2C.guest.fing.edu.uy (172.16.148.78) en 80:2b:f9:ca:90:8b [ether] en wlp2s0
LAPTOP-18FGHHAJ.guest.fing.edu.uy (172.16.119.63) en 90:0f:0c:36:de:dd [ether] en wlp2s0
DESKTOP-3B0GTH7.guest.fing.edu.uy (172.16.118.202) en 50:c2:e8:ac:e6:e3 [ether] en wlp2s0
DESKTOP-HO7JH8R.guest.fing.edu.uy (172.16.117.1) en 94:e7:0b:d9:3a:c3 [ether] en wlp2s0
raspberrypl.local (169.254.110.82) en b8:27:eb:3d:d8:48 [ether] en enp1s0
root@mueller-NV52L:/home/mueller/Escritorio/MWSI_REPO#
```

Figura 2.15: Ejecución del comando `arp -a`.

En la Figura 2.16 vemos la configuración de red cableada, dejando remarcada la opción 'Solo enlace local' para definir que la red sea local.

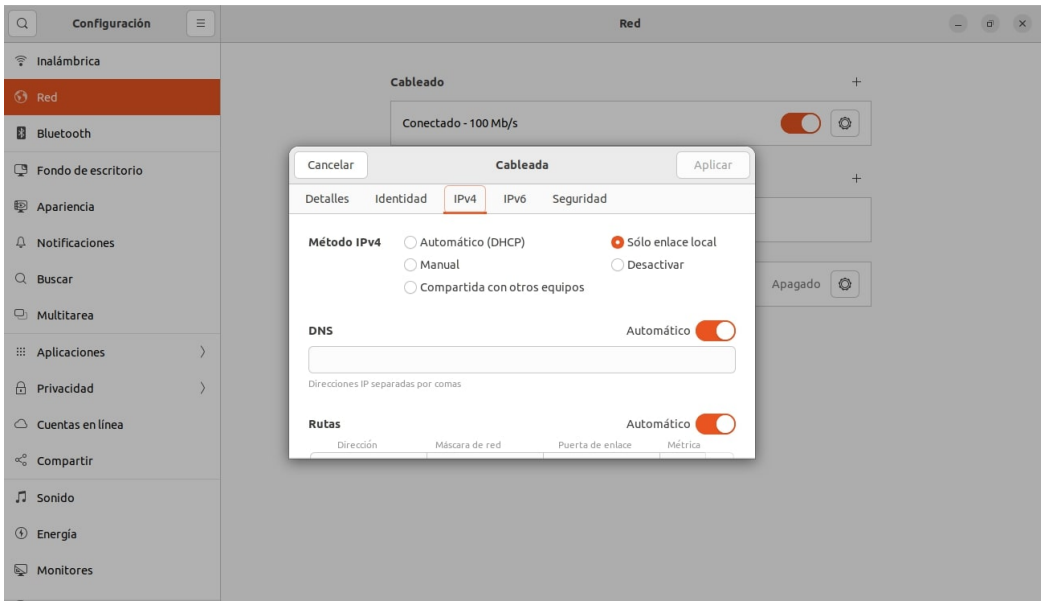

Figura 2.16: Configuración Red Local.

2.3.3. De-mosaicing de la Máscara Polarizada

Para restaurar la imagen de intensidad polarizada en cada canal de color, se requiere decodificar la información cruda proveniente de la cámara polarizada. Los voltajes capturados por cada fotodiodo del sensor se envían en una secuencia de 2448×2048 enteros sin signo de 8 bits que podemos agrupar en un arreglo unidimensional que denominaremos $\boldsymbol{A}$. El primer paso para la reconstrucción es mapear $\boldsymbol{A}$ sobre un arreglo bidimensional $\boldsymbol{B}$ con la misma cantidad de datos mediante una operación de remodelado o *reshape*. De esta forma recuperamos la correlación espacial de cada medida. El segundo paso es dividir $\boldsymbol{B}$ en cuatro arreglos $\boldsymbol{B}_0$, $\boldsymbol{B}_{45}$, $\boldsymbol{B}_{90}$ y $\boldsymbol{B}_{135}$ de 1224×1024 enteros sin signo de 8 bits, uno por cada ángulo de polarización. Este paso separa los canales de polarización, pero reduce la resolución final de las imágenes. Por último, para reconstruir la información cromática en cada canal de polarización, empleamos una técnica de interpolación. Para esta parte recurrimos a dos métodos muy conocidos: vecinos más cercanos y bilineal. Ver Figura 2.17.

Para aplicar el método de vecinos más cercanos, interpolamos el valor desconocido de cada píxel con el valor del píxel más cercano en un entorno de radio uno. En el caso de que haya varios píxeles dentro del entorno, nos quedamos con uno elegido al azar. Por otro lado, para aplicar interpolación bilineal, cada valor de píxel desconocido se interpola con el promedio de todos los valores de píxel dentro del entorno de radio uno. La principal diferencia de ambos métodos es la velocidad de cómputo y la calidad. Vecinos más cercanos es más rápido, pero la calidad de

la imagen es menor.

Implementamos la función `polarization_full_dec_array()` para decodificar la información recibida de la cámara. La función tiene como argumento la secuencia de símbolos $\boldsymbol{A}$ y el tipo de interpolación deseada entre dos posibles opciones: 'vecinos' y 'bilineal'. Retorna los cuatro canales de polarización I_0, I_{45}, I_{90} y I_{135}, como arreglos de $1224 \times 1024 \times 3$ enteros sin signo de 8 bits.

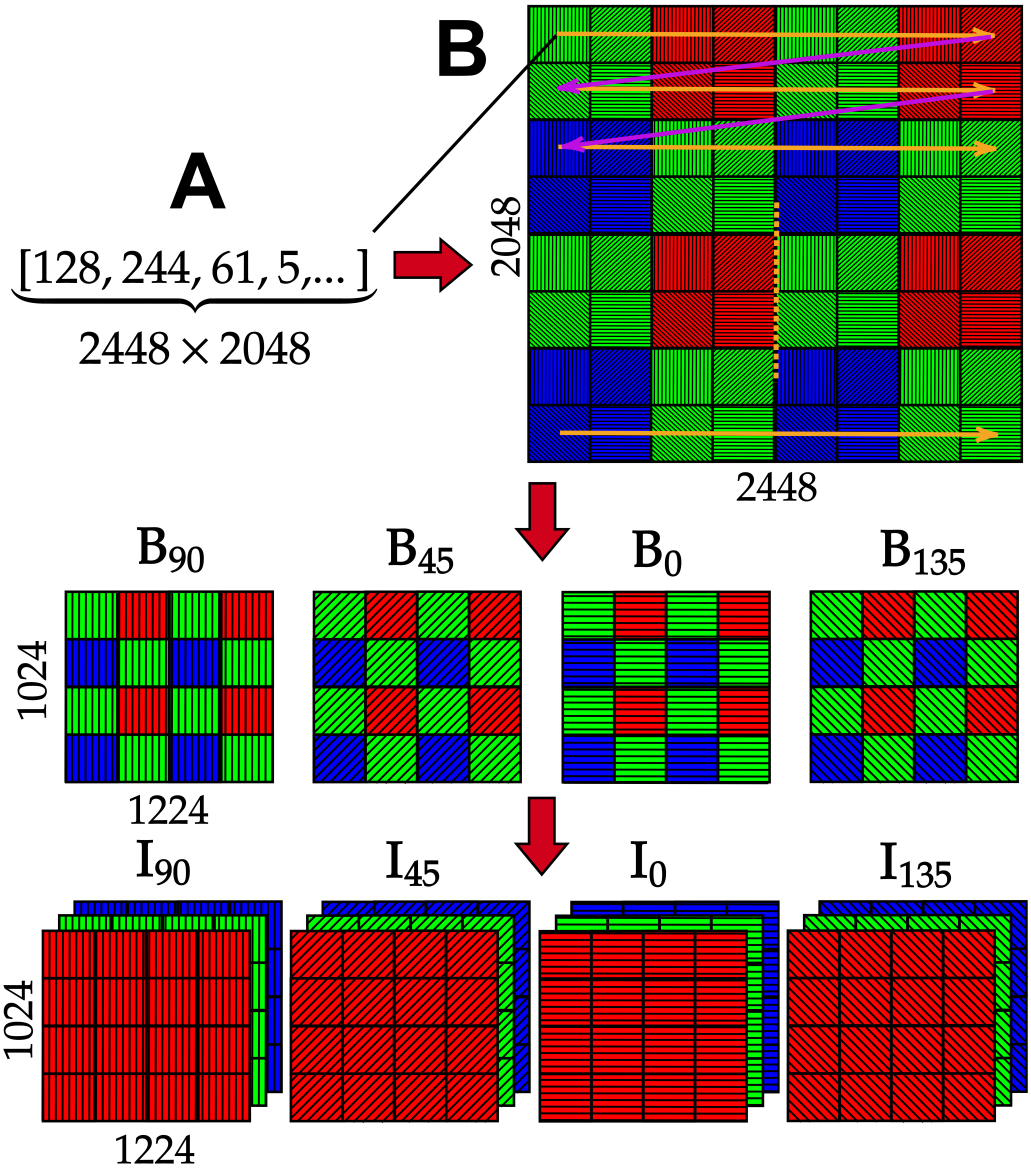

Figura 2.17: *De-mosaicing* de la máscara polarizada. Remodelamos el arreglo unidimensional $\boldsymbol{A}$ para convertirlo a un arreglo bidimensional $\boldsymbol{B}$, separamos por canal de polarización y finalmente interpolamos. Realizado en Mathcha.

Capítulo 3

Desarrollo del Microscopio: *Software*

Teniendo en cuenta la magnitud del trabajo, el hecho de que es grupal y la importancia de mantener un control de los programas, se decidió crear un repositorio en GitHub para administrar el desarrollo de código [25]. El mismo es accesible de forma pública, es código abierto y fue totalmente desarrollado en Python (ver funciones en Apéndice E). En este capítulo mostraremos los programas que controlan el microscopio, cómo lo calibramos, el escaneo de la muestra completa y la interfaz de usuario. En la Figura 3.1 vemos un esquema de cómo funciona el algoritmo para capturar matrices de Mueller de muestra completa.

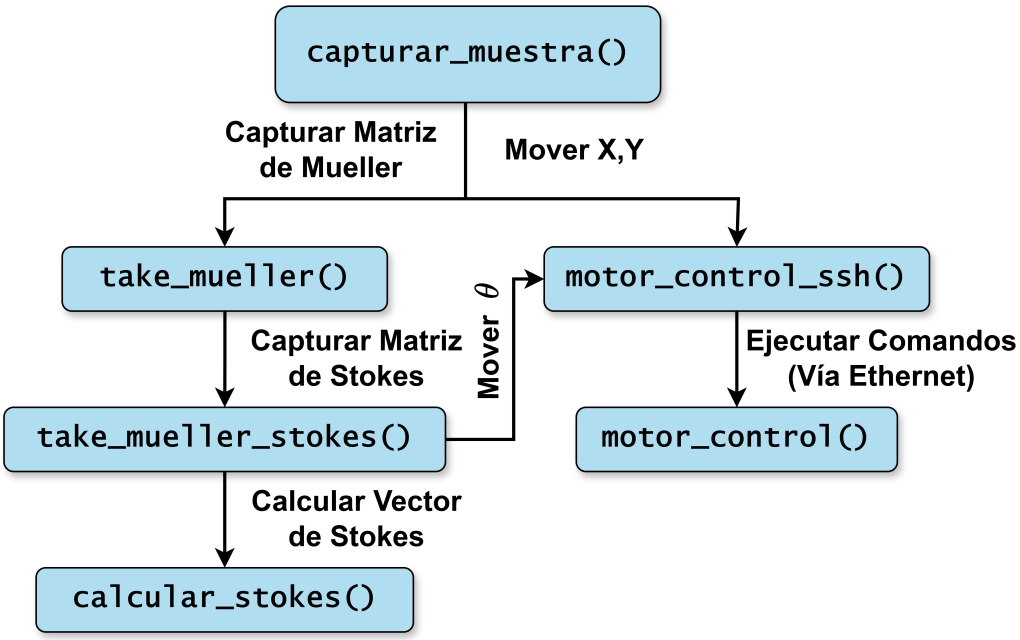

Figura 3.1: Flujo de ejecuciones en el *software* diseñado. Las flechas indican el orden de jerarquía y los comentarios la ejecución. Realizado en Drawio.

Diseñamos `capturar_muestra()` para fotografiar la muestra completa, la cual coordina el cálculo de Mueller a través de `take_mueller()` y el control de los moto-

res X, Y y θ a través de `motor_control_ssh()`. La función `motor_control_ssh()` envía los comandos desde la PC a la Raspberry Pi por medio de la conexión Ethernet, tomando como argumento de entrada qué motor mover y en qué dirección. Dentro de la Raspberry Pi, se encuentra otra función llamada `motor_control()`, la cual recibe los comandos de `motor_control_ssh()` y se encarga de ejecutarlos. La función `take_mueller()` ejecuta `take_mueller_stokes()`, cuya función es coordinar la captura de los vectores de Stokes con ayuda de `calcular_stokes()`, y el movimiento del polarizador motorizado por medio de `motor_control_ssh()`.

3.1. Calibración del Sistema

La calibración del sistema se puede dividir en dos partes. Por un lado, cuando se utiliza un sistema DoFP, la calibración de los parámetros de Stokes suele ser necesaria debido a la baja eficiencia de los micro-polarizadores, problema que puede ser resuelto utilizando la metodología demostrada en la Sección 1.6.2. Por otro lado, puesto que el polarizador del iluminador está motorizado, debemos asegurarnos que se mueva siempre lo mismo para evitar un corrimiento constante que se propague en nuestras medidas. Además, debemos asegurarnos de ajustar el potenciómetro del LED al principio de cada captura completa. El brillo de los píxeles debe ser lo más alto posible sin que ninguno de ellos sature en ningún momento.

3.1.1. Cálculo de Parámetros de Stokes

El Sistema ilumina la muestra con luz polarizada a través de un polarizador lineal con ángulo de polarización θ. Los observables de la cámara polarizada $\boldsymbol{I} = (I_0, I_{45}, I_{90}, I_{135})^T$ dependen no solo de la posición del píxel $\boldsymbol{p} = (x, y)$ y la longitud de onda central λ, sino también del ángulo θ. La forma adecuada de obtener los parámetros de Stokes $\boldsymbol{S} = (S_0, S_1, S_2)^T$ es a través de la matriz de instrumentación del sistema $\boldsymbol{A}$ (1.62):

$$\boldsymbol{I}(\boldsymbol{p}, \lambda, \theta) = \boldsymbol{A}(\boldsymbol{p}, \lambda)\boldsymbol{S}(\boldsymbol{p}, \lambda, \theta). \tag{3.1}$$

Puesto que nuestra cámara es moderna y cuenta con polarizadores con muy baja tasa de extinción, en este caso particular la calibración no es necesaria [6]. Entonces, podemos suponer que la matriz de instrumentación es aproximadamente igual a la matriz ideal (1.43):

$$\boldsymbol{A}(\boldsymbol{p}, \lambda) \simeq \frac{1}{2}\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix}. \tag{3.2}$$

Por lo tanto, podemos implementar una función que llamaremos `calcular_stokes()` para calcular los parámetros de Stokes de la siguiente forma (1.45):

$$\begin{aligned} S_0(\boldsymbol{p}, \lambda, \theta) &= 0{,}5[I_0(\boldsymbol{p}, \lambda, \theta) + I_{45}(\boldsymbol{p}, \lambda, \theta) + I_{90}(\boldsymbol{p}, \lambda, \theta) + I_{135}(\boldsymbol{p}, \lambda, \theta)], \\ S_1(\boldsymbol{p}, \lambda, \theta) &= I_0(\boldsymbol{p}, \lambda, \theta) - I_{90}(\boldsymbol{p}, \lambda, \theta), \\ S_2(\boldsymbol{p}, \lambda, \theta) &= I_{45}(\boldsymbol{p}, \lambda, \theta) - I_{135}(\boldsymbol{p}, \lambda, \theta). \end{aligned} \tag{3.3}$$

El programa `calcular_stokes()` mide los parámetros de Stokes con una representación de 16 bits con signo, utilizando como interpolador por defecto: vecinos más cercanos. Para capturar los datos crudos de la cámara utilizamos la librería 'Simple_PySpin' [26]. Si el arreglo $\boldsymbol{B}$ contiene los datos crudos entregados por la cámara polarizada, entonces la forma de proceder es la siguiente:

```
from stokeslib import polarization_full_dec_array, calcular_stokes

#Obtener medibles
I0, I45, I90, I135 = polarization_full_dec_array(B, 'vecinos')

#Calcular Parametros de Stokes
S0, S1, S2 = calcular_stokes(I90, I45, I135, I0)
```

Para asegurarnos que los medibles I_0, I_{45}, I_{90}, I_{135} tengan una representación digital fiel a su magnitud física, es muy importante asegurarse que ningún píxel de las imágenes sature en 255. Para esto se puede ajustar el brillo del LED girando el potenciómetro a mano, dejando el brillo de cada píxel lo más alto posible. Otra posible solución podría ser ajustando el tiempo de exposición de la cámara.

3.1.2. Ajuste del Iluminador Polarizado

Recordemos que el motor elegido, modelo 28BYJ-48, tiene 4 fases, un total de 2048 pasos *full-step* y una reductora de $64 \pm 1 : 1$. Nuestro algoritmo requiere completar una secuencia de 4 pasos *full-step* 512 veces para recorrer una vuelta entera. Para que el motor gire 30°, la cantidad de secuencias que debe completar es de $512/12 \simeq 41$. Como hicimos varias aproximaciones en el cálculo, en 41 pasos el motor no se moverá exactamente un ángulo de 30°. En principio no representa un problema, dado que los vectores de Stokes de entrada tienen AoP bien distribuido en los 180° y, por lo tanto, $\boldsymbol{\mathcal{S}}_{in}$ está bien condicionada. Como el error de precisión en la rotación del motor es de aproximadamente $\delta = 4 \times \frac{360°}{2048} \simeq 1°$, cada vez que ejecutemos el programa el polarizador se moverá un ángulo de $\Delta\theta = 30° \pm 1°$. La matriz de Stokes de entrada $\boldsymbol{\mathcal{S}}_{in}$ que elegimos tiene idealmente la siguiente estructura, en cada canal de color (ver Ecuación (1.31) para más información):

$$\boldsymbol{\mathcal{S}}_{in} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ \cos(0°) & \cos(30°) & \cos(60°) & \cos(90°) & \cos(120°) & \cos(150°) \\ \sin(0°) & \sin(30°) & \sin(60°) & \sin(90°) & \sin(120°) & \sin(150°) \end{pmatrix}. \quad (3.4)$$

Una medida indirecta del ángulo de rotación del polarizador es el AoP de la luz que transmite. En la Tabla 3.1 podemos apreciar el AoP de los seis vectores de Stokes, en cada canal de color. Observamos que los ángulos de polarización medidos no coinciden con los elegidos. Esto no es un problema, puesto que solo representa una rotación del sistema de coordenadas que no afecta las medidas más importantes del cálculo de Mueller. Lo importante es verificar que la diferencia entre ángulos $\Delta\theta$ sea $30° \pm 1°$ como habíamos calculado, asegurando así que el motor siempre se mueva igual.

Entrada	AoP		
-	R	G	B
1	37°	37°	37°
2	67°	67°	67°
3	95°	95°	95°
4	123°	123°	123°
5	153°	153°	153°
6	2°	3°	2°

Tabla 3.1: Ángulo de Polarización por canal.

Para la coordinación motor-cámara y capturar cada matriz de Stokes se diseña `take_mueller_stokes()` (ver Algoritmo 1). La matriz de Stokes de entrada se guarda invertida y comprimida en la computadora para facilitar los cálculos. La matriz de Stokes de salida se usa directamente para computar la matriz de Mueller y se debe volver a medir cada vez que se cambie de muestra.

Algoritmo 1 take_mueller_stokes(cam_config, theta_list)

```
thetas_list = [0, 30, 60, 90, 120, 150]  ▷ Comienza siempre en 0
for i, θ in enumerate(theta_list) do
    Guardar Vector de Stokes bajo polarización de ángulo θ
    if i ≠ 0 then
        Δθ = θ[i] − θ[i − 1]
        Girar polarizador un ángulo Δθ
    end if
end for
Girar polarizador un ángulo theta_list[i] en sentido opuesto
```

Para asegurarnos que el polarizador vuelva exactamente a su posición inicial, podemos calcular la matriz de Mueller del aire (sin muestra) y compararla con la matriz identidad. Este resultado es particularmente importante porque nos asegura que el cálculo que hagamos en las distintas secciones de tejido utilizan la misma metodología. Los resultados del experimento serán presentados en la Sección 3.3.1.

3.2. Estimación de Magnificación Efectiva

Una medida importante es la magnificación efectiva del sistema. La misma nos ayudará para obtener información sobre el tamaño de cada píxel de nuestra imagen. Una forma de medirla es a través de un target de calibración. El mismo cuenta con varios conjuntos de barras enumerados, como se puede observar en la Figura 3.2.

Una forma robusta de estimar la magnificación es midiendo la cantidad de píxeles N que hay de extremo a extremo en algún conjunto de barras (ver línea

roja en la Figura 3.3). El problema se puede resolver automáticamente utilizando herramientas de procesamiento de imágenes [27]. Según nuestros cálculos:

$$N = 275. \tag{3.5}$$

La distancia imagen y entre ambos extremos se calcula en función del número de píxeles N y el ancho de cada superpíxel:

$$y = N \times 2 \times 3{,}45\ \mu m = 1898\ \mu m. \tag{3.6}$$

La distancia objeto x se calcula como el cociente entre la cantidad de líneas blancas sobre la barra roja de la Figura 3.3 y el número de líneas por milímetro que indica el mismo patrón:

$$x = \frac{4{,}5\ Lp}{26\ Lp/mm} = 173\ \mu m. \tag{3.7}$$

Finalmente, la magnificación efectiva se define como el cociente entre la distancia imagen y la distancia objeto:

$$M_{eff} = \frac{y}{x} \simeq 11. \tag{3.8}$$

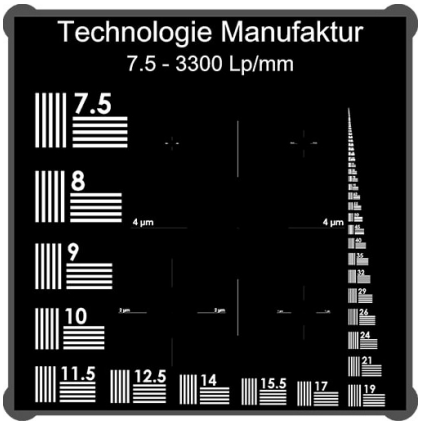

Figura 3.2: Target de calibración [28].

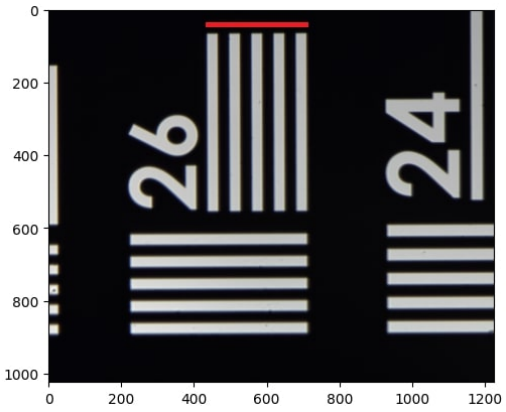

Figura 3.3: Zona del target elegida.

3.3. Cálculo y Representación de Matrices de Mueller

Para calcular la matriz de Mueller de la muestra necesitamos conocer cómo responde la misma bajo la incidencia de al menos tres vectores de Stokes. Si denominamos $\boldsymbol{\mathcal{S}}(\boldsymbol{p}, \lambda)$ a la matriz de Stokes evaluada en el píxel $\boldsymbol{p}$ y longitud de onda central λ, la matriz de Mueller se calcula (1.50):

$$\boldsymbol{M}(\boldsymbol{p}, \lambda) = \boldsymbol{\mathcal{S}_{out}}(\boldsymbol{p}, \lambda)\boldsymbol{\mathcal{S}_{in}}^{+}(\boldsymbol{p}, \lambda). \tag{3.9}$$

La matriz $\boldsymbol{\mathcal{S}_{in}}^{+}$ se captura sin la muestra en la platina, se calcula en formato de punto flotante de 64 bits y se guarda comprimida. Esto agiliza mucho el problema, puesto que $\boldsymbol{\mathcal{S}_{in}}$ es común para todos los cálculos de Mueller. Por cada vez que movemos el sistema motorizado o cambiamos de muestra, lo único que debemos hacer es volver a capturar $\boldsymbol{\mathcal{S}_{out}}$ y calcular la matriz de Mueller $\boldsymbol{M}$. El procedimiento que implementamos se denomina `take_mueller()` y es el siguiente:

```
from camaralib import take_mueller_stokes

#Toma Matriz de Stokes de entrada
S_in_stat = take_mueller_stokes(cam_config, thetas_list)

#Invierte Matriz de Stokes de entrada
S_in_stat_inv = np.linalg.pinv(S_in_stat)

#Se coloca la Muestra en la platina

#Toma Matriz de Stokes de salida
S_out_stat = take_mueller_stokes(cam_config, thetas_list)

#Calcula Matriz de Mueller
M = np.einsum('ijklm,ijkmn->ijkln', S_out_stat, S_in_stat_inv)
```

Para guardar apropiadamente la información, debemos normalizar la matriz de Mueller dividiendo cada una de sus componentes por m_{00}. De esta forma nos aseguramos que la información física no dependa de la transmitancia y esté siempre representada con valores dentro del intervalo $[-1, 1]$:

$$\hat{\boldsymbol{M}}(\boldsymbol{p}, \lambda) = \frac{1}{m_{00}(\boldsymbol{p}, \lambda)} \boldsymbol{M}(\boldsymbol{p}, \lambda). \tag{3.10}$$

Para almacenarla en la computadora utilizando la menor cantidad de memoria posible, optamos por digitalizar la información física con enteros sin signo de 16 bits. La forma indicada de hacerlo es a través del mapeo $f : [-1, 1] \to [0, 2^{16} - 1]$:

$$f(x) = \frac{2^{16} - 1}{2}(x + 1). \tag{3.11}$$

Por último, en cada canal de color debemos acoplar las 9 componentes de 1024×1224 en una sola de 3072×3672. La matriz $\hat{\boldsymbol{M}}$ y la componente m_{00} son finalmente almacenadas como imágenes, utilizando el formato de compresión sin pérdidas PNG. Nuestro sistema de imaginería es capaz de calcular, en cada superpíxel del sensor, la respectiva matriz de Mueller. Para representar gráficamente la matriz de Mueller sobre un campo de visión acotado, debemos extender sobre el sensor cada componente de la matriz, como podemos apreciar en la Figura 3.4.

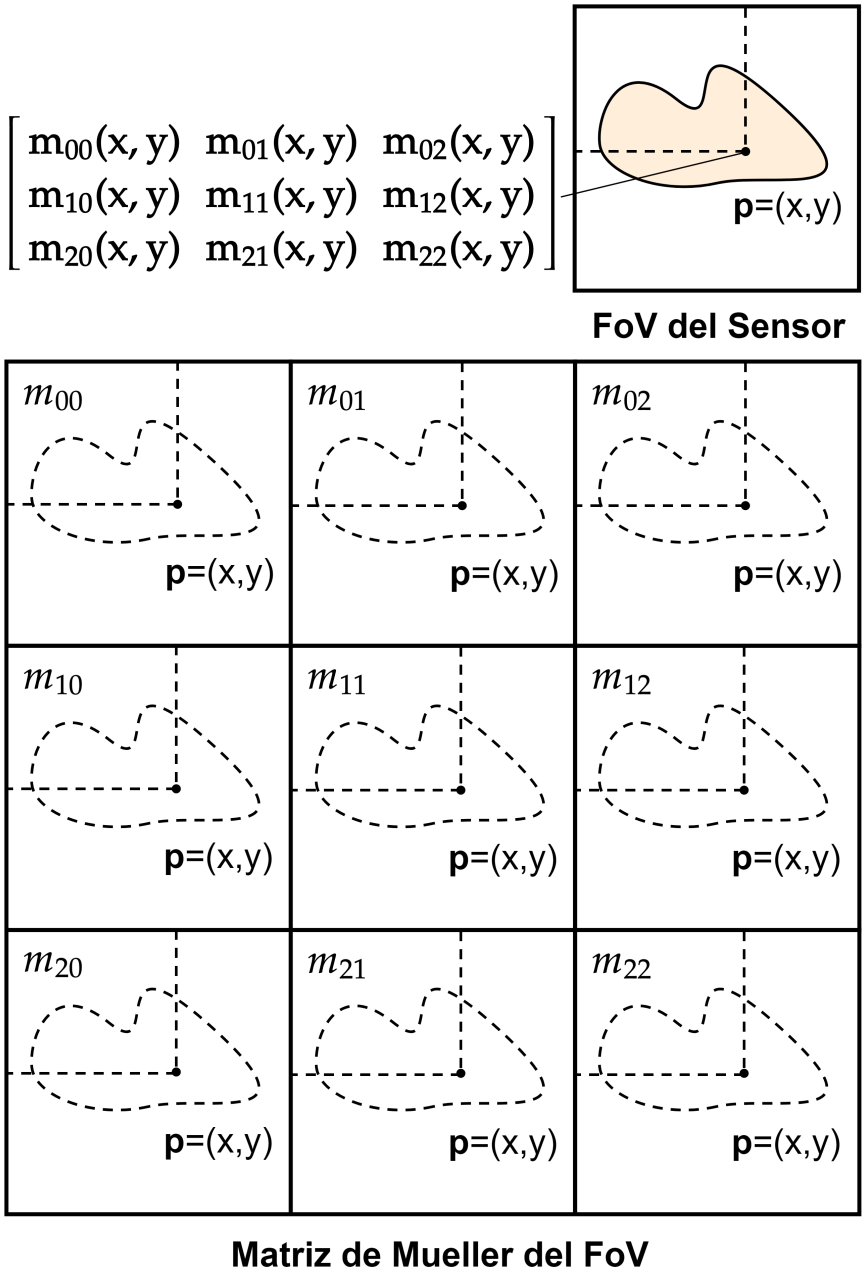

Figura 3.4: Representación gráfica de una matriz de Mueller calculada sobre un campo de visión acotado. Las componentes de la matriz se extienden sobre todos los superpíxeles del sensor y se agrupan en un arreglo bidimensional. Realizado en Mathcha.

A continuación mostraremos las matrices de Mueller de elementos conocidos que son de interés para verificar el procedimiento implementado. Además, calcularemos algunas propiedades polarimétricas utilizando una variante de la descomposición de Lu-Chipman para matrices de Mueller 3×3, válida únicamente cuando la matriz de despolarización es isotrópica. En los tejidos biológicos, la suposición de despolarización isotrópica suele cumplirse en buena medida [29].

3.3.1. Ejemplo 1: Aire

La matriz de Mueller del aire es la más sencilla de todas y nos ayuda a verificar la calibración del polarizador. Si el polarizador y el LED están correctamente posicionados, el resultado debería ser exactamente la matriz identidad. En la Figura 3.5 podemos apreciar los resultados del cálculo de Mueller.

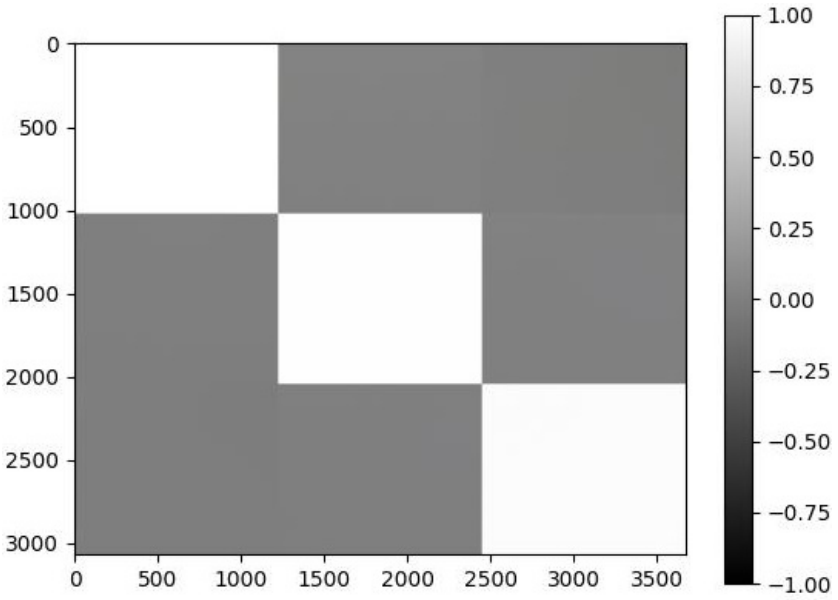

Figura 3.5: Matriz de Mueller del aire.

La imagen en escala de grises indica que las propiedades polarimétricas del aire son acromáticas en el espectro visible. Como la matriz es homogénea, si promediamos en cada componente obtenemos:

$$\hat{\boldsymbol{M}}_A = \begin{pmatrix} 1{,}00 & 0{,}02 & 0{,}00 \\ 0{,}00 & 1{,}00 & 0{,}01 \\ -0{,}01 & 0{,}00 & 0{,}98 \end{pmatrix}. \tag{3.12}$$

Como la matriz medida es prácticamente la identidad, podemos deducir que la matriz de Stokes de entrada y de salida son iguales a menos de una constante multiplicativa. Entonces, el polarizador está girando prácticamente igual en cada cálculo. Descomponiendo por Lu-Chipman:

$$\hat{\boldsymbol{M}}_A = \underbrace{\begin{pmatrix} 1{,}00 & 0{,}00 & 0{,}00 \\ -0{,}02 & 1{,}01 & 0{,}00 \\ 0{,}00 & 0{,}00 & 1{,}01 \end{pmatrix}}_{\boldsymbol{M_\Delta}} \underbrace{\begin{pmatrix} 1{,}00 & 0{,}00 & 0{,}00 \\ 0{,}00 & 0{,}99 & 0{,}02 \\ 0{,}00 & 0{,}01 & 0{,}97 \end{pmatrix}}_{\boldsymbol{M_R}} \underbrace{\begin{pmatrix} 1{,}00 & 0{,}02 & 0{,}00 \\ 0{,}02 & 1{,}00 & 0{,}00 \\ 0{,}00 & 0{,}00 & 1{,}00 \end{pmatrix}}_{\boldsymbol{M_D}}. \tag{3.13}$$

Observamos que las tres matrices son prácticamente la matriz identidad. El aire, al igual que cualquier medio isotrópico, no cambia la polarización de la luz.

3.3.2. Ejemplo 2: Papel de Calco

El papel de calco es un elemento que despolariza muy bien la luz. Su estructura áspera esparce la luz incidente, destruyendo casi por completo su polarización. En la Figura 3.6 vemos la imagen de la matriz de Mueller del papel de calco.

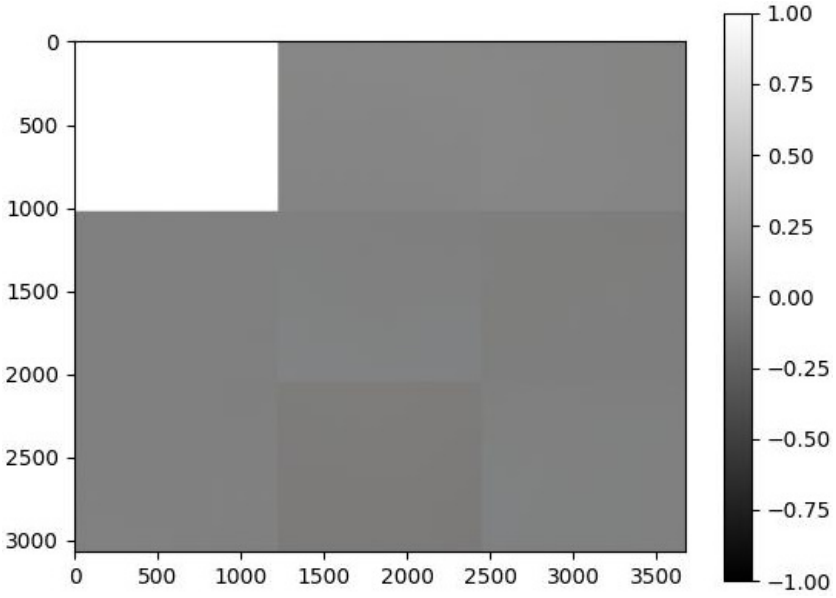

Figura 3.6: Matriz de Mueller del papel de calco.

Si promediamos en cada componente obtenemos:

$$\hat{\boldsymbol{M}}_P = \begin{pmatrix} 1{,}00 & 0{,}05 & 0{,}05 \\ 0{,}01 & 0{,}02 & 0{,}00 \\ 0{,}01 & -0{,}02 & 0{,}01 \end{pmatrix}. \tag{3.14}$$

Descomponiendo por Lu-Chipman:

$$\hat{\boldsymbol{M}}_P = \underbrace{\begin{pmatrix} 1{,}00 & 0{,}00 & 0{,}00 \\ 0{,}01 & 0{,}03 & 0{,}00 \\ 0{,}01 & 0{,}00 & 0{,}03 \end{pmatrix}}_{\boldsymbol{M_\Delta}} \underbrace{\begin{pmatrix} 1{,}00 & 0{,}00 & 0{,}00 \\ 0{,}00 & 0{,}67 & -0{,}02 \\ 0{,}00 & -0{,}70 & 0{,}32 \end{pmatrix}}_{\boldsymbol{M_R}} \underbrace{\begin{pmatrix} 1{,}00 & 0{,}05 & 0{,}05 \\ 0{,}05 & 1{,}00 & 0{,}00 \\ 0{,}05 & 0{,}00 & 1{,}00 \end{pmatrix}}_{\boldsymbol{M_D}}. \tag{3.15}$$

Es interesante notar cómo la descomposición pierde estabilidad cuando $\Delta \simeq 0$, devolviendo una matriz de retardancia $\boldsymbol{M_R}$ que carece de sentido. Esto se debe a que $\boldsymbol{m_\Delta}$ está mal condicionada (A.131). En el caso particular $\Delta = 0$, la descomposición de Lu-Chipman deja de ser única y $\boldsymbol{M_R}$ puede ser una matriz cualquiera.

3.3.3. Ejemplo 3: Polarizador

A diferencia de los elementos anteriores, el polarizador tiene una matriz de Mueller que depende de cómo esté orientado. En la Ecuación (1.28) podemos observar cómo depende la matriz de Mueller del polarizador respecto al ángulo de

diatenuación θ. Veamos cómo responde el cálculo cuando en lugar de la muestra ponemos un polarizador en posición vertical u horizontal. En las Figuras 3.7 y 3.8 podemos ver los resultados del cálculo.

Polarizador Vertical

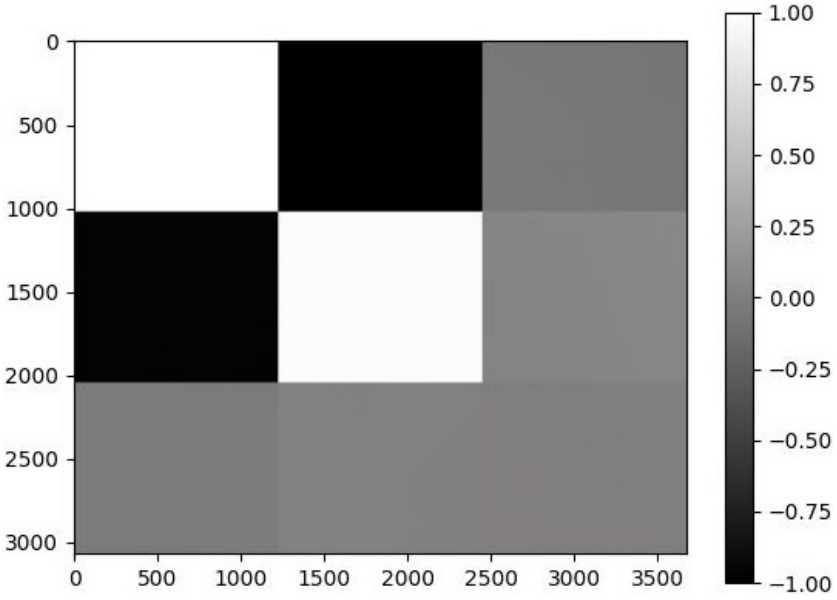

Figura 3.7: Matriz de Mueller del polarizador vertical.

Promediando en cada componente:

$$\hat{\boldsymbol{M}}_V = \begin{pmatrix} 1{,}00 & -0{,}99 & -0{,}05 \\ -0{,}96 & 0{,}98 & 0{,}06 \\ -0{,}02 & 0{,}03 & 0{,}01 \end{pmatrix}. \tag{3.16}$$

Descomponiendo por Lu-Chipman:

$$\hat{\boldsymbol{M}}_V = \underbrace{\begin{pmatrix} 1{,}00 & 0{,}00 & 0{,}00 \\ 0{,}76 & 1{,}84 & 0{,}00 \\ 0{,}59 & 0{,}00 & 1{,}84 \end{pmatrix}}_{\boldsymbol{M}_\Delta} \underbrace{\begin{pmatrix} 1{,}00 & 0{,}00 & 0{,}00 \\ 0{,}00 & 0{,}94 & 0{,}09 \\ 0{,}00 & 0{,}33 & 0{,}05 \end{pmatrix}}_{\boldsymbol{M}_R} \underbrace{\begin{pmatrix} 1{,}0 & -0{,}99 & -0{,}05 \\ -0{,}99 & 1{,}00 & 0{,}04 \\ -0{,}05 & 0{,}04 & 0{,}13 \end{pmatrix}}_{\boldsymbol{M}_D}. \tag{3.17}$$

La descomposición de Lu-Chipman comienza a ser inestable. Las matrices $\boldsymbol{M}_\Delta$ y $\boldsymbol{M}_R$ carecen de interés. La razón se debe al mal condicionamiento de la matriz $\boldsymbol{M}_D$ (A.122) cuando $D \simeq 1$. En el caso del polarizador ideal, la descomposición deja de ser única y tanto $\boldsymbol{M}_\Delta$ como $\boldsymbol{M}_R$ pueden ser matrices cualesquiera.

Polarizador horizontal

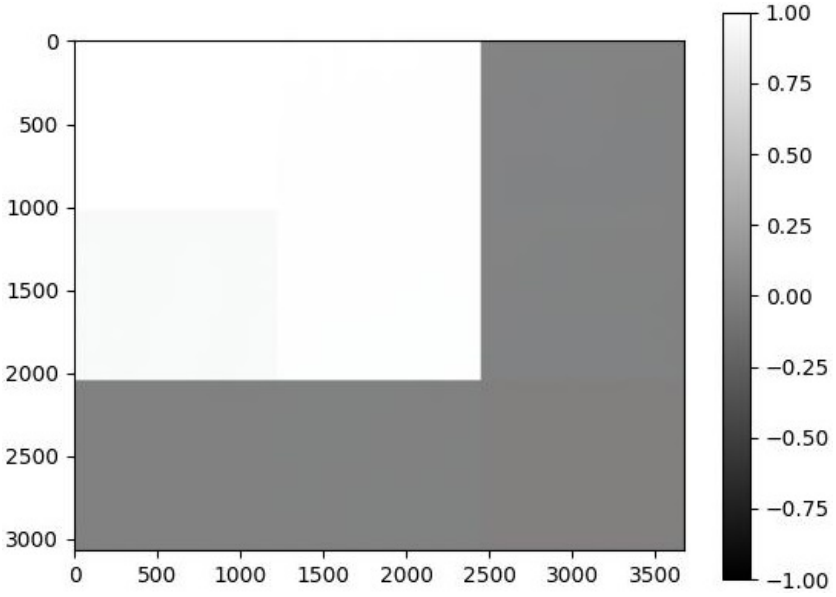

Figura 3.8: Matriz de Mueller del polarizador horizontal.

Promediando en cada componente:

$$\hat{\boldsymbol{M}}_H = \begin{pmatrix} 1{,}00 & 1{,}00 & 0{,}03 \\ 0{,}96 & 1{,}00 & 0{,}03 \\ 0{,}02 & 0{,}02 & 0{,}01 \end{pmatrix}. \tag{3.18}$$

Descomponiendo por Lu-Chipman:

$$\hat{\boldsymbol{M}}_H = \underbrace{\begin{pmatrix} 1{,}00 & 0{,}00 & 0{,}00 \\ 45 & 44 & 0{,}00 \\ 0{,}33 & 0{,}00 & 44 \end{pmatrix}}_{\boldsymbol{M}_\Delta} \underbrace{\begin{pmatrix} 1{,}00 & 0{,}00 & 0{,}00 \\ 0{,}00 & -1{,}00 & -0{,}03 \\ 0{,}00 & -0{,}01 & 0{,}00 \end{pmatrix}}_{\boldsymbol{M}_R} \underbrace{\begin{pmatrix} 1{,}00 & 1{,}00 & 0{,}03 \\ 1{,}00 & 1{,}00 & 0{,}03 \\ 0{,}03 & 0{,}03 & 0{,}00 \end{pmatrix}}_{\boldsymbol{M}_D}. \tag{3.19}$$

Las características más relevantes en estos dos casos son la diatenuación y el ángulo de diatenuación. Se puede verificar en ambos casos que la diatenuación es prácticamente unitaria y el ángulo de diatenuación coincide con el ángulo de rotación del polarizador.

3.3.4. Ejemplo 4: Tejido Denso Conectivo

A diferencia de los casos anteriores, el tejido denso conectivo muestra propiedades polarimétricas que dependen de la longitud de onda λ y la posición del píxel $\boldsymbol{p}$. En la Figura 3.9 podemos ver la transmitancia bajo luz despolarizada m_{00} y en la Figura 3.10 la matriz de Mueller normalizada $\hat{\boldsymbol{M}}$ de una porción de la muestra.

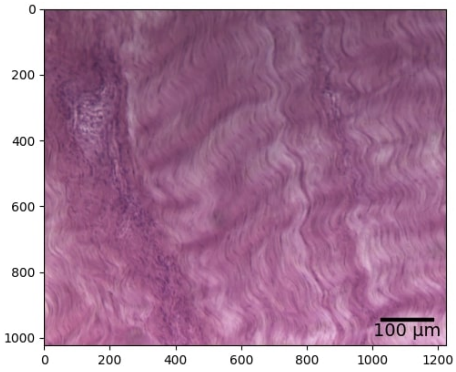

Figura 3.9: Transmitancia de tejido denso conectivo de AmScope [30].

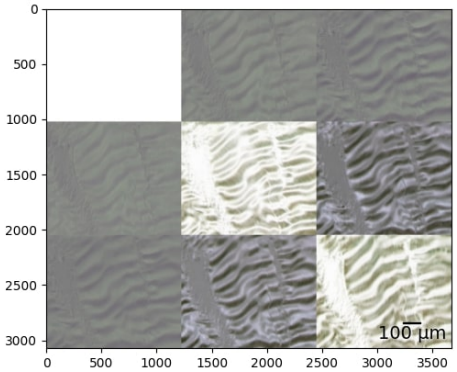

Figura 3.10: Matriz de Mueller normalizada de tejido denso conectivo en codificación RGB.

La estructura colorida de la matriz de Mueller demuestra la dependencia espacial y frecuencial de las propiedades polarimétricas del tejido. Continuamos el análisis en el canal verde, puesto que los tejidos biológicos suelen polarizar mejor la luz con longitud de onda próxima al verde (550 nm). En las Figuras 3.11 - 3.14 vemos la matriz de Mueller del tejido y su descomposición de Lu-Chipman $\boldsymbol{M} = \boldsymbol{M}_\Delta \boldsymbol{M}_R \boldsymbol{M}_D$. Destacamos la predominancia de la matriz de birrefringencia $\boldsymbol{M}_R$, al notar que $\boldsymbol{M}_D$ y $\boldsymbol{M}_\delta$ demuestran un parecido a la matriz identidad.

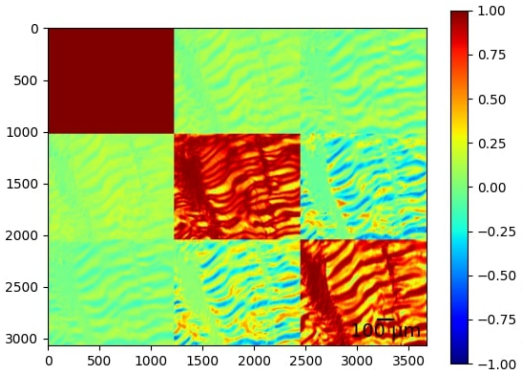

Figura 3.11: Matriz $\boldsymbol{M}$ en canal verde.

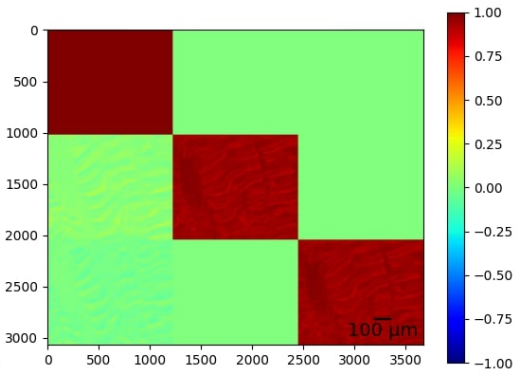

Figura 3.12: Matriz $\boldsymbol{M}_\Delta$ en canal verde.

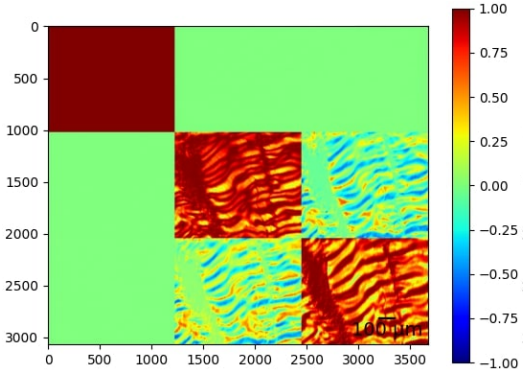

Figura 3.13: Matriz $\boldsymbol{M}_R$ en canal verde.

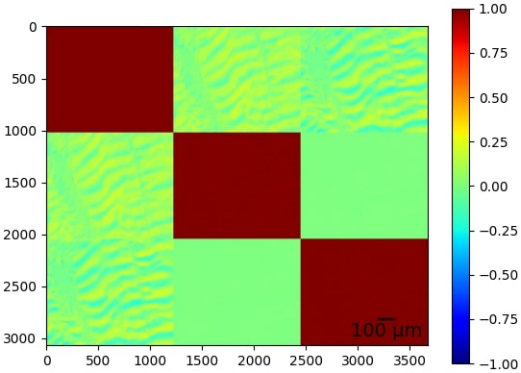

Figura 3.14: Matriz $\boldsymbol{M}_D$ en canal verde.

En las Figuras 3.15 - 3.20 vemos algunas características polarimétricas. Estas están asociadas a las propiedades mecánicas del tejido. La retardancia lineal es la característica de mayor magnitud, asociada a las tensiones internas en el tejido. La diatenuación y la polarizancia son medidas parecidas e indican el nivel de ordenamiento en las fibras de colágeno. El ángulo de polarizancia brinda una idea sobre la distribución de los tejidos. La retardancia circular indica presencia de carbohidratos, principalmente glucosa y fructosa. Por último, la despolarizancia mide irregularidades, tanto en el tejido como en el interior de las células.

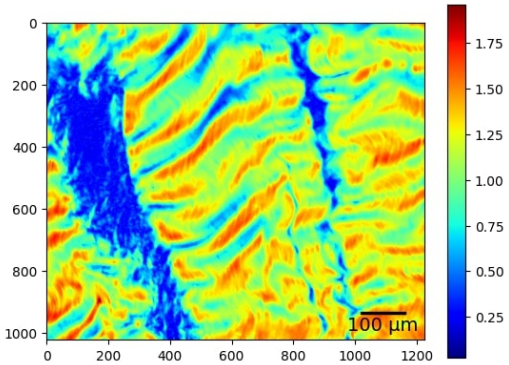

Figura 3.15: Retardancia Lineal.

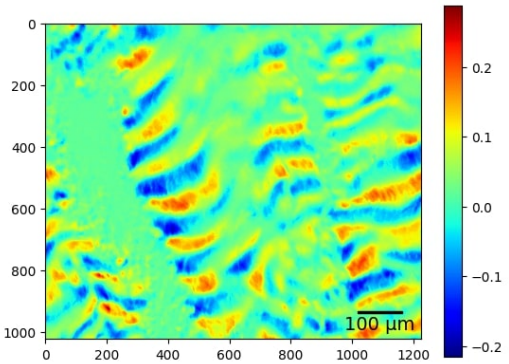

Figura 3.16: Retardancia Circular.

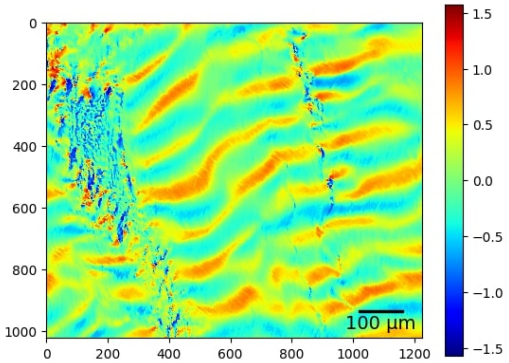

Figura 3.17: Ángulo de Polarizancia.

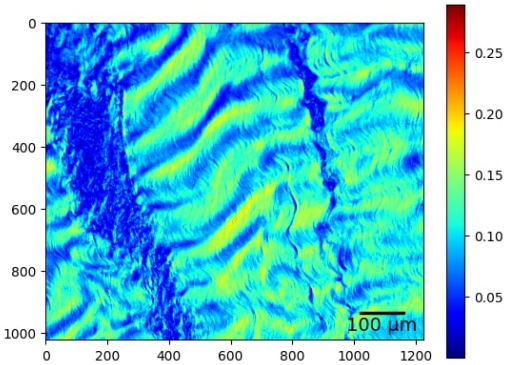

Figura 3.18: Polarizancia Lineal.

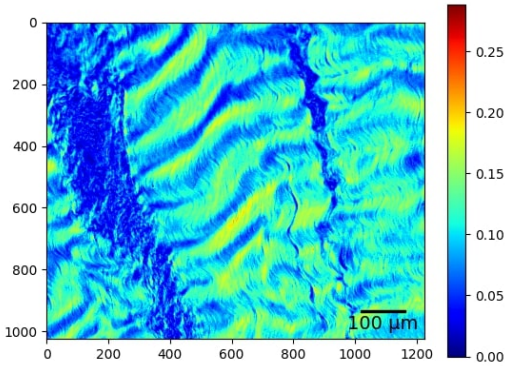

Figura 3.19: Diatenuación Lineal.

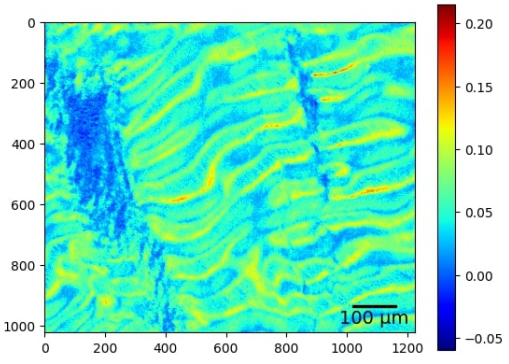

Figura 3.20: Poder de Despolarizancia.

3.4. Escaneo de Muestra Completa

Diseñamos e implementamos el código necesario para controlar los motores que mueven el portamuestras, capturar la imagen de la muestra en cada posición y rotar el polarizador. Para controlar los motores debemos programar la Raspberry Pi, utilizando alguno de los distintos módulos de Python ya implementados. Se optó por utilizar la librería 'RPi.GPIO', dada la disponibilidad de ejemplos y documentación. Teniendo el esquema de conexión de los motores a la placa, se asignan los pines GPIO a su función correspondiente, los cuales serán utilizados para accionar los motores. Definimos funciones que, a partir de dos parámetros de entrada, uno correspondiente a la longitud del desplazamiento deseado y otro a la dirección, envía al motor seleccionado una secuencia de pasos acorde a su modelo.

El siguiente elemento clave para el sistema que debe controlarse es el sensor (cámara polarizada). Este se conecta mediante USB a la computadora y debe ser capaz de capturar la imagen en cada posición de la muestra a la que llega el sistema. Esto requiere también de una librería específica para el *hardware*. Para esto decidimos utilizar 'Simple_PySpin', un módulo basado en clases simplificado de la librería original 'PySpin'. Haciendo uso de distintas funcionalidades incluidas se logró definir una función que capture la imagen correctamente, pudiendo modificar valores relacionados con la configuración de la cámara, como el modo de captura de imagen, tiempo de exposición, etc.

Habiendo logrado el correcto funcionamiento del código para controlar cada motor y capturar las imágenes, comenzamos el desarrollo del programa integrador que combine ambas partes para poder hacer un barrido fotográfico de la muestra.

Algoritmo 2 capturar_muestra(pasos_x, pasos_y)

```
x = 0; y = 0; dx = 1; dy = 0                          ▷ Inicializa estado
for i = 1 to pasos_x × pasos_y do
    take_mueller_stokes()                  ▷ Capturamos girando polarizador
    if x in Borde Lateral then     ▷ Si alcanza borde, actualiza estado
        dx, dy = actualizar_incrementos(dx,dy)
    end if
    x, y = x+dx, y+dy                                  ▷ Aplica actualización
    mover_motores(dx,dy)                               ▷ Según signo de dx, dy
end for
#Volvemos a posición inicial
mover_motor('Y', 'Backwards', pasos_y)
mover_motor('X', 'Backwards', pasos_x)
```

Para esta parte secuenciamos los pasos de movimiento en las coordenadas X e Y, la rotación de polarizador y la captura de imagen, de una manera eficiente y que abarque la totalidad de la muestra. Por simplicidad se probó con un movimiento en líneas horizontales (*raster scan*). Es decir, comienza en la esquina superior izquierda de la muestra y se avanza paso a paso hasta el borde derecho, desciende

una posición y se mueve hasta el borde izquierdo, desciende de nuevo y continúa la secuencia hasta capturar la totalidad de las fotos. En cada paso se toman seis fotografías: una por cada ángulo en que se coloca el polarizador motorizado. Finalizado el escaneo, se vuelve a posicionar la platina en la posición original regresando sobre los pasos realizados, como se indica en `capturar_muestra()` (ver Algoritmo 2). En la Figura 3.21 presentamos un diagrama del escaneo descrito superpuesto a la muestra completa de tejido de hoja de pino obtenida mediante *stitching*.

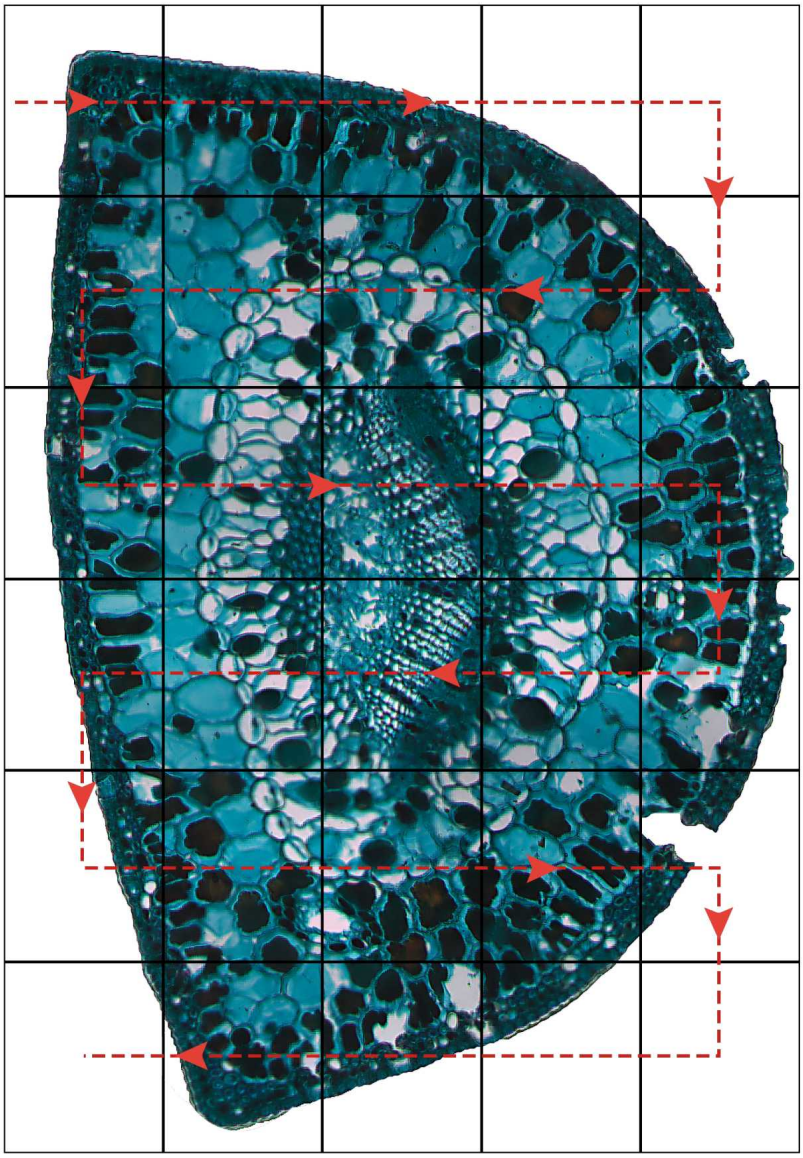

Figura 3.21: Diagrama del escaneo superpuesto a la muestra completa de tejido de hoja de pino obtenida mediante *stitching*. Imagen modificada en Photoshop [31].

3.5. Interfaz de Usuario

Para controlar el movimiento de la platina, el polarizador y capturar las imágenes con mayor comodidad, se desarrolló una interfaz gráfica (GUI) utilizando la librería 'PyQt' [32]. La GUI asigna a distintos botones en pantalla las funciones correspondientes al movimiento de los motores y de captura de la cámara, permitiendo controlar el sistema de microscopía completamente. Los botones en forma de punta de flecha mueven la platina en los ejes horizontal y vertical, mientras que los de flecha curva hacen girar el polarizador en un sentido u otro. Cuenta además con un *display* en tiempo real de lo que observa la cámara, obteniendo en cada instante la imagen que se observaría al dar la orden de captura. Ver Figura 3.22.

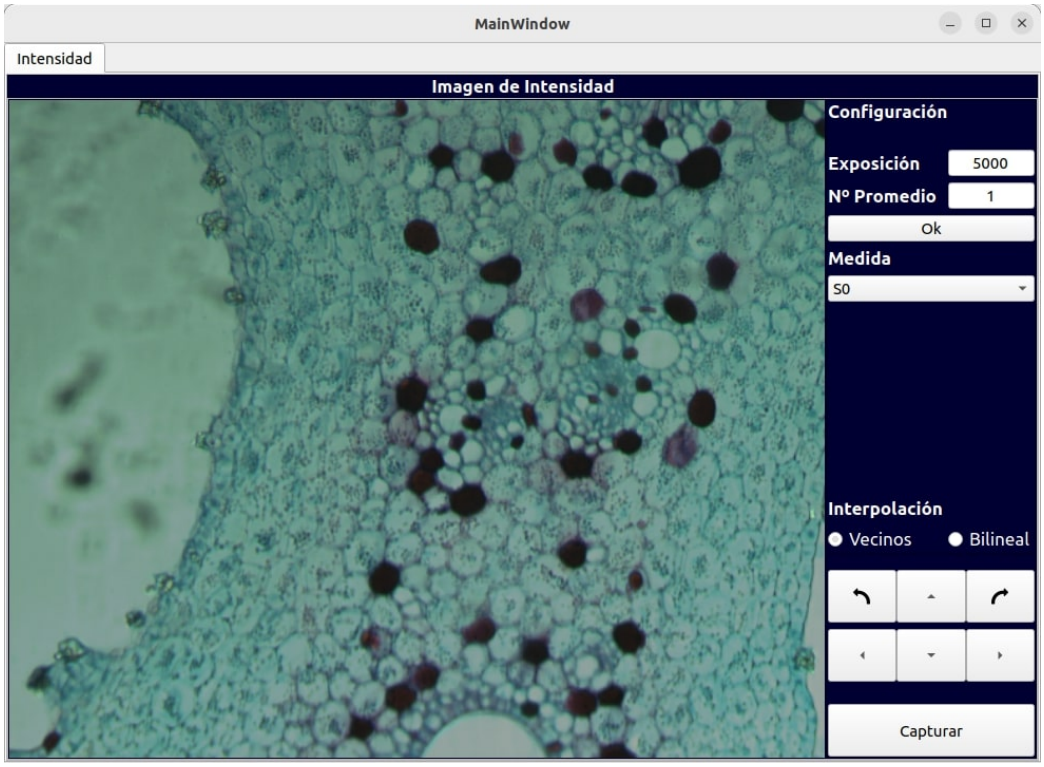

Figura 3.22: GUI funcionando sobre tejido de calabaza.

Para configurar correctamente el sensor, se agregaron entradas correspondientes al tiempo de exposición T y el número N de capturas que se usan para promediar los parámetros de Stokes. Ambos se recuperan y se usan como parámetros una vez ingresados. El tiempo de exposición T es el intervalo de tiempo durante el cual se promedia la información de la luz enfocada en el sensor, mientras que N indica cuántas muestras a tomar (medidas de intensidad) para promediar y obtener una medida con mitigación del ruido en cada uno de los píxeles. Se permite seleccionar un tipo de interpolación cromática entre dos opciones: vecinos más cercanos y bilineal. El botón "Capturar" añade la función de escaneo automático, que ejecuta el código descrito en la Sección 3.4. Esto quiere decir que desde la GUI podemos

posicionar la platina en un punto que consideremos propicio para capturar de forma automática la muestra completa de esa porción de tejido. La lista desplegable "Medida", proporciona al usuario la posibilidad de elegir qué tipo de información mostrar en la pantalla. Existen en total nueve medidas que se pueden observar en tiempo real: I_0, I_{45}, I_{90}, I_{135}, S_0, S_1, S_2, DoLP y AoP. En la Figura 3.22 se puede observar con nitidez la imagen S_0 de la muestra a capturar y la configuración previamente descrita.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 4

Stitching de Muestra Completa

El *stitching* o *image stitching* es el proceso de combinar múltiples imágenes fotográficas con campos de visión superpuestos para producir una nueva imagen compuesta más grande y completa. Es decir, combina varias imágenes tomadas, en general a alta resolución, en una imagen compuesta que debe consistir en imágenes colocadas en la posición correcta. Uno de sus principales objetivos es hacer que los bordes entre las imágenes (costuras/*seams*) no sean visibles, de forma tal que la imagen completa parezca una única toma.

Su aplicación más común se da en fotografía y visión por computadora, para crear imágenes panorámicas de alta resolución o para generar vistas con amplio campo de visión (FoV). Aunque, con el estudio y desarrollo de esta técnica, se ha extendido su uso a múltiples áreas. El mosaico de documentos (*document mosaicing*), la estabilización de cámaras, la creación de mapas por imagen satelital y la creación de imágenes médicas son ejemplos de su versatilidad. En nuestro caso, entonces, el *stitching* nos permitirá unir imágenes de distintas zonas de un tejido microscópico si aseguramos que las imágenes capturadas tienen un grado de superposición suficiente.

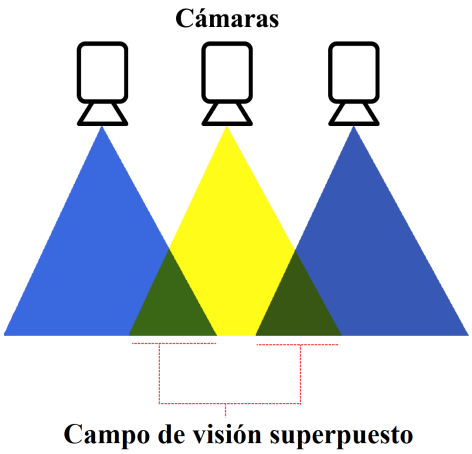

Figura 4.1: Ejemplo de *stitching* panorámico. Realizado en Photoshop.

Lograr construir una imagen a partir de varias fotografías incompletas representa un desafío debido a la variabilidad que puede haber entre una foto y otra (perspectiva, exposición, rotación, etc.). Además, la limitada información que puede existir en nuestro conjunto de imágenes es un obstáculo, siendo la presencia de superposición entre imágenes uno de los requisitos esenciales para lograr un buen *stitching*. Otros problemas importantes a tener en cuenta son presencia de paralaje, distorsión de lente y movimiento de escena.

En las Figuras 4.2 - 4.5 se aprecia un ejemplo de esta técnica aplicada sobre unas capturas fotográficas del edificio de la Facultad de Ingeniería, UdelaR. Dos imágenes I_1 e I_2, capturadas con la cámara de un celular, muestran diferentes partes de la misma escena, con una región de superposición presente en ambas imágenes. Las imágenes deben ser ensambladas (*stitched*) para generar una imagen de mosaico final I. Sin embargo, simplemente pegando una región de la izquierda de I_1 y una región de la derecha de I_2 produce bordes artificiales visibles en la costura entre las imágenes. Esto se debe a diferencias en la ganancia de la cámara, la iluminación de la escena o desalineaciones geométricas. Además, es muy notable la diferencia del ángulo de dirección de la cámara en cada foto, lo que genera una diferencia en la perspectiva, problema que el *stitching* intenta solucionar realizando una proyección de las imágenes a un plano.

Figura 4.2: Imagen de entrada I_1.

Figura 4.3: Imagen de entrada I_2.

Figura 4.4: Pegado de imágenes I_1 e I_2.

Figura 4.5: Mosaico I realizado con *stitching*.

El objetivo de un algoritmo de *stitching* es producir un mosaico visualmente plausible con dos propiedades deseables. En primer lugar, el mosaico debe ser lo más parecido a las imágenes de entrada, tanto geométrica como fotométricamente. En segundo lugar, la costura entre las imágenes ensambladas debe ser invisible.

Las etapas a nivel general de un proceso de *stitching* de imágenes basado en características puede reducirse a cuatro pasos: Adquisición, Calibración, Registro y Fusión (*Acquisition*, *Calibration*, *Registration* y *Blending*) [33].

La Adquisición de imágenes puede realizarse de distintas maneras e incluso de distintas fuentes; dependiendo de cómo se lleve a cabo, podría cambiar drásticamente el resultado final. En esta etapa se define el porcentaje de solapamiento entre fotos y la diferencia fotométrica del conjunto de imágenes de entrada.

Las imperfecciones ópticas se reducen mediante la Calibración, que también implica la disminución de la diferencia de exposición entre imágenes de entrada. En esta etapa se suele extraer los parámetros de la cámara para la imagen final.

El Registro es el proceso de alinear múltiples imágenes utilizando las transformaciones apropiadas. En esta parte del proceso se determinan las correspondencias geométricas entre cada par de imágenes a fusionar. El Registro usualmente se clasifica en dos tipos: directo o basado en características (*feature-based method*). Cada uno puede llevarse a cabo de distintas maneras y con distintos algoritmos. En este trabajo nos centraremos en el Registro basado en características.

La Fusión de imágenes se utiliza para eliminar la costura visible a lo largo del área solapada entre las imágenes de entrada. Durante el *stitching* es posible que se produzcan desalineaciones geométricas que causen discontinuidades en la imagen final, a las que llamamos costuras. El algoritmo de *blending* tiene como finalidad eliminarlas.

4.1. Algoritmo de *Stitching*

A un nivel más específico y dependiendo de qué algoritmo se utilice, el proceso típico de *stitching* basado en características, en general y sin tener en cuenta la adquisición de imágenes, implica:

- ***Feature detection*** (Detección de características): Identificar puntos de interés y descriptores (esquinas, bordes, etc.) en cada imagen de entrada.

- ***Feature matching*** (Emparejamiento de características): Comparar las características entre pares de imágenes para encontrar correspondencias entre ellas en zonas de solapamiento. Esto se puede lograr utilizando algoritmos como ORB (*Oriented FAST and Rotated BRIEF*), SIFT (*Scale-Invariant Feature Transform*) o SURF (*Speeded Up Robust Features*) para la detección de características y comparando los puntos clave.

- ***Homography estimation*** (Estimación de la homografía): Utilizar las correspondencias de características para calcular la homografía que describe la transformación geométrica (rotación, escalado, traslación) necesaria para alinear las imágenes.

- ***Warping*** (Alineación de imágenes): Aplicar la homografía estimada a cada imagen para alinearlas correctamente en un mismo sistema de coordenadas.

- ***Calibration & Blending*** (Calibración y Fusión de imágenes): Corregir los parámetros de imagen para minimizar las diferencias (lente, exposición, color, etc.), alinear las imágenes y fusionarlas para crear una imagen panorámica única sin costuras (*seamless*).

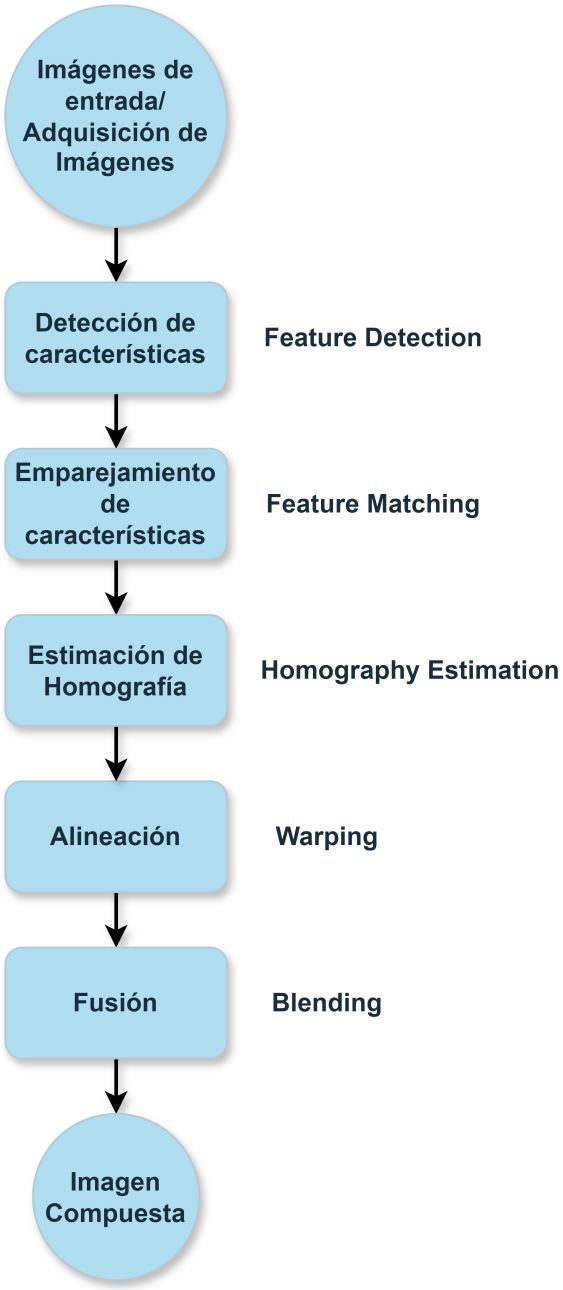

Figura 4.6: Diagrama de bloques de un algoritmo de *stitching*. Realizado en Drawio [34].

En la siguiente sección se detallará el procedimiento de cada uno de estos pasos para el algoritmo específico que se utilizó en este trabajo. Cada uno de estos

procesos influye en el resultado final con cierto peso y por cuenta propia. Elegir un algoritmo de detección de características u otro puede variar la imagen final, e incluso la posibilidad o no de llevar a cabo el *stitching*. Por esta razón vale la pena ahondar en cada uno de ellos y conocer cómo funcionan.

4.1.1. *Feature Detection & Feature Matching*

La detección de características es un problema fundamental en visión por computadora y procesamiento de imágenes. Si bien es un paso de procesamiento de bajo nivel, constituye una parte esencial en muchas aplicaciones basadas en tratamiento de imágenes, principalmente el *stitching*. La comparación directa de las intensidades de los píxeles es uno de los enfoques posibles del registro de imágenes. El otro enfoque principal consiste en extraer primero las características distintivas de cada imagen, a continuación emparejar las características individuales para establecer una correspondencia global y, por último, estimar la transformación geométrica entre las imágenes. Este tipo de enfoque se ha utilizado desde los comienzos del *matching* de imágenes [35] y, más recientemente, ha ganado popularidad en aplicaciones de unión de imágenes [36][37][38][39].

No existe una definición universal de lo que es una característica (*feature*) y la definición puede variar dependiendo del problema o del tipo de aplicación en el que nos encontremos. En el contexto actual, se suele denominar característica a un elemento de información sobre el contenido de una imagen. Usualmente, queremos saber si una determinada región de la imagen tiene ciertas propiedades. Las características pueden ser estructuras específicas de la imagen, como bordes, curvas u objetos. Además, también pueden ser el resultado de una operación de detección de características aplicada a la imagen, como el análisis por vecindad. Por lo tanto, podemos decir que una característica es cualquier elemento de información relevante para resolver la tarea computacional en determinada aplicación. En este sentido, es fácil identificar el concepto de características con el manejado en aprendizaje automático y reconocimiento de patrones en general, aunque el procesamiento de imágenes suele abordar características más complejas. Podemos dividir las características de una imagen en dos grandes grupos: globales y locales. Las globales describen la imagen como un todo y pueden interpretarse como una propiedad descriptora de la imagen a partir de todos sus píxeles. Por otro lado, las locales se centran en partes de la imagen en las que se tratan de encontrar puntos clave para describir esas regiones. Los algoritmos de *stitching* suelen concentrarse en estas últimas.

En general, la detección de características implica una función con intensidades de píxel como entrada y estructuras de imagen que indican diferentes propiedades específicas como salida. Es decir, una descripción de la imagen o de alguna región. De esta manera, se puede obtener en cada imagen un punto o conjunto de puntos clave (*key points*) que posean ciertas características, teniendo también su ubicación en la imagen, lo que es particularmente importante en *stitching*. La apariencia local alrededor de cada punto clave se describe de alguna manera que idealmente es invariante a cambios de iluminación, traslación, escala y rotación en

el plano, obteniendo así un vector descriptor para cada punto clave. Puede inferirse aquí que la detección puede, a su vez, dividirse en identificación y descripción de puntos clave. Los algoritmos más usuales para identificación y/o descripción de puntos claves son el detector de esquinas de Harris, SIFT (*Scale Invariant Feature Transform*), SURF (*Speeded Up Robust Feature*), ORB (*Oriented FAST and Rotated BRIEF*) y FAST (*Features from Accelerated Segment Test*). En la Figura 4.7 vemos un ejemplo de características extraídas de una imagen. Una introducción y comparativa de estos métodos puede encontrarse en la literatura [40][41][42].

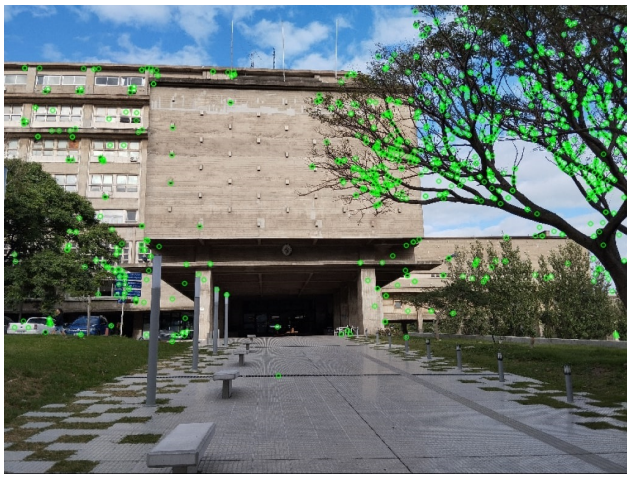

Figura 4.7: Ejemplo de características detectadas (círculos verdes) mediante SIFT en una fotografía del edificio de Facultad de Ingeniería, UdelaR.

Teniendo en mente el objetivo de formar una imagen completa a partir de un conjunto de imágenes, el siguiente paso es encontrar cuáles de estos puntos se repiten y en qué imágenes del conjunto. Esto es lo que buscamos con el emparejamiento de características o *feature matching*. Una vez que se han detectado las características y descriptores de dos o más imágenes, se puede establecer coincidencias de características entre estas imágenes. Ver Figura 4.8.

La interrogante de cómo encontrar estas coincidencias de manera eficiente y precisa no es trivial. El rendimiento de los métodos de emparejamiento basados en puntos clave depende tanto de las propiedades de los puntos de interés subyacentes como de la elección de los descriptores de imagen asociados. Por lo que debe utilizarse detectores y descriptores adecuados al contenido de las imágenes en la aplicación concreta en que se esté trabajando. Por ejemplo, no hubiera sido beneficioso usar un detector de manchas sobre uno de esquinas en la Figura 4.8, puesto que se trata de la imagen de un edificio, con geometrías bien definidas por segmentos de recta. Si tenemos buenos descriptores, será más fácil encontrar correspondencias entre imágenes. Los algoritmos *Brute-Force Matcher*, FLANN (*Fast Library for Approximate Nearest Neighbors*) o RANSAC (*Random Sample Consensus*) son usualmente utilizados para esta tarea. Normalmente, dada la descripción de una característica v, en una imagen I_1, para encontrar la mejor coincidencia en otra imagen I_2, se define una distancia que compare los descriptores y

Figura 4.8: Ejemplo de características emparejadas en las fotografías del edificio de Facultad de Ingeniería, UdelaR.

se calcula con ella la distancia para cada característica de I_2. La distancia mínima corresponde a la mejor coincidencia, aunque esto no necesariamente implica que el emparejamiento sea válido. Entonces, se definen reglas decisorias para separar las coincidencias verdaderas (*actual matches*) de las falsas. Más concisamente, tomando como distancia entre dos vectores de características la Suma de Diferencias al Cuadrado (SSD), la mejor coincidencia para la característica v de I_1 será la característica w^* de I_2 que cumpla:

$$SSD(v, w^*) = \min_{w \in I_2} \{SSD(v, w)\} = \min_{w \in I_2} \left\{ \sum_i (v_i - w_i)^2 \right\}. \tag{4.1}$$

A partir de estas coincidencias se debe definir cuáles tomar como válidas, por ejemplo, usando un valor de umbral o alguna operación a partir de las distancias calculadas. Se obtiene entonces un conjunto de características coincidentes (o no, si no se encuentra ninguna que cumpla los requerimientos) entre dos imágenes. Entonces, se pueden emparejar y continuar con el resto de las imágenes. En *stitching*, este emparejamiento sirve para completar las imágenes, usando la información en un entorno de la región que se repite.

Algortimo SIFT

El algoritmo que finalmente utilizamos para esta parte del Registro es SIFT, por lo que una breve introducción a su funcionamiento es menester. Esta técnica permite identificar características que son invariantes a cambios en la iluminación, el tamaño y la rotación, así como robustas frente a pequeñas variaciones en la perspectiva. Para lograr esto, primero se crea un espacio-escala piramidal utilizando una aproximación del operador LoG (*Laplacian of Gaussian*), mediante filtros de diferencias gaussianas de varios tamaños. Luego, se identifican esquinas que actúan como máximos locales y se asigna una dirección principal a cada punto de interés. A continuación, se representa cada punto de interés mediante vectores que describen los histogramas de orientación de gradientes, calculados en una cuadrícula que

cubre el entorno de los puntos. Los pasos principales para detectar y emparejar puntos característicos en SIFT son los listados a continuación [43].

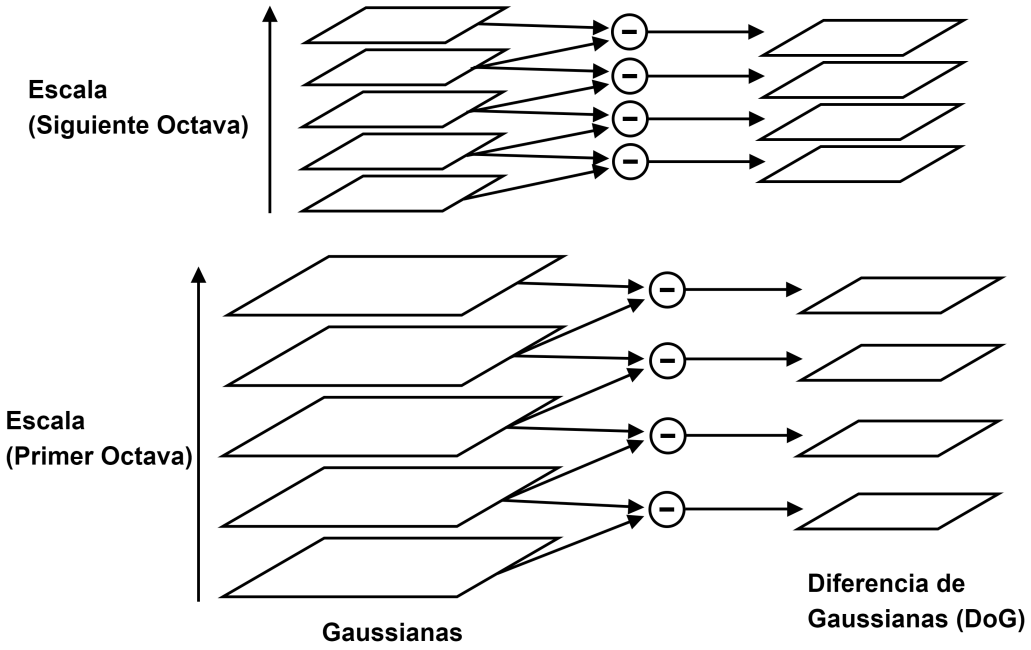

Figura 4.9: Representación del espacio de escala Gaussiano. Realizado en Mathcha.

- **Detección de extremos en el espacio de escala**

  Para lograr la invariancia de escala, SIFT se construye sobre el espacio de escala Gaussiano: una representación multiescala de imagen que simula la familia de todos los acercamientos (*zoom*) posibles a través de versiones cada vez más borrosas de la imagen de entrada. En este popular marco multiescala, la convolución Gaussiana actúa como una aproximación del desenfoque óptico, y el núcleo Gaussiano aproxima la función de dispersión puntual de la cámara. Así, el espacio de escala puede interpretarse como una familia de imágenes, cada una de las cuales corresponde a un factor de *zoom* diferente. Por lo tanto, llamaremos *espacio de escala* digital a una familia de imágenes digitales relativas a un conjunto discreto de niveles de desenfoque y diferentes frecuencias de muestreo, todas ellas derivadas de una imagen de entrada con un nivel de desenfoque supuesto. Esta familia se divide en subfamilias de imágenes que comparten una misma frecuencia de muestreo. Dado que en el algoritmo la frecuencia de muestreo se reduce iterativamente en un factor de dos, estas subfamilias se denominan *octavas*. Una representación de esto puede verse en la Figura 4.9.

  Las características de los puntos clave son definidas en SIFT como los extremos del espacio de escala Laplaciano normalizado $\sigma^2 \Delta v$ ($\sigma \in \mathbb{R}^+$ desviación estándar del kernel gaussiano; Δ operador laplaciano; v lista de octavas del espacio de escala Gaussiano digital: $v = v^o; o = 1, \ldots, n_{oct}$) [44]. El extremo

Laplaciano se caracteriza inequívocamente por sus coordenadas espacio de escala Gaussiano (σ, x), donde x se refiere a su posición espacial central y σ se relaciona con su tamaño (escala).

Para encontrar la característica de unicidad, la primera etapa busca en el espacio de escala utilizando una función de DoG para identificar los posibles puntos de interés que no varían con la escala y la orientación. El espacio de escala de una imagen se define como $L\,(x, y, \sigma)$, que se obtiene de la convolución de una Gaussiana de escala variable $G\,(x, y, \sigma)$ con una imagen de entrada de entrada $I\,(x, y)$:

$$\begin{aligned} \mathbf{L}(x, y, \sigma) &= \mathbf{G}\,(x, y, \sigma) * \mathbf{I}\,(x, y), \\ \mathbf{G}(x, y, \sigma) &= \tfrac{1}{2\pi\sigma^2}\mathbf{e}^{-\frac{x^2+y^2}{2\sigma^2}}. \end{aligned} \tag{4.2}$$

Para detectar puntos clave, eficientes y robustos, los extremos del espacio de escala se encuentran a partir de la Diferencia de Gaussianas múltiple $D\,(x, y, \sigma)$, que puede calcularse a partir de la diferencia de dos escalas cercanas separadas por un factor multiplicativo constante k:

$$\mathrm{D}(x, y, \sigma) = (\mathrm{G}(x, y, k\sigma) - \mathrm{G}(x, y, \sigma)) * \mathrm{I}(x, y, \sigma) - \mathrm{L}(x, y, \sigma). \tag{4.3}$$

- **Localización de puntos clave (*Key Points*)**

  En cada punto candidato hallado en el paso anterior, se refina un modelo detallado para determinar la ubicación y la escala. Los puntos clave se filtran y se seleccionan en función de medidas de su estabilidad (por ejemplo, el contraste); y luego se calcula una ponderación del gradiente y su orientación para formar un vector de características $f(m, \theta)$:

$$\begin{aligned} &\mathrm{m}(x, y) = \sqrt{(\mathrm{L}(x+1, y) - \mathrm{L}(x-1, y))^2 + (\mathrm{L}(x, y+1) - \mathrm{L}(x, y-1))^2}, \\ &\theta(x, y) = \arctan 2\left(\frac{(\mathrm{L}(x, y+1) - \mathrm{L}(x, y-1)}{\mathrm{L}(x+1, y) - \mathrm{L}(x-1, y)}\right). \end{aligned} \tag{4.4}$$

- **Descripción de puntos clave (*Key Points*)**

  Los gradientes locales de la imagen se miden en la escala seleccionada en la región alrededor de cada punto clave. Se transforman en una representación que tolera niveles significativos de distorsión local de la forma y cambios en la iluminación.

- ***Matching***

  Sean $\mathcal{L}^A$ y $\mathcal{L}^B$ conjuntos de descriptores asociados a los puntos clave detectados en las imágenes u^A y u^B . El emparejamiento es realizado considerando cada descriptor asociado a la lista $\mathcal{L}^A$ y encontrando una posible coincidencia en la lista $\mathcal{L}^B$. El primer descriptor $f^a \in \mathcal{L}^A$ se empareja con el descriptor $f^b \in \mathcal{L}^B$ que minimiza la distancia euclídea entre descriptores:

$$f^b = \underset{f \in \mathcal{L}^B}{\arg\min}\, ||f - f^a||_2. \tag{4.5}$$

Emparejar un punto clave con el descriptor f^a requiere entonces calcular las distancias a todos los descriptores en $\mathcal{L}^B$. Un par se considera confiable solo si su distancia absoluta está por debajo de cierto umbral $C^{\text{match}}_{\text{absoluto}}$. De lo contrario, se descarta para evitar la dependencia de una distancia absoluta. El método SIFT utiliza el segundo vecino más cercano para definir qué constituye una coincidencia confiable. SIFT aplica un umbral adaptativo $\|f^a - f^{b'}\| C^{\text{match}}_{\text{relativo}}$ donde $f^{b'}$ es el segundo vecino más cercano, que se agrega a la lista de coincidencias:

$$f^{b'} = \underset{f \in \mathcal{L}^B \setminus \{f^b\}}{\arg\min} ||f - f^a||_2. \tag{4.6}$$

4.1.2. *Homography Estimation & Warping*

Estimación de Transformaciones

Antes de poder alinear imágenes, necesitamos establecer las relaciones matemáticas que mapean las coordenadas de los píxeles de una imagen a otra [39]. Existe una gran variedad de modelos de movimiento paramétricos, desde simples transformaciones 2D a modelos de perspectiva plana, rotaciones de cámara 3D, distorsiones de lente y el mapeado de imágenes no planas (por ejemplo, cilíndricas). Para facilitar el trabajo con imágenes a distintas resoluciones, adoptamos una variante de las coordenadas normalizadas utilizadas en gráficos de computadora. Para una imagen rectangular, las coordenadas de los píxeles oscilan en el intervalo $[-1, 1]$ en el eje mayor y $[-a, a]$ a lo largo del eje más corto, donde a es la inversa de la relación de aspecto. Para una imagen con anchura W y altura H, las ecuaciones que asignan coordenadas de píxeles enteros $\overline{\boldsymbol{x}} = (\overline{x}, \overline{y})$ a coordenadas de dispositivo normalizadas $\boldsymbol{x} = (x, y)$ son:

$$\begin{cases} x = \frac{2\overline{x} - W}{S}, \\ y = \frac{2\overline{y} - H}{S}, \end{cases} \quad \text{donde } S = \text{máx}(W, H). \tag{4.7}$$

Obsérvese que si trabajamos con imágenes en pirámide, tenemos que reducir a la mitad el valor de S después de cada paso de decimación en lugar de volver a calcularlo a partir de $\text{máx}(W, H)$, ya que los valores (W, H) pueden redondearse o truncarse de forma impredecible. En el resto de esta subsección, utilizaremos coordenadas de dispositivo normalizadas para referirnos a las coordenadas de los píxeles.

Como en este trabajo nos enfocamos en conjuntos de imágenes tomadas solamente con traslación de cámara, sin diferencias de ángulo o distancia entre fotos, las transformaciones de imágenes utilizadas durante esta etapa deben ser planas. A pesar de ello, se presentarán algunos ejemplos usando transformaciones 3D durante el *warping* final. Dentro las transformaciones planas (2D), ilustradas en la Figura 4.10, encontramos:

- **Traslación**

  Las traslaciones en 2D pueden escribirse como: $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}$. Matricialmente:

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{t} \end{bmatrix} \tilde{\boldsymbol{x}}, \tag{4.8}$$

donde $\boldsymbol{I}$ es la matriz identidad 2×2 y $\tilde{\boldsymbol{x}} = (x, y, 1)$ es la coordenada 2D homogénea o proyectiva.

- **Rotación y Traslación**

  Esta transformación también se conoce como movimiento rígido en 2D o transformación euclidiana en 2D (ya que las distancias euclidianas se conservan). Puede escribirse como: $\boldsymbol{x}' = \boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}$. Matricialmente:

$$\boldsymbol{x}' = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \tilde{\boldsymbol{x}}, \tag{4.9}$$

$$\text{donde } \boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{4.10}$$

  es una matriz de rotación ortogonal con $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$ y $|\boldsymbol{R}| = 1$.

- **Rotación escalada**

  La rotación escalada, también conocida como transformación de similitud, puede expresarse como: $\boldsymbol{x}' = s\boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}$, donde s es un factor de escala arbitrario. También puede escribirse como:

$$\boldsymbol{x}' = \begin{bmatrix} s\boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \tilde{\boldsymbol{x}} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} \tilde{\boldsymbol{x}}, \tag{4.11}$$

  donde ya no requerimos que $a^2 + b^2 = 1$. La transformación de similitud preserva los ángulos entre las líneas.

- **Afín**

  La transformación afín se escribe como $\boldsymbol{x}' = \boldsymbol{A}\tilde{\boldsymbol{x}}$, donde $\mathbf{A}$ es una matriz arbitraria de 2×3, es decir:

$$\boldsymbol{x}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \tilde{\boldsymbol{x}}. \tag{4.12}$$

  Las líneas paralelas permanecen paralelas bajo transformaciones afines.

- **Homografía**

  La transformación proyectiva, también conocida como transformación de perspectiva u homografía, opera en coordenadas homogéneas $\tilde{\boldsymbol{x}}$ y $\tilde{\boldsymbol{x}}'$:

$$\tilde{\boldsymbol{x}}' \sim \tilde{\boldsymbol{H}}\tilde{\boldsymbol{x}}, \tag{4.13}$$

  donde $\sim$ denota igualdad a escala y $\tilde{\boldsymbol{H}}$ es una matriz arbitraria de 3×3. Nótese que $\tilde{\boldsymbol{H}}$ es en sí misma homogénea, i.e, solo está definida a escala. La coordenada homogénea resultante $\tilde{\boldsymbol{x}}'$ debe ser normalizada para obtener un resultado inhomogéneo $\boldsymbol{x}'$, es decir:

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \quad \text{y} \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}. \tag{4.14}$$

  Las homografías preservan las líneas rectas y, junto con la transformación afín, son las predilectas para esta etapa del *stitching*.

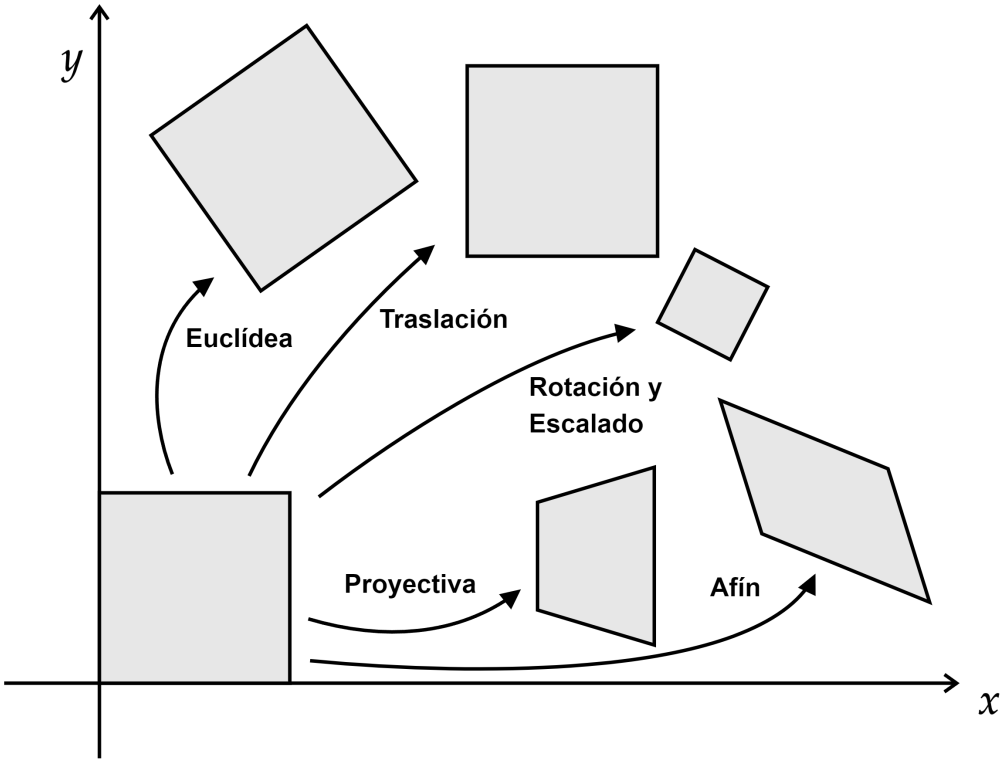

Figura 4.10: Conjunto básico de transformaciones planas 2D. Realizado en Mathcha.

Aplicación

La homografía es una transformación proyectiva que determina una correspondencia entre dos figuras geométricas planas. De esta manera, se puede mapear puntos en una imagen a sus puntos correspondientes en otra imagen. Esto permite alinear y registrar imágenes tomadas desde diferentes perspectivas o con diferentes cámaras, estimando una transformación que convierta una región de una imagen en una región de otra imagen. Cabe destacar que podríamos estimar otro tipo de transformación, como la afín. Una transformación afín conserva líneas rectas y paralelas, mientras que la homografía conserva solo líneas rectas, por lo que la segunda contempla casos que la primera no. Se puede decir que una transformación afín es una subclase de transformaciones proyectivas (homografías), por lo que es más restrictiva y tiene menos grados de libertad.

Por ejemplo, supongamos que tenemos dos imágenes tomadas desde diferentes ángulos de la misma escena y queremos alinearlas para que coincidan y se puedan superponer correctamente, como en el caso presentado en la Sección 4.5. Para hacer esto, es necesario estimar la homografía que describe la transformación geométrica entre las dos imágenes. Para lograr este objetivo hacemos uso de los pasos previos, donde ya contamos con puntos emparejados entre dos imágenes. Utilizamos algoritmos como RANSAC para encontrar una estimación inicial de la homografía basada en una muestra aleatoria de estos puntos clave. Una vez realizada una es-

timación inicial, se intenta refinarla mediante métodos de optimización. Se puede aplicar la homografía para transformar una de las imágenes y alinearla con la otra. Esto implica proyectar los puntos de una imagen a la otra utilizando la homografía estimada. Esto es parte de lo que llamamos *warping*, ver Figura 4.11.

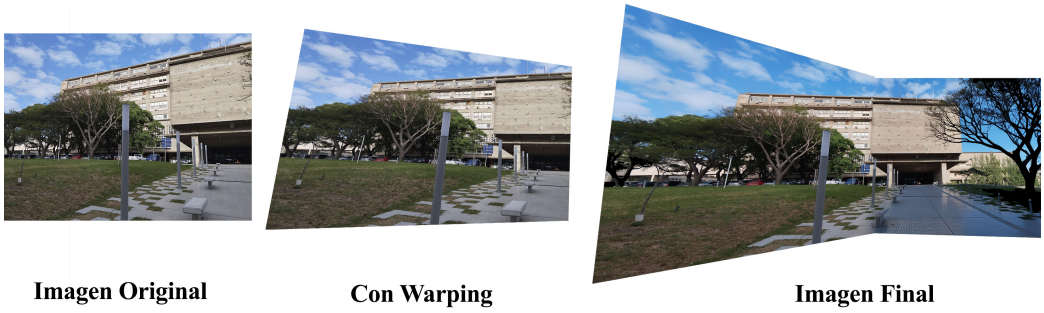

Figura 4.11: Ejemplo del proceso de *warping* plano en fotografía del edificio de Facultad de Ingeniería, UdelaR.

El *warping* implica aplicar una transformación geométrica a una imagen para mapear sus píxeles desde una ubicación en la imagen de origen a su ubicación correspondiente en la imagen de destino. Esta transformación puede incluir rotación, escalado, traslación y, en algunos casos, deformación no lineal, como las causadas por la distorsión de la lente. Siguiendo con el ejemplo anterior, no solo debemos tener en cuenta la homografía estimada entre dos imágenes, sino la imagen final que queremos lograr. Por ejemplo, podríamos querer una imagen final con proyección plana, esférica, cilíndrica, etc. El proceso de *warping* se encargará de transformar las imágenes de tal manera que se proyecte de la forma deseada, teniendo en cuenta las homografías estimadas. Ver ejemplos en las Figuras 4.12 - 4.15.

Figura 4.12: Imagen resultante con *warping* plano.

Figura 4.13: Imagen resultante con *warping* cilíndrico.

Figura 4.14: Imagen resultante con *warping* esférico.

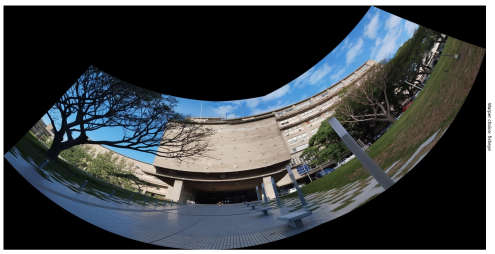

Figura 4.15: Imagen resultante con *warping* ojo de pez.

4.1.3. *Calibration & Blending*

Cuando se tienen las imágenes con su proyección final, hace falta un proceso de calibración donde se estimen ciertos parámetros de cámara para corregir una variedad de defectos. Por ejemplo, las diferencias de exposición, el tipo de lente usado, la distorsión y las aberraciones, son algunos de los efectos que contribuyen a que la imagen final no parezca pertenecer a una única toma. Es decir, se trata de llevar cada imagen utilizada a una configuración "ideal" de cámara para lograr una imagen homogénea. Posteriormente, se alinean y se realiza la composición. Las imágenes rectificadas se alinean de tal manera que parecen forman un único plano de la escena. Sin embargo, este proceso de alineación por sí solo dejaría visibles costuras en la panorámica. Para eliminarlas, es necesario un paso más que forma parte del fusionado o *blending*. Promediar directamente los píxeles superpuestos da como resultado "efectos fantasma" (*ghosting*): las regiones de costura de la imagen final se manifiestan como áreas borrosas o con efecto similar a la doble exposición. La razón detrás de esto es que, al promediar los píxeles, se pueden combinar características de diferentes partes de la escena que no están perfectamente alineadas. Para evitar esto, se utilizan técnicas de *blending* más sofisticadas que consideran la alineación precisa de las imágenes superpuestas y aplican métodos de fusión que preservan mejor los detalles y las características de la escena. Las técnicas de *blending* pueden incluir la ponderación de píxeles basada en la contribución relativa de cada imagen, la interpolación para suavizar las transiciones entre las imágenes superpuestas y la optimización de corte de grafos para encontrar límites óptimos entre las regiones superpuestas. Una técnica común es el *Feathering* o *Alpha Blending*: un valor de transparencia (α) para cada píxel en las imágenes superpuestas. La contribución de cada imagen a la imagen compuesta se pondera según el valor α, lo que permite un mezclado suave y controlable, ya que este puede cambiar para distintas coordenadas de la imagen:

$$\mathbf{I}_{\text{blend}} = \alpha\mathbf{I}_1 + (1-\alpha)\mathbf{I}_2. \tag{4.15}$$

Una mejor opción es realizar un *blending* recortando las imágenes de tal manera que se minimice el error de solapamiento (queremos hacer el corte entre dos regiones superpuestas en un camino de píxeles donde las dos imágenes coinciden mejor, es

decir, donde la el error de superposición es bajo). La trayectoria de coste mínimo a través de la superficie de error se calcula como sigue. Si $\boldsymbol{B}_1$ y $\boldsymbol{B}_2$ son dos bloques que se solapan a lo largo de su borde vertical con las regiones de solapamiento B_1^s y B_2^s, respectivamente, entonces podemos definir la superficie de error como $\boldsymbol{e} = (B_1^s - B_2^s)^2$. Para hallar el corte vertical mínimo a través de esta superficie, iteramos sobre e $(i = 2...N)$ y calculamos el error mínimo acumulado E, para todas las trayectorias:

$$E_{ij} = e_{i,j} + \min(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1}). \tag{4.16}$$

Al final, el final de la trayectoria vertical mínima a través de la superficie será indicado por el valor mínimo de la última fila de E y se puede, así, rastrear y encontrar el camino del mejor corte. Un procedimiento similar puede aplicarse a los solapamientos horizontales. Cuando hay un solapamiento vertical y horizontal, las trayectorias mínimas se encuentran en el centro y el mínimo global se elige para el corte [45].

Para ejemplificar esto podemos, de manera análoga al artículo citado, tomar una imagen de muestra para generar una textura. Tomando la textura de la fachada de la Facultad de Ingeniería, para continuar con el mismo ejemplo que se viene desarrollando, se puede crear una imagen más grande a partir de parches o retazos (*Image Quilting*). En la Figura 4.16 presentamos los resultados de este pequeño proceso por tres métodos distintos:

- Aleatoria: seleccionar aleatoriamente parches de una textura y juntarlos.
- Minimizando SSD: muestrear los parches y colocarlos en patrones superpuestos de forma que el parche muestreado tenga un SSD relativamente pequeño con los parches anteriores (izquierdo o superior) en la región superpuesta.
- Minimizando el error de solapamiento: seguir la ruta SSD mínima en la superficie de error de la región de solapamiento (se usó algoritmo *Seam Carving* [46]) y tratar esta ruta óptima como el límite de la región de solapamiento.

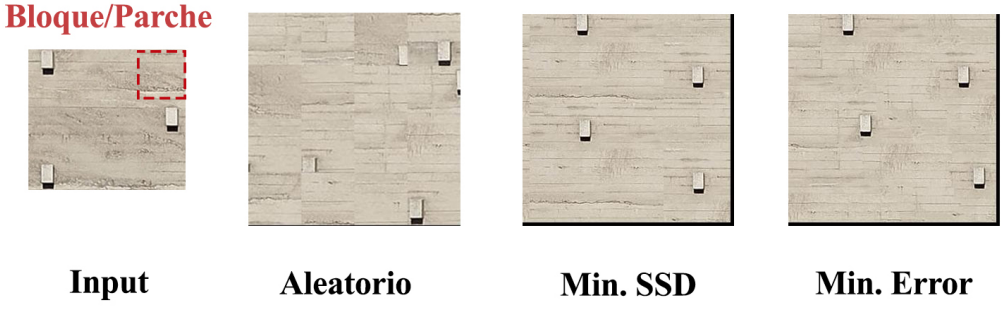

Figura 4.16: Ejemplo de *Image Quilting* con foto de edificio de Facultad de Ingeniería, UdelaR.

Se puede observar que con el último método se eliminan las repeticiones de bordes e imágenes que se ven solamente minimizando SSD, y se logra un fusionado mucho más suave. Lo mismo se practica en el plano general de la Facultad durante el *stitching*, logrando hacer invisible las costuras. En la Figura 4.17 se marcan las costuras, que corresponden al camino óptimo para el error de solapamiento.

Figura 4.17: Costuras marcadas en *stitching* con fotografías del edificio de Facultad de Ingeniería, UdelaR.

4.2. Imaginería de Muestra Completa

En medicina, en muchos estudios clínicos, incluidos los relacionados con el cáncer, es muy deseable adquirir una imagen del tejido completo a la vez que se conserva una resolución microscópica. Una forma de lograrlo es creando una imagen compuesta, superponiendo imágenes individuales adquiridas a gran aumento con un microscopio [47].

El ejemplo de la imagen panorámica del edificio descrito en la sección anterior es fácilmente extrapolable al ámbito de imaginería médica, y más aún a la microscopía. Lograr ensamblar un mosaico a partir de varias tomas que sea fiel a las imágenes originales reviste una importancia trascendental en cualquiera de sus posibles aplicaciones. Esto es lo que proponemos en este trabajo, a partir de las imágenes obtenidas con el sistema descrito en los Capítulos 2 y 3.

Importamos la librería de código abierto 'Open Stitching' [48], basada en el módulo de *stitching* de OpenCV. Este paquete proporciona funciones de utilidad para analizar en profundidad lo que sucede detrás de la costura (*stitching*) de las imágenes procesadas. A partir de esta librería y luego de explorar sus posibilidades, construimos funciones correspondientes a cada paso del algoritmo de

stitching, pudiendo trabajar de manera eficiente con nuestras muestras. A un nivel muy general y de manera robusta, el funcionamiento de este método se podría descomponer en pasos similares a los mencionados anteriormente. Estos son: buscar características relevantes (*features*) en cada imagen; detectar los mismos *features* en dos imágenes, comparándolas una a una (*pairwise*); generar un conjunto de imágenes relevantes y modificarlas (*warp*, *crop*, etc.) para fusionarlas en un plano final. En la Figura 4.18 vemos un diagrama completo del procesamiento. Muchos de los pasos incluidos estarán presentes o no dependiendo de la configuración de parámetros que usemos al realizar el *stitching*.

A partir de las funciones creadas, logrando un código modularizado, se dividió el *stitching* en dos grandes funciones. La función `registration()` preprocesa las imágenes y "entrena" al algoritmo de *stitching*. La función `compose_panoramic()` ejecuta el algoritmo entrenado sobre el conjunto de imágenes de entrada para lograr la imagen final. Si lo comparamos con el diagrama de la Figura 4.18, la función `registration()` incluye todo el cambio de tamaño (*resizing*) de imágenes y toda la etapa etiquetada como *Registration*. Por otro lado, `compose_panoramic()` encierra la parte que se ve etiquetada como *Composting*. Un hecho importante que nos permitió avanzar a una velocidad significativamente mayor a la que teníamos al principio fue haber logrado guardar los parámetros de cámara estimados (`cv2.detail.CameraParams`: transformaciones, exposición, color, etc.) por el algoritmo durante el entrenamiento. Para esto desarrollamos una función que crea un diccionario conteniendo los parámetros y los guarda en un archivo .pkl; y una función que carga este tipo de archivos y carga los parámetros para ser usados nuevamente. En el Apéndice C presentamos una descripción de estas funciones para mayor claridad.

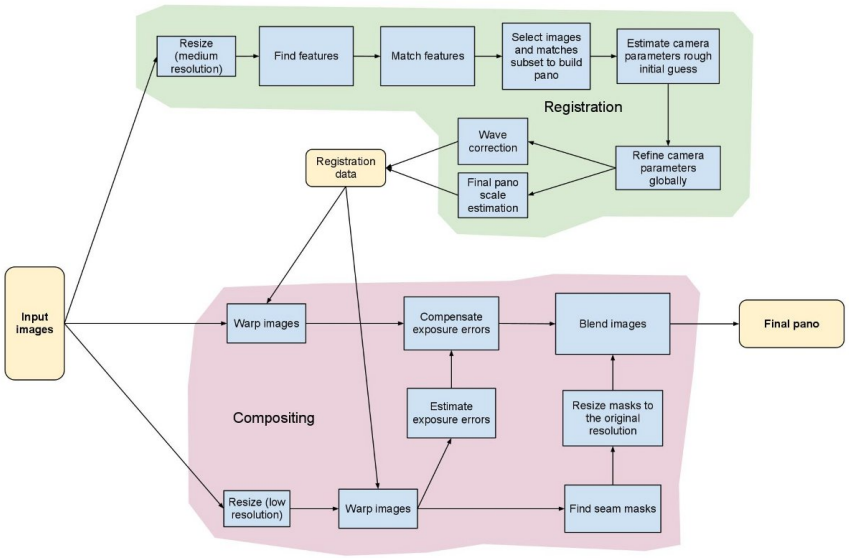

Figura 4.18: Flujo de trabajo del *Pipeline* de *stitching*. Documentación de OpenCV [49].

Tomemos como ejemplo la muestra de hoja de pino escaneada utilizando el algoritmo descrito en la Sección 3.4. El procedimiento será similar al descrito anteriormente, por lo que solo se detallarán los aspectos relevantes a nuestro caso particular. Partimos de un conjunto de fotografías que cubre completamente la muestra que queremos capturar de forma completa, con cierto solapamiento entre fotografías. Los detalles de la adquisición de imagen del sistema pueden verse en la Figura 4.19. Aquí es relevante destacar que lo crucial es que exista solapamiento y que el aumento de redundancia (misma región en más de dos fotos) no se traduce necesariamente en un mejor *stitching*.

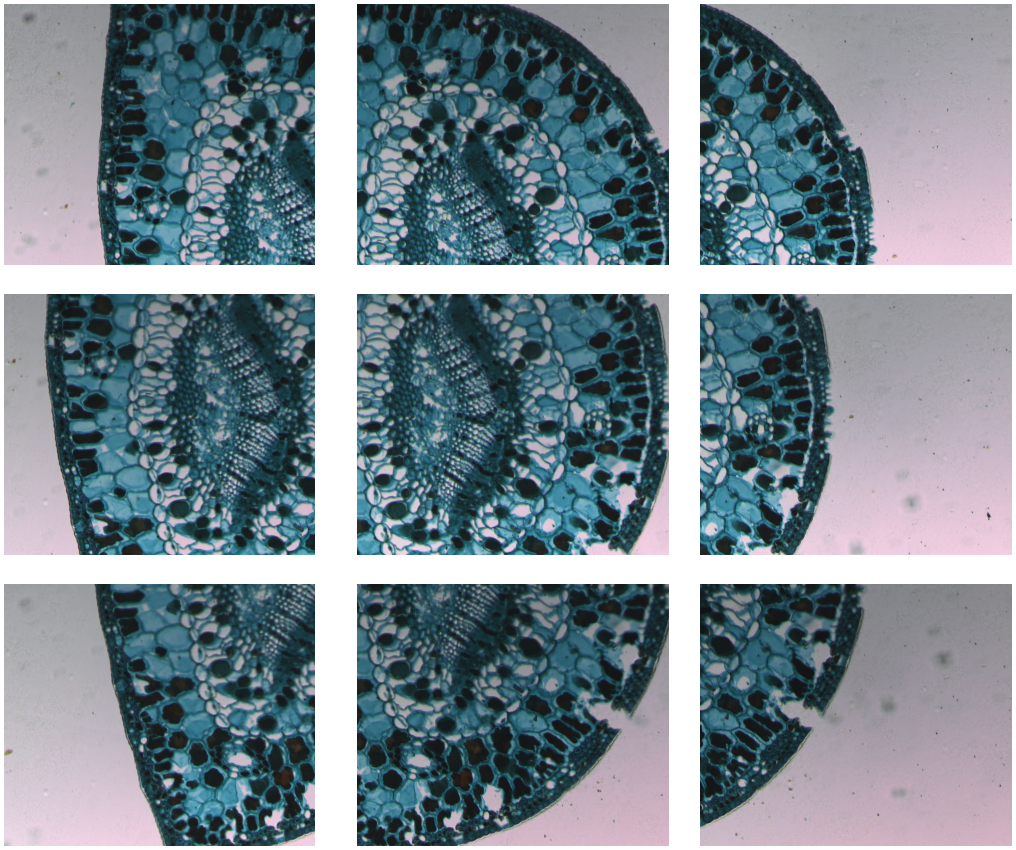

Figura 4.19: Muestra de fotografías de entrada al algoritmo de *stitching*.

Idealmente, nuestro conjunto de fotografías de entrada incluye toda la muestra a explorar y nada más. Es decir, es deseable que dentro de nuestras imágenes encontremos solamente porciones del tejido de interés. Entonces, es sensato configurar nuestro sistema para que cubra toda la zona de la muestra, o bien hacer una selección de imágenes antes de ingresarlas al algoritmo de *stitching*. Teniendo una lista de imágenes, ejecutamos la función de detección de características a cada imagen. Se genera, entonces, un conjunto de características y descriptores para ser comparados en la etapa de *Feature Matching*. En las Figuras 4.20 - 4.21 vemos las *features* calculadas en dos secciones de tejido.

Se buscan las mejores coincidencias entre las características con descriptores de cada par de imágenes. Luego, se decide cuáles de las imágenes son "buenas" (forman parte de la panorámica) a partir de las coincidencias (*matches*) utilizando un umbral de confianza `confidence_threshold` especificado por nosotros. Es decir, se crea un subconjunto conteniendo solo las imágenes relevantes. La clase que utilizamos es la llamada `Subsetter()`. Ver un ejemplo en la Figura 4.22.

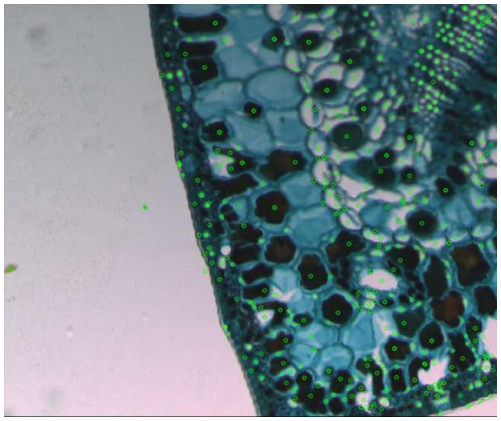

Figura 4.20: Imagen resultante con *warping* plano.

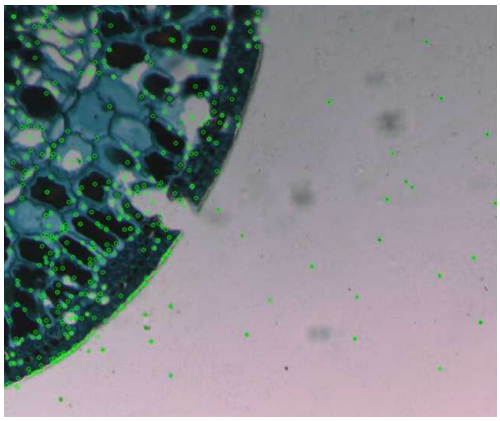

Figura 4.21: Imagen resultante con *warping* cilíndrico.

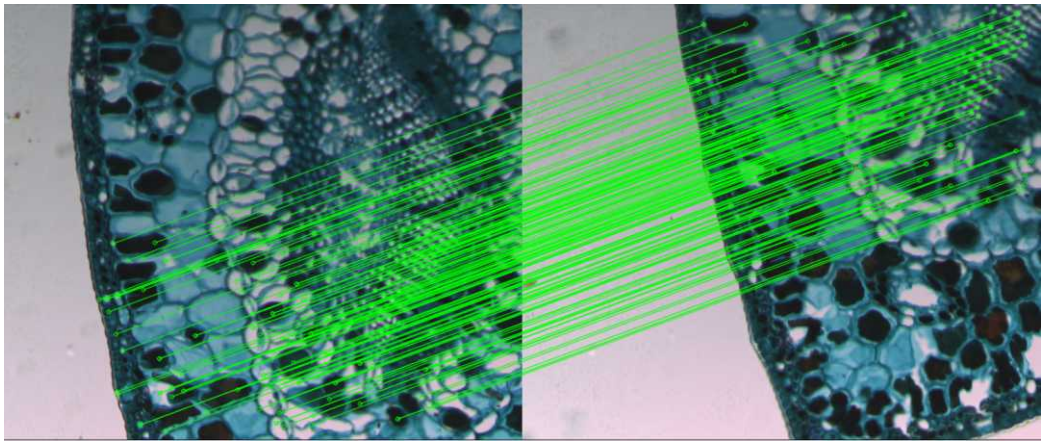

Figura 4.22: Coincidencias de *features* detectadas con *Feature Matching* en un par de imágenes.

A partir de este punto, si se tienen dos o más imágenes relevantes (y, por tanto, con coincidencias encontradas), se estiman las transformaciones de perspectiva (cámaras) entre cada par de imágenes, terminando así la etapa de Registro. Las cámaras e imágenes pasan a la etapa de Composición, donde, según el tipo de *warper* elegido, se utilizan para crear el panorama final. Luego de aplicar el *warping*, se compensa y estima el error de exposición para lograr que la iluminación sea pareja a lo largo de todo el conjunto de imágenes. Por último, se forman las máscaras

de las costuras (*seam masks*). Si se desea, podemos recortar el conjunto de imágenes utilizando una máscara tentativa del panorama (imagen final). Para obtener un panorama sin regiones negras, podemos estimar el mayor rectángulo interior incluido dentro del mismo y recortar cada una de las imágenes individualmente. Para esto usamos la clase llamada `Cropper()`. Ver Figura 4.23.

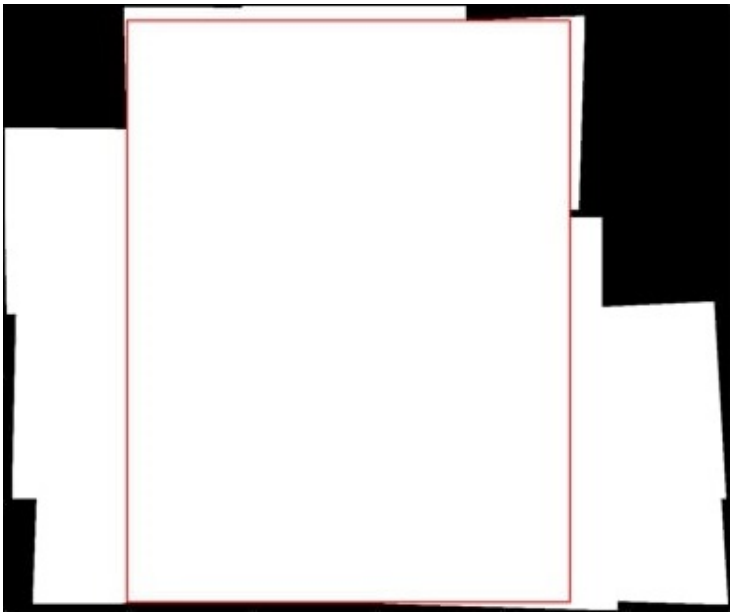

Figura 4.23: Rectángulo de mayor área incluido dentro del conjunto de imágenes.

Para que el recorte no elimine información relevante (porciones de la muestra), el conjunto de imágenes de *stitching* debe proporcionar una región de fondo alrededor de la muestra no menor al rectángulo de área igual al producto del máximo ancho de la muestra (medido desde el eje vertical) y el máximo alto (medido desde el eje horizontal). Esto puede ser calculado o "calibrado". Por un lado, podemos medir manualmente y configurar el sistema de microscopía para aproximar este rectángulo modificando la cantidad de pasos. Por otro lado, podemos asegurarnos de cubrir un área mucho mayor a la muestra, puesto que el sistema lo permite para el tamaño de muestras usadas, cumpliendo así el objetivo sin necesidad de realizar dichos cálculos. Las máscaras de costura encuentran una línea de transición entre imágenes con la menor cantidad de interferencia. La clase utilizada se llama `SeamFinder()` y tiene la función de minimizar el error de solapamiento, como se aprecia en la Sección 4.1.3. Las costuras se obtienen en las imágenes deformadas de baja resolución y luego se redimensionan a las imágenes deformadas de resolución final. Las máscaras pueden utilizarse en el paso de fusionado para especificar cómo deben componerse las imágenes.

Con el procesamiento descrito, las imágenes pueden ser mezcladas para formar un panorama completo usando la clase `Blender()`. Esta fusión puede convertirse en líneas o ponderarse sobre el panorama resultante. Además, el panorama final lo podemos realizar usando las imágenes con o sin recortes. Ver Figuras 4.24 - 4.27.

Si bien la panorámica lograda cuenta ya con la suficiente calidad para trabajar sobre ella, por ejemplo, a través de cálculos estadísticos (desviación, promedio de intensidad, etc.) o algoritmos de detección de patrones, se puede añadir un paso de postprocesamiento en donde se remueve el fondo y se mejora la nitidez. Esto fue lo que hicimos para lograr la imagen desplegada en la Figura 3.21.

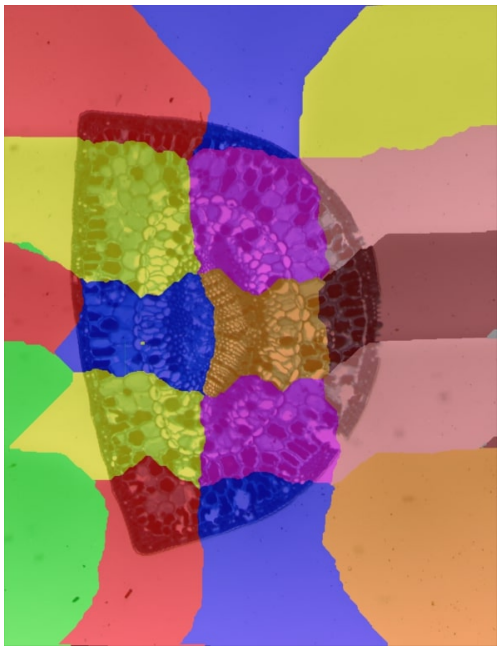

Figura 4.24: Máscara de costuras.

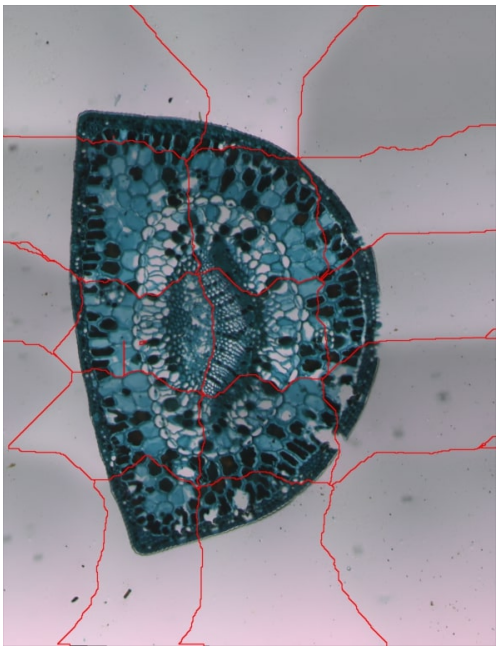

Figura 4.25: Costuras.

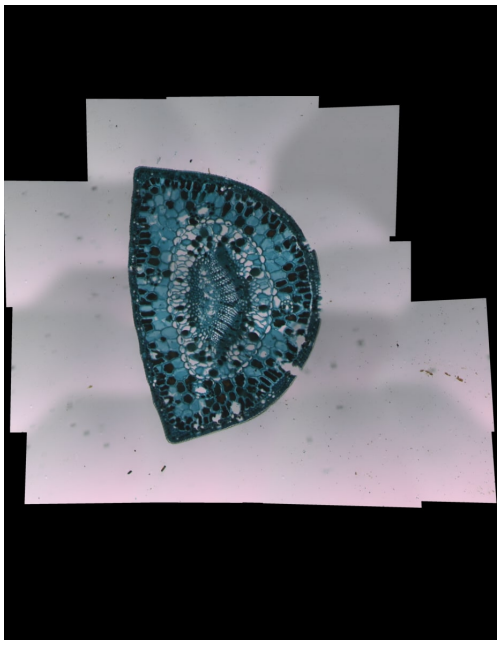

Figura 4.26: Panorámica sin recortes.

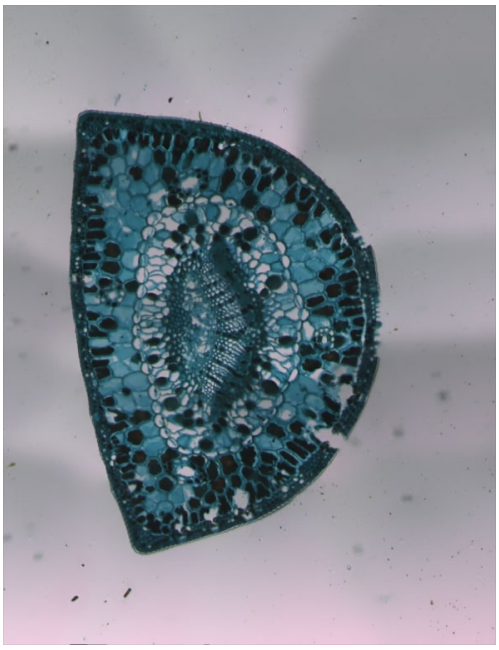

Figura 4.27: Panorámica final.

Este es el algoritmo de *stitching* implementado y aplicado a todas nuestras muestras. De esta manera, contamos con un sistema que, en primer lugar, puede escanear automáticamente una muestra microscópica y, en segundo lugar, generar una imagen completa de la muestra al ejecutar un único *script* de Python. Esto fue fundamental para agilizar el proceso de generación de datos para los posteriores procesamientos y análisis realizados.

4.2.1. Calidad del *Stitching*

El objetivo del *stitching* es componer una imagen nítida, completa y sin costuras visibles. Por más claro que suene esta primera oración, realizar una evaluación criteriosa y certera de la calidad del resultado de un *stitching* de cierto conjunto de imágenes no es una tarea trivial. Garantizar la precisión, la coherencia y la fiabilidad de los mosaicos de imágenes generados significaría, en nuestro caso, la seguridad de que la muestra completa de tejido puede ser estudiada en reemplazo de analizar porción por porción bajo el microscopio. Esta evaluación implica analizar diversos aspectos de la imagen final para determinar si cumple con los estándares de calidad deseados y si es adecuado para su uso previsto.

Durante la evaluación de la calidad del *stitching*, se pueden considerar varios aspectos importantes, que incluyen, entre otros, la precisión geométrica, la consistencia fotométrica, la integridad visual y la ausencia de artefactos o defectos. La precisión geométrica se refiere a la alineación precisa de las características visuales y los contornos entre las imágenes individuales en el mosaico final. La consistencia fotométrica se relaciona con la uniformidad en la iluminación, el color y el contraste entre las imágenes, garantizando una apariencia visual coherente en todo el mosaico. La integridad visual se refiere a la ausencia de discontinuidades, distorsiones o artefactos visuales que puedan afectar la percepción o interpretación de la imagen completa. Además, es importante evaluar la presencia de artefactos de *stitching*, como bordes visibles, superposiciones incorrectas o deformaciones geométricas.

Para llevar a cabo una evaluación cuantitativa del *stitching*, se pueden utilizar diversas métricas y métodos de análisis, que intenten medir la presencia de los errores mencionados, así como la presencia de las cualidades deseadas.

Errores comunes

Como ya mencionamos, la calidad del resultado se expresa, por lo tanto, mediante la medición de la visibilidad de la costura entre las imágenes ensambladas, y de la correspondencia entre las imágenes ensambladas adyacentes que forman la imagen compuesta [9]. En otras palabras, no deberían ser notables discontinuidades en términos de objetos (aparición de artefactos), luz, perspectiva y otras características de la imagen resultante; así como tampoco debe haber artefactos extraños o con enfoques distintos al que corresponde.

La lista de posibles errores puede ser extensa, ejemplificaremos los más relevantes respecto al problema que estamos abarcando; los que pueden con mayor probabilidad perjudicar un mosaico de imágenes de microscopía. Los defectos que llamaremos *discontinuidades* son causadas por un Registro fallido de las imágenes

de entrada, cuando el conjunto de imágenes que tenemos como entrada no cuenta con la información necesaria para realizar un *stitching* correcto. Otra posible causa de este tipo de errores es el uso de un método de registro inadecuado, como puede ocurrir si el paso de Registro del algoritmo de *stitching* es incapaz de corregir los cambios de perspectiva de las imágenes de origen, causando límites notables entre imágenes adyacentes.

Figura 4.28: Errores de alineación y *ghosting* en un *stitching* de la fachada de la Facultad de Ingeniería, UdelaR.

El *desenfoque* es un defecto común en la obtención de imágenes y puede estar causado por el dispositivo de captura de imagen o por causas extrínsecas, como el movimiento del dispositivo al capturar. Constituye un error del *stitching* cuando se genera un desenfoque en la imagen final que no tiene correspondencia en las imágenes de entrada, debiéndose normalmente a una mezcla (*blending*) inadecuada del conjunto de entrada. Minimizar o eliminar el movimiento del microscopio y la muestra durante el proceso de adquisición de imágenes puede reducir significativamente los artefactos de movimiento en las imágenes individuales y el mosaico final.

El recorte de objetos (*object clipping*) se produce cuando un objeto cambia de posición o forma en el FoV de la cámara entre las capturas de imagen. Este fenómeno suele ir acompañado de la aparición de artefactos fantasma (*ghosting*) que responde usualmente a las mismas causas. En nuestro caso no debemos preocuparnos con la aparición o movimiento de objetos, ya que tratamos con tejido inerte. Los cambios de intensidad resultan del proceso de mezcla de los fotogramas de entrada, cuando el algoritmo de *blending* equilibra los brillos de las distintas imágenes para mitigar las diferencias de luz existentes.

En la Figura 4.27 podemos ver claramente cómo se manifiesta este error, que si bien es un problema desde el punto de vista estético y de claridad en la muestra completa, no representa conflictos con el objetivo de este trabajo, ya que las Matrices de Mueller normalizadas son independientes de la intensidad, como vimos en el Capítulo 1. De todos modos, un procesamiento posterior podría mitigar este efecto, y es posible que una configuración distinta de la iluminación del microscopio y del ambiente resulte en una intensidad más uniforme en el resultado final.

Quizás uno de los errores más relevantes sean los errores de alineación, cuando la alineación incorrecta de las imágenes individuales puede conducir a un mosaico desalineado o distorsionado, lo que afecta la precisión de la representación del tejido biológico. Para esto es fundamental contar con un buen solapamiento en el conjunto de imágenes de entrada. Podemos ver un ejemplo en la parte izquierda Figura 4.28.

Evaluación

Los problemas listados en la sección anterior son, en su mayoría, fácilmente detectables a simple vista. Una evaluación subjetiva de los resultados del *stitching* para WSI que se han mostrado en este documento podría ser positiva, pero es necesario dar cuenta de la importancia de conseguir no solo una fotografía coherente y sin errores a simple vista, sino que respete las propiedades de las imágenes de entrada para representar fielmente el tejido. Un abordaje a una evaluación de calidad subjetiva y objetiva de *stitching* panorámico es presentado en la literatura [50]. En nuestro caso, parece ser más relevante asegurarnos de no modificar la información de entrada, ya que parte del trabajo fue generar una buena base de imágenes de cada muestra, y realizar el *stitching* correspondiente asegurando que no había artefactos visibles (*ghosting*, *blur*, *misalignmet*) provocados por errores en la imagen final. Las deformaciones leves de la estructura del tejido pueden no ser detectables a simple vista, pero podrían revestir problemas al realizar un estudio más detallado de la imagen en busca de características que revelen información patológica de la muestra. Siguiendo este razonamiento, una de las primeras formas de evaluar el *stitching* que deben tenerse en cuenta es la similitud del resultado con las imágenes de entrada.

Obtener un valor cuantitativo de la calidad de la costura es un reto que ya ha sido estudiado [51][52][53][54]. En varias ocasiones, se opta por estimar la calidad estructural del *stitching* y su calidad fotométrica. Algunas de las métricas usuales para representar la calidad estructural de una imagen resultante de *stitching* son SSIM (ver Sección 4.2.1) y MSSIM (Min. SSIM), mientras que índices como SAM (*Spectral Angle Mapper*), RMSE (ver Sección 4.2.1), PSNR (ver Sección 4.2.1) y el IMR (*Intensity Magnitude Ratio*) son utilizados para representar la calidad fotométrica de la imagen resultante. Sin embargo, muchos de estos estudios se centran en la aplicación del *stitching* para fotos panorámicas, generalmente centrando sus análisis en fotografías de paisajes o edificios, por lo que en nuestro caso debemos elegir con cuidado cuáles de estas métricas nos aportan información realmente relevante para la aplicación que tratamos.

La metodología de evaluación propuesta para nuestro algoritmo de *stitching* es partir de una imagen base como *Ground Truth*; dividirla en varios rectángulos con solapamiento de manera análoga al proceso que lleva a cabo el sistema de microscopía; usar estas imágenes como entrada en el algoritmo de *stitching* y luego comparar la imagen resultante con el *Ground Truth* utilizando algunos de los indicadores mencionados. Se detallan las métricas consideradas:

- **SSIM** : El índice de similitud estructural es un método para predecir la calidad percibida de las imágenes, es comúnmente usado para medir la similitud entre dos imágenes, ya que es una métrica de referencia completa, por lo que se basa en una imagen inicial como referencia para realizar el cálculo.

  El SSIM se diferencia del resto de las métricas principalmente por ser un modelo basado en la percepción humana que considera la degradación de la imagen como un cambio percibido en la información estructural. La medición de la luminancia se considera cualitativamente en coherencia con la Ley de Weber, que indica como el cambio perceptible en la luminancia es aproximadamente proporcional a la luminancia del fondo. Del mismo modo, la medición del contraste también es coherente con el sistema visual humano. La distinción respecto a otras técnicas como MSE o PSNR radica en que estos métodos calculan errores absolutos. El concepto de información estructural se basa en la premisa de que los píxeles, especialmente cuando están cercanos espacialmente, presentan dependencias significativas entre sí. Se calcula entre ventanas de las imágenes fórmula para calcular el SSIM entre dos ventanas x e y de tamaño $N \times N$ es:

  $$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)\,(2\sigma_{xy} + c_2)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)}, \tag{4.17}$$

  donde:

  - μ_x, μ_y son la media muestral de los píxeles de x e y, respectivamente.
  - σ_x^2 , σ_y^2 son la varianza de x e y, respectivamente.
  - σ_{xy} es la covarianza x e y.
  - $c_1 = (k_1 L)^2 , c_2 = (k_2 L)^2$ son dos variables para estabilizar la división con denominador débil ($k_1 = 0{,}01$ y $k_2 = 0{,}03$ por defecto).
  - L es el rango dinámico de los valores de píxel ($2^{\#\text{ bits por píxel}} - 1$).

    El valor máximo posible se da cuando se calcula entre dos imágenes idénticas, y es igual a 1. La SSIM varía según el canal de color porque considera la similitud estructural en cada componente de color de las imágenes. Cada canal de color puede representar diferentes características y detalles de la imagen, y la similitud estructural puede variar entre ellos. En imágenes tomadas por fotografía, como es el caso del *Ground Truth*, es esperable que haya algún grado de aberración cromática, ya que es posible que el MO no enfoque de igual manera

todos los colores en cada punto, lo que podría explicar esta variación en nuestros resultados.

En una imagen que contiene una gran cantidad de detalles verdes, es posible que el canal verde tenga una similitud estructural más alta que los otros canales. Por lo tanto, al calcular la SSIM para cada canal de color por separado, se puede obtener una evaluación más precisa de la similitud entre las imágenes en términos de su contenido y estructura específicos de color.

En nuestra evaluación, calculamos el índice para cada canal y luego lo promediamos. Entre dos imágenes separadas por canales tenemos: R_1, G_1, B_1 y R_2, G_2, B_2; se calcula:

$$\mathrm{SSIM}_r\left(R_1, R_2\right), \mathrm{SSIM}_g\left(G_1, G_2\right), \mathrm{SSIM}_b\left(B_1, B_2\right),$$

y luego:

$$SSIM_{rgb} = w_r SSIM_r + w_g SSIM_g + w_b SSIM_b.$$

Se elige $w_r = w_g = w_b = \frac{1}{3}$ para promediar, pero la ponderación podría ser otra si buscamos una métrica que imponga mayor relevancia a alguno de los canales.

- **RMSE** :

  El error cuadrático medio es uno de los estimadores más comunes de la métrica de medición de la calidad de imagen. Es una métrica de referencia completa y los valores más cercanos a cero indican mayor similitud. Básicamente, se toma el cuadrado de la diferencia entre cada píxel de una imagen g y el píxel correspondiente de $\hat{g}$, se suman, se dividen por el número de píxeles y se toma la raíz cuadrada. Si tenemos dos imágenes $g(x, y)$ y $\hat{g}(x, y)$:

$$\mathrm{RMSE} = \sqrt{\frac{1}{MN} \sum_{n=0}^{M} \sum_{m=1}^{N} [\hat{g}(n, m) - g(n, m)]^2}. \tag{4.18}$$

- **PSNR** :

  La Proporción Máxima de Señal a Ruido (PSNR) se utiliza para calcular la relación entre la potencia máxima posible de la señal y la potencia del ruido distorsionador que afecta a la calidad de su representación. Esta relación entre dos imágenes se mide en decibelios. La PSNR suele calcularse como el término logarítmico de la escala de decibelios debido a que las señales tienen un rango dinámico muy amplio. En la degradación de la calidad de compresión de imágenes y vídeo, el valor PSNR varía de 30 dB a 50 dB para la representación de datos de 8 bits y de 60 dB a 80 dB para datos de 16 bits. Por lo tanto, obtener valores dentro de estos rangos sería un indicador de similitud positivo.

Para calcularla se usa el MSE, que para dos imágenes monocromáticas I y K de tamaño $M \times N$ se define como:

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \|I(i,j) - K(i,j)\|^2. \tag{4.19}$$

Por lo tanto, el PSNR se calcula:

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) = 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right), \tag{4.20}$$

donde MAX_I es el máximo valor que puede tomar un píxel en la imagen. Cuando se representan usando B bits por muestra: $MAX_I = 2^B - 1$.

Se ejemplificará con la evaluación de una única imagen de microscopía tomada como *Ground Truth*. Se trata de una muestra de adenocarcinoma de próstata teñida en hematoxilina y eosina. Ambos compuestos tiñen distintos tipos de estructuras celulares en el tejido y facilitan su visualización. El núcleo celular y otras componentes nucleares se tiñen de violeta por acción de la hematoxilina. El citoplasma, el colágeno, el tejido conjuntivo y otras estructuras que rodean las células se tiñen de rosa anaranjado por acción de la eosina. Ver Figura 4.29.

Una evaluación más extensiva y de carácter estadístico es necesaria para poder formular un juicio certero sobre la calidad del algoritmo utilizado. Como se mencionó anteriormente, comenzamos dividiendo la imagen base en rectángulos solapados (ver Figura 4.30). Se realiza una grilla de 6×6 rectángulos con solapamiento de 40 %. Tomando estas imágenes como entrada para el algoritmo de *stitching*, conseguimos una imagen casi idéntica como resultado. Ver Figuras 4.31-4.32.

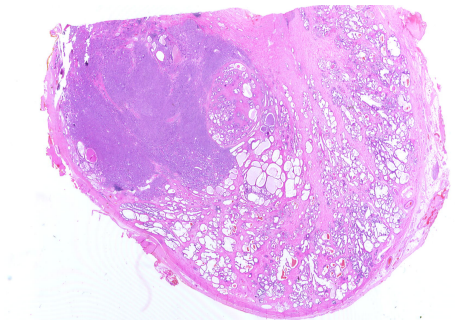

Figura 4.29: Adenocarcinoma de Próstata utilizado como *Ground Truth* para evaluación. Cortesía de Alex Brollo [55].

Figura 4.30: División de *Ground Truth*.

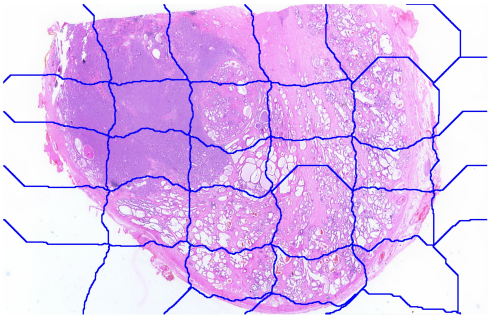

Figura 4.31: Costuras para *stitching* para la evaluación.

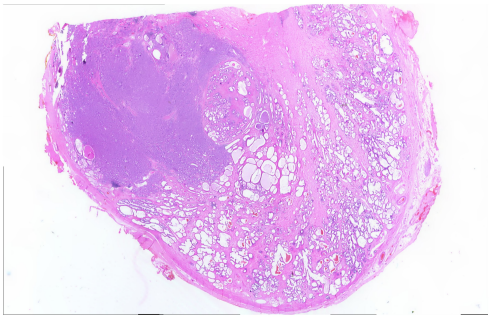

Figura 4.32: Resultado del *stitching* para la evaluación.

Es notable, como se muestra en la Figura 4.31, cómo los caminos que elige como costuras el algoritmo forman una grilla de 6×6, pero no son exactamente rectos. Esto responde a que, en cada caso, el solapamiento repite regiones de distintas características, y el *stitching* busca el mínimo error dentro de ellas. Por las mismas razones, el algoritmo no necesariamente devuelve una imagen de igual tamaño a la original, ya que, como se aprecia en la Figura 4.23, para recortar la imagen final se estima el mayor rectángulo interior, por lo que puede ser necesario ajustar el tamaño de la imagen para que sea igual al tamaño original antes de evaluar la similitud. La diferencia suele ser de unos pocos píxeles, que varían según la imagen y se subsana encontrando la ubicación del máximo valor de correlación entre la imagen original y la resultante de *stitching*, y luego recortando ambas imágenes para que tengan el mismo tamaño. Finalmente, los valores para este caso de ejemplo se presentan en la Tabla 4.1.

Métrica	**Valor**
SSIM *Mean*	0.973
SSIM *Grayscale*	0.988
SSIM *Red*	0.956
SSIM *Green*	0.981
SSIM *Blue*	0.966
PSNR	55.785 dB
RMSE	0.002

Tabla 4.1: Valores de métricas de evaluación de calidad.

Se comprueba que en general los resultados son muy favorables, indicando un alto grado de similitud entre el *Ground Truth* y la imagen obtenida por *stitching*. En los valores calculados para SSIM se comprueba la diferencia entre canales, obteniendo máximo valor en escala de grises. En la Figura 4.33 vemos la imagen separada por canales y en escala de grises.

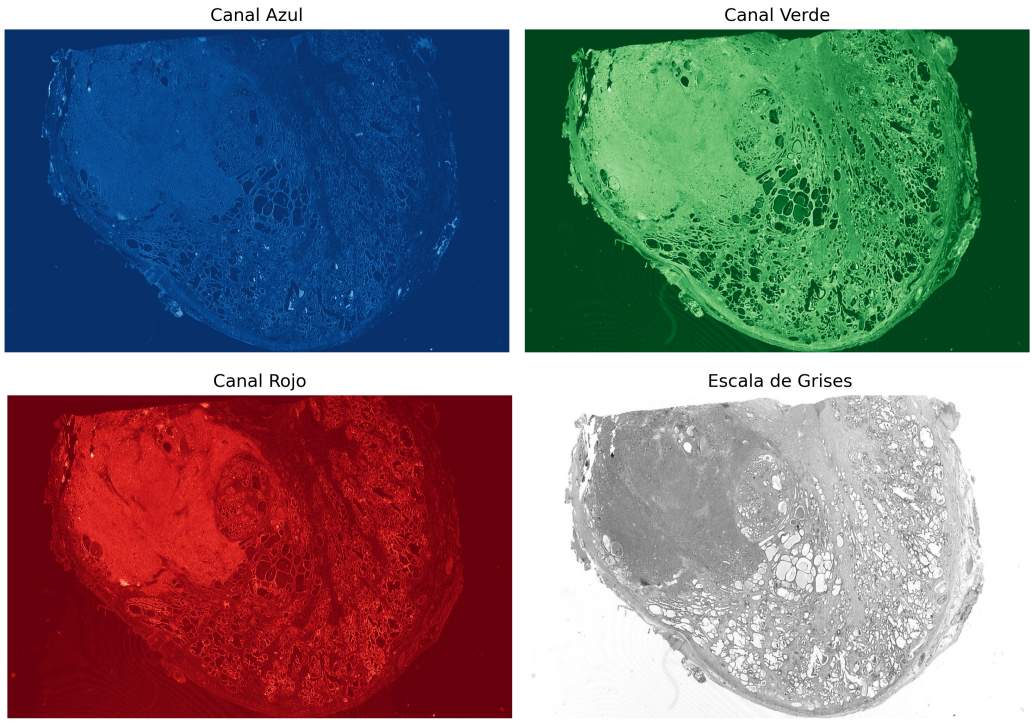

Figura 4.33: *Ground Truth* por canales.

El alto valor de PSNR indica que la calidad de la imagen reconstruida es excelente en comparación con la imagen original, por lo que se puede inferir que no hay casi distorsión. Coherentemente, el valor de RMSE es cercano a cero. Si bien estos valores son los obtenidos para un único caso, dan muestra de la capacidad del algoritmo para generar resultados de alto nivel para imágenes de microscopía.

4.2.2. Alcance del *Stitching*

Un cálculo importante para determinar el alcance del escaneo es la cantidad de fotos requeridas para realizar el *stitching* de la muestra. Puesto que el microscopio debe ser supervisado para evitar un posible desenfoque, una muestra muy grande podría significar una tarea compleja. Un factor que regula la cantidad de fotos a capturar es la magnificación efectiva, M, del sistema de microscopía. Teniendo en cuenta que el píxel de nuestro sensor es cuadrado y tiene $3{,}45\ \mu m$ de ancho (ver Sección 2.3.1), podemos hallar el área vista por la cámara, S, proyectando el tamaño del sensor sobre la muestra a través del factor de magnificación:

$$S = \frac{2048 \times 3{,}45\ \mu m}{M} \times \frac{2448 \times 3{,}45\ \mu m}{M} \simeq \frac{60}{M^2}. \tag{4.21}$$

Si suponemos que la muestra está completamente contenida dentro de un rectángulo de área A con la misma relación de aspecto que el sensor, la cantidad mínima N_{min} de fotos necesarias para cubrir todo el rectángulo es $N_{min} = \lceil A/S \rceil$.

La cantidad de fotos N_{min} es una cota inferior de la cantidad de fotos necesarias para cubrir toda la muestra. Además, no estamos teniendo en cuenta el solapamiento que debe existir entre las imágenes para realizar el *stitching*. Para calcular el área de solapamiento debemos calcular las traslaciones δ_x y δ_y de los motores X e Y descritos en el Capítulo 2. Ver Figura 4.34, en donde se aprecia un esquema del escaneo descrito.

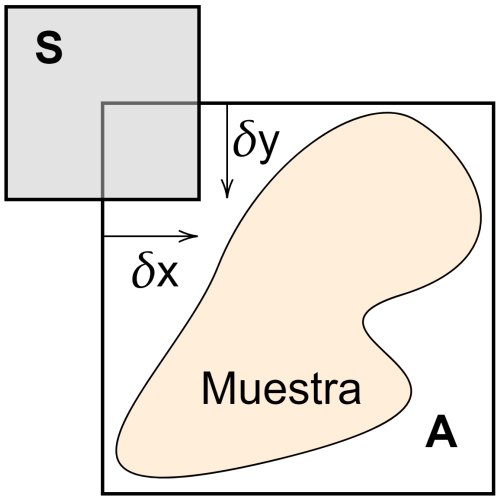

Figura 4.34: Esquema de escaneo. La muestra está completamente contenida en el rectángulo de área A. Los motores mueven la región de área S observada, un diferencial δx en la horizontal y un diferencial δy en la vertical. Realizado en Mathcha.

Primero, recordemos que las barras hiladas conectadas a ambos motores tienen un *pitch* de 1 mm. Entonces, el motor requiere 50 pasos *full-step* para una rotación completa, lo que corresponde a una traslación total de 1 mm. Además, el algoritmo que usamos para controlar ambos motores ejecuta $N = 20$ iteraciones por cada movimiento *full-step*, por lo tanto:

$$\delta_x = \delta_y = N \times \frac{1\ mm}{50} = 0{,}4\ mm. \tag{4.22}$$

Finalmente, el porcentaje de área solapada α se calcula en función del movimiento de traslación δ_x y δ_y de los motores y la magnificación M:

$$\alpha = \frac{\delta_x \delta_y}{S} \simeq \frac{\delta_x \delta_y M^2}{60}. \tag{4.23}$$

En la práctica, ajustamos el número de iteraciones N de nuestro algoritmo para que el solapamiento sea próximo al 25 %, i.e: $\alpha = \frac{1}{4}$. Entonces, necesitamos una cantidad de fotos de aproximadamente $N \simeq 4N_{min}$ para capturar la muestra completa. Si el tejido está contenido en un rectángulo de área en A mm, la cantidad de capturas N necesarias para construir la imagen de muestra completa será:

$$N = \left\lceil \frac{M^2 A}{60\alpha} \right\rceil = \left\lceil \frac{4M^2 A}{60} \right\rceil. \tag{4.24}$$

Si por ejemplo la muestra está completamente contenida en un área cuadrada $A = 5\,mm^2$ y el sistema de microscopía tiene magnificación $M = 10$, entonces necesitamos 34 fotos para construir una imagen de muestra completa. Si subimos la magnificación a $M = 20$, la cantidad de fotos se cuadriplica a un total de 136 fotos. En nuestro caso particular, un MO de magnificación $M = 10$ es una solución bastante estándar y que se adecúa bien al problema.

4.2.3. Imaginería de Mueller de Muestra Completa

Hasta ahora, hemos cubierto la teoría necesaria para el cálculo de la matriz de Mueller de una muestra con FoV limitado y su representación en una imagen. Además, utilizando el *hardware* que hemos desarrollado, podemos calcular automáticamente todas las matrices requeridas para abarcar la muestra completa, siempre que el sistema motorizado lo permita. Por otro lado, mediante la técnica de *stitching*, contamos con las herramientas necesarias para fusionar un conjunto de imágenes y construir un mosaico que capture la totalidad de la muestra.

En esta última sección, presentaremos una técnica de imaginería que combina los dos conceptos más importantes de este trabajo: la polarimetría de matrices de Mueller y la imaginería de muestra completa. La denominamos: "Polarimetría de Matrices de Mueller de Muestra Completa" (MWSI).

Un principio que fundamenta esta metodología es que el *stitching* establece una correspondencia entre los puntos clave de dos imágenes. Esto significa que una vez calculadas las transformaciones proyectivas que llevan un conjunto de imágenes al mosaico final, el valor de cada píxel en el conjunto de imágenes de entrada carece de interés. Por lo tanto, estas mismas transformaciones pueden aplicarse a cualquier otro conjunto de imágenes que comparta la misma geometría que el conjunto utilizado para entrenar el *stitching*.

Supongamos que disponemos de una muestra de tejido y dos conjuntos de imágenes, $\mathcal{A}$ y $\mathcal{B}$, que representan dos cantidades físicas distintas de todo el tejido y comparten la misma geometría. Si entrenamos el *stitching* en el conjunto $\mathcal{A}$, podemos aplicarlo al conjunto $\mathcal{B}$, obteniendo así un sistema de imaginería de muestra completa bimodal.

El segundo aspecto importante es que al capturar las matrices de Mueller de diferentes secciones de un mismo tejido, las áreas solapadas entre las imágenes deben contener la misma información. Esto se verifica, puesto que una matriz de Mueller representa propiedades mecánicas del tejido, las cuales permanecen invariantes bajo las traslaciones relativas del portamuestras.

Una vez que nos hemos convencido de estos dos últimos conceptos, ideamos una metodología para fusionar matrices de Mueller. Dado que la matriz calculada tiene nueve componentes reales, tenemos entonces nueve conjuntos de imágenes de entrada que comparten la misma geometría: $\mathcal{M}_{00}$, $\mathcal{M}_{01}$, $\cdots$, $\mathcal{M}_{22}$. La estrategia consiste en elegir un conjunto cualquiera y fusionar las imágenes para obtener un mosaico. Si el mosaico es "bueno", guardamos las transformaciones proyectivas y las aplicamos al resto de los conjuntos. Si, en cambio, el mosaico es "malo", volvemos a probar con otro conjunto y así sucesivamente.

Para asegurarnos la veracidad del procedimiento, diseñamos un *Ground Truth*. Tomamos como muestra el target utilizado en la Sección 3.2 con un polarizador a 90° colocado justo encima. Capturamos nueve matrices de Mueller localmente separadas y las fusionamos. Para asegurarnos que la información física es invariante bajo traslaciones, calculamos la polarizancia de la matriz de Mueller completa, su promedio μ y su desviación estándar σ. Los resultados del experimento fueron los siguientes:

$$\mu = 1{,}09 \quad \sigma = 0{,}05. \tag{4.25}$$

En las Figuras 4.35 y 4.36 se pueden ver los resultados. Destacamos la homogeneidad de las imágenes y cómo la propiedad física calculada no se ve alterada por los movimientos del portamuestras. En este caso podemos entrenar el algoritmo de *stitching* con las componentes: $\mathcal{M}_{00}$, $\mathcal{M}_{01}$, $\mathcal{M}_{10}$ o $\mathcal{M}_{11}$. Sin embargo, utilizar cualquiera de las otras no permitiría el entrenamiento, puesto que son muy planas y carecen de suficientes características.

En la Figura 4.38 capturamos la matriz de Mueller normalizada completa de tejido denso conectivo. Entrenamos el algoritmo de *stitching* con el conjunto de imágenes $\mathcal{M}_{11}$ y lo aplicamos al resto de los conjuntos, incluida la transmitancia del tejido (ver Figura 4.37). Este último resultado es muy interesante porque si hubiéramos intentado realizar el *stitching* únicamente con las imágenes de transmitancia, no habría sido posible debido a la escasa cantidad de características. En este caso, la birrefringencia del tejido es la que permite la fusión de la muestra. Esto abre, entonces, la posibilidad de mejorar los resultados del *stitching* de un conjunto de imágenes a partir de la utilización de matrices de Mueller, por ejemplo, buscando cuál de las componentes brinda mayor información a nivel estructural.

Por último, debemos tener especial cuidado con el enfoque. Cuando el sistema se desenfoca, podemos modelar el fenómeno mediante una convolución de los observables I_0, I_{45}, I_{90} y I_{135} con un *kernel* gaussiano de varianza σ^2:

$$I_{\theta,blur}(x,y) = I_\theta(x,y) * \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{4.26}$$

Como las componentes de la matriz de Mueller m_{ij} se pueden escribir como una combinación lineal de los observables, entonces el desenfoque del sistema las afecta de igual forma:

$$m_{ij,blur}(x,y) = m_{ij}(x,y) * \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{4.27}$$

Sin embargo, las componentes de la matriz de Mueller normalizada no verifican el mismo modelado:

$$\begin{aligned} \hat{m}_{ij,blur}(x,y) &= \frac{1}{m_{00}(x,y) * \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}} \left(m_{ij}(x,y) * \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \right) \\ &\neq \hat{m}_{ij}(x,y) * \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}. \end{aligned} \tag{4.28}$$

Entonces, las características físicas no se desenfocan en sentido gaussiano, sino que lo hacen de forma más compleja. Esto puede generar visibles variaciones espaciales en el resultado final del *stitching* de propiedades físicas.

Muestra Completa de Target con Polarizador a $90°$

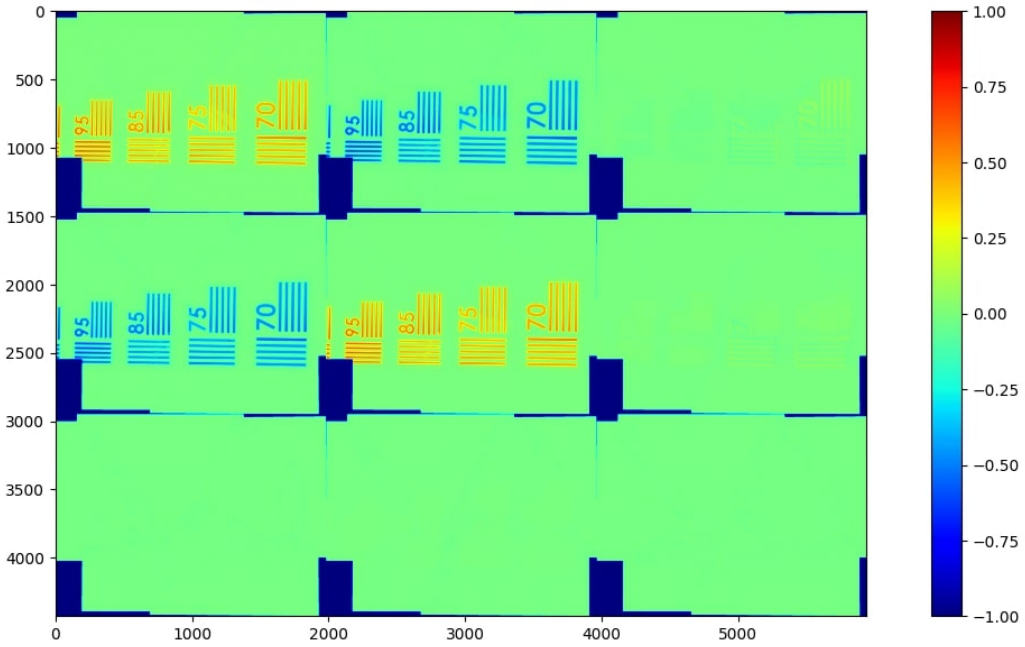

Figura 4.35: Matriz de Mueller.

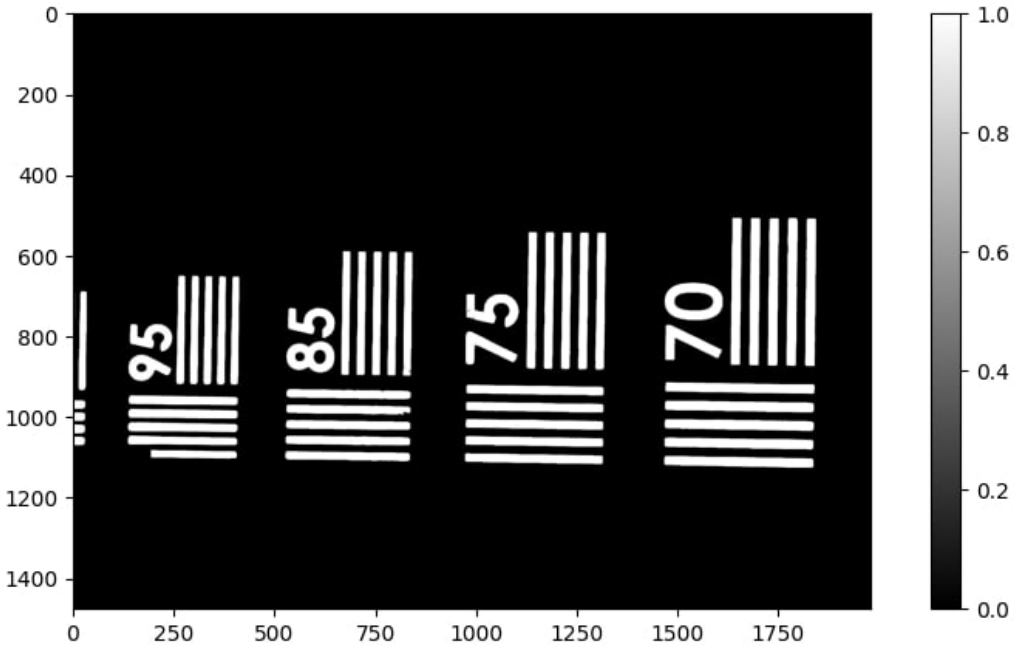

Figura 4.36: Polarizancia.

Muestra Completa de Tejido Denso Conectivo

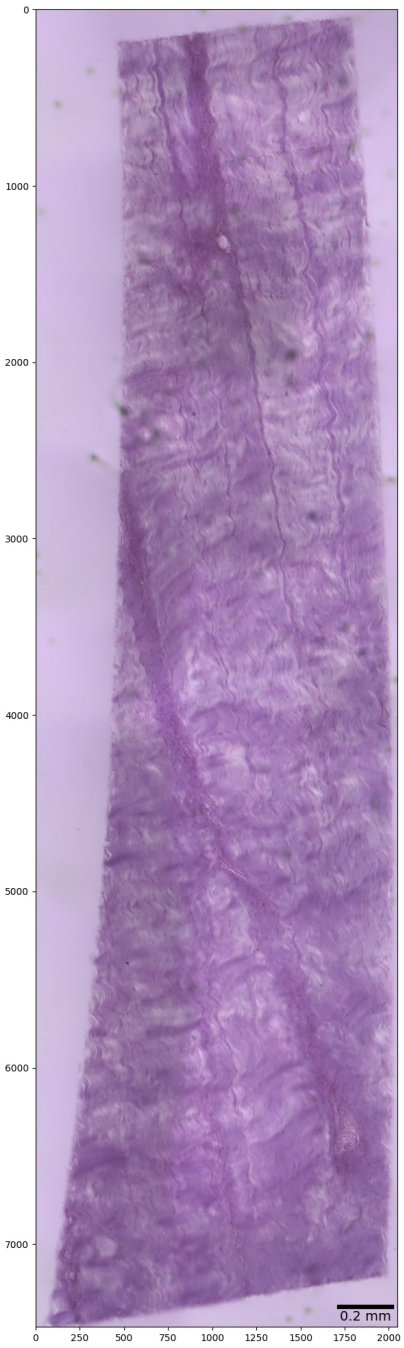

Figura 4.37: Transmitancia.

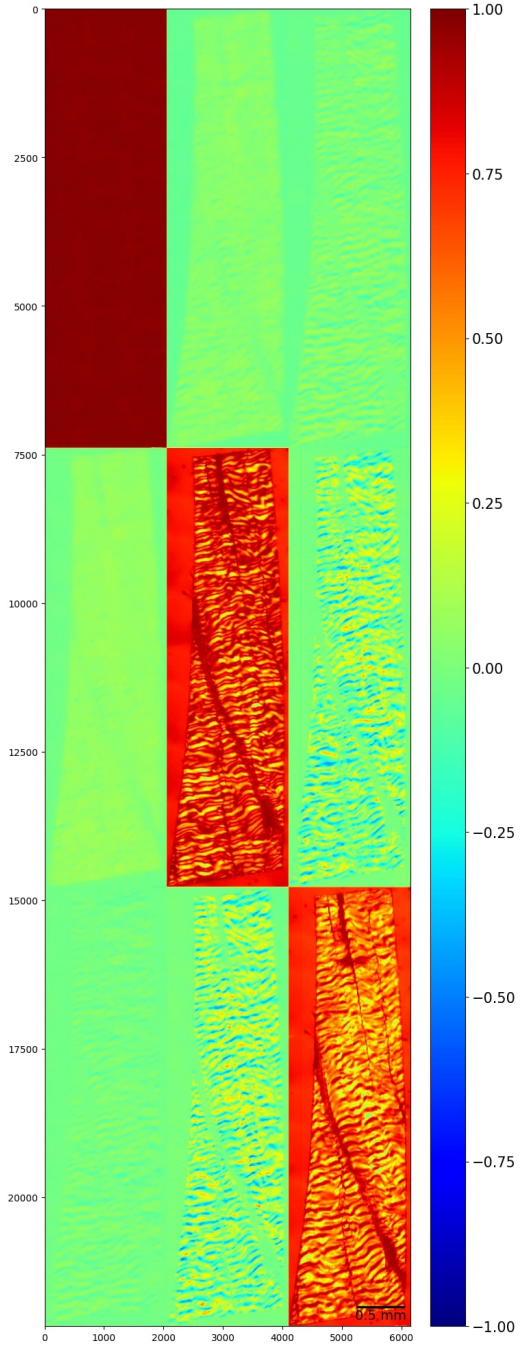

Figura 4.38: Matriz de Mueller.

Capítulo 5

Análisis de Tejido de Piel Humana

En la base de la epidermis de la piel, se encuentran los melanocitos: una clase de células cuya principal función es la producción de melanina, un pigmento que protege la dermis de la radiación ultravioleta. Casi la totalidad de la población tiene manchas en la piel producidas por una proliferación de melanocitos, mejor conocidas como lunares. Este tipo de tumor casi siempre es benigno (nevo) y suele ser inofensivo. No obstante, existe un tipo de proliferación denominado melanoma (ver Figura 5.1), un cáncer de piel que puede llegar a ser mortal. El melanoma en sus fases tempranas puede ser tratado y generalmente el paciente sobrevive. Sin embargo, cuando el cáncer alcanza la metástasis (propagación de la enfermedad a otros órganos), la tasa de supervivencia baja drásticamente. Por lo tanto, el correcto diagnóstico de la enfermedad es importante. Para esto, se cuenta con la experiencia de un dermatólogo mediante una examinación histopatológica [8].

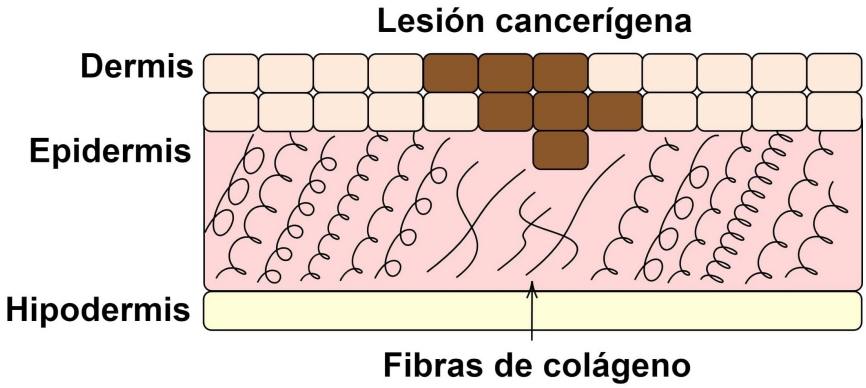

Figura 5.1: Proliferación melanocítica maligna o melanoma. La lesión cancerígena alcanza un crecimiento que logra atravesar la dermis y dañar la estructura de fibras de colágeno de la epidermis. Realizado en Mathcha.

Los principales indicadores macroscópicos para clasificar melanoma y nevo incluyen: asimetría, borde, color, diámetro y evolución (regla ABCDE) [56]. Desafortunadamente, existen características en común y la clasificación es difícil. Patológicamente, se sabe que el estado de las fibras de colágeno está asociada con

el grado de avance de la enfermedad. Por tal razón, un indicador que caracterice el estado mecánico de estas estructuras podría ser de mucha ayuda para clasificar melanoma. En un tejido sano, la fibra de colágeno está organizado en una ordenada estructura cilíndrica. Esta propiedad induce la polarización de la luz que lo atraviesa. A medida que el melanoma evoluciona, las fibras de colágeno pierden cohesión, el tejido pierde elasticidad y el núcleo de las células se expande (ver Figura 5.2). Esta transformación del tejido se ve reflejado en los cambios de sus propiedades polarimétricas: diatenuación, polarizancia, retardancia lineal y despolarizancia [57]. Como estas propiedades son independientes del brillo de la imagen, son ideales para ser utilizadas en algoritmos de reconocimiento de patrones.

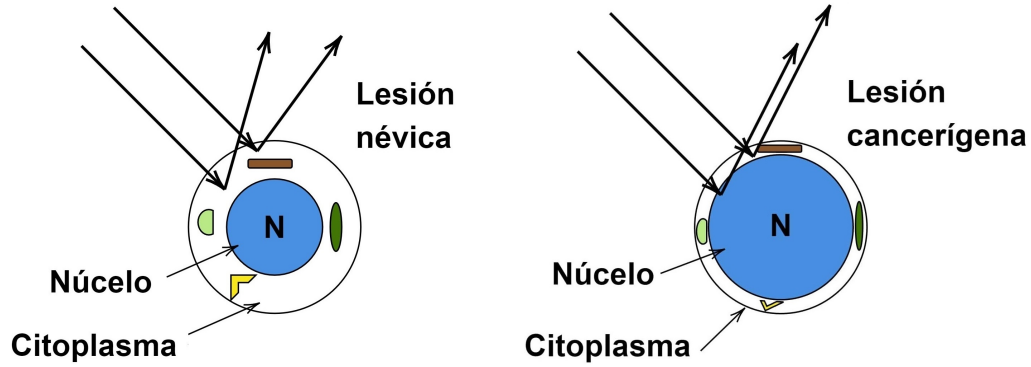

Figura 5.2: Comparación entre célula sana y cancerosa en tejido de piel humana. Cuando la célula es cancerosa, el núcleo se expande y empuja los orgánulos hacia el citoplasma. Esta transformación logra disminuir el esparcimiento de la luz incidente. Realizado en Mathcha.

En este capítulo analizaremos muestras de tejido humano, clasificadas previamente por un experto como melanoma y nevo. Generamos el conjunto de datos para medir las propiedades físicas del tejido basado en matrices de Mueller; y mediante técnicas de procesamiento de imágenes y aprendizaje automático, clasificamos patrones que podrían indicar presencia de cáncer utilizando cuatro algoritmos: *Support Vector Machine* (SVM), *Random Forest* (RF), *Multi-Layer Perceptron* (MLP) y *Convolutional Neural Network* (CNN).

5.1. Recolección de Datos

Las muestras son proporcionadas por el Instituto Pasteur de Montevideo, y fueron preparadas por la Unidad Académica de Dermatología del Hospital de Clínicas. Lesiones melanocíticas benignas y malignas de pacientes de distintas edades y sexo fueron tomadas por medio de biopsia entre los años 2015 y 2018. Las muestras tienen un espesor de 5 μm y no fueron teñidas. Elegimos cuatro muestras que constituyen una base de datos de 8 GB, distribuidas en dos categorías: dos están etiquetadas como melanoma y las otras dos como nevo (ver imágenes de muestras completas en Apéndice B).

En las Figuras 5.3 - 5.8 se aprecian la transmitancia, matriz de Mueller y polarizancia de dos secciones de tejido en el canal verde. Los tejidos fueron clasificados

como melanoma (columna izquierda) y nevo (columna derecha). Las matrices de Mueller (no normalizadas) son parecidas a la matriz identidad e indican una fuerte isotropía de las propiedades polarimétricas del tejido. Como el espesor de las muestras es fino, la señal que recibimos sobre la polarización del tejido es muy baja y difícil de medir. En particular, la birrefringencia y la despolarizancia dependen del espesor de la muestra y son casi inmensurables. Por lo tanto, nos concentraremos en la transmitancia y la polarizancia.

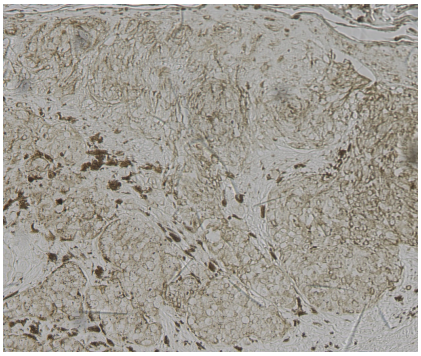

Figura 5.3: Transmitancia de melanoma.

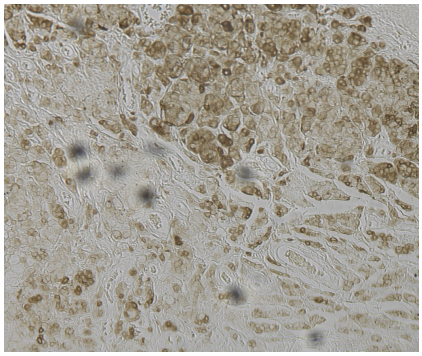

Figura 5.4: Transmitancia de nevo.

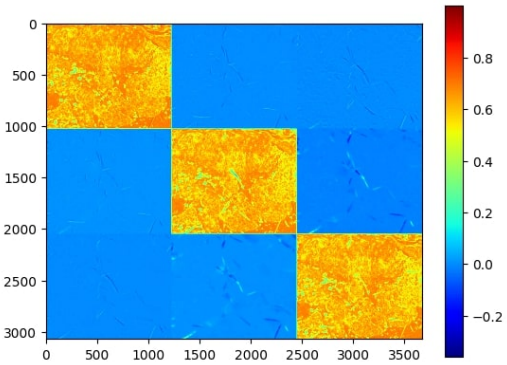

Figura 5.5: Matriz de Mueller de melanoma.

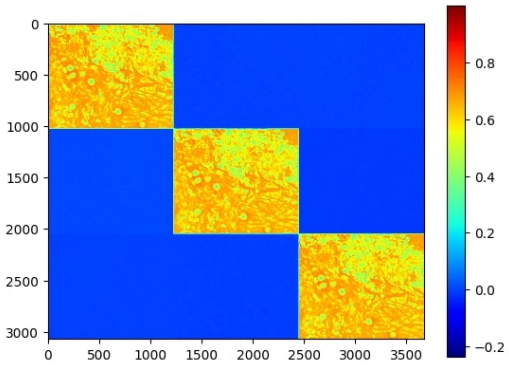

Figura 5.6: Matriz de Mueller de nevo.

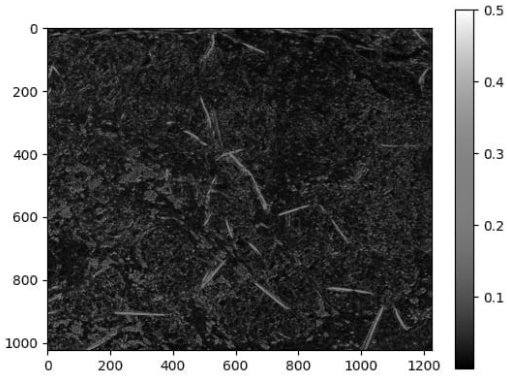

Figura 5.7: Polarizancia de melanoma.

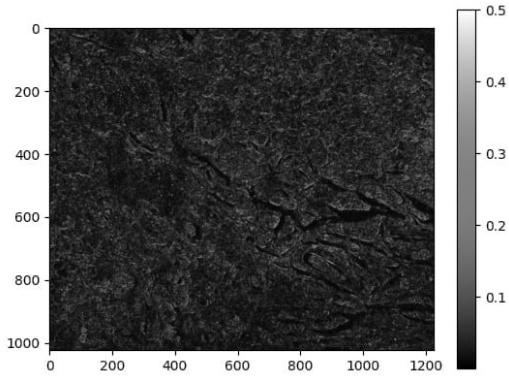

Figura 5.8: Polarizancia de nevo.

5.2. Pre-Procesamiento

Una vez adquirido el conjunto de imágenes que usaremos como base de datos, procedemos a procesarlas para que estén listas a ser sometidas a los algoritmos de aprendizaje automático. Podemos dividir el preprocesamiento en tres partes: *stitching* del conjunto de imágenes, segmentación de la zona de interés y filtrado de segmentos de recta (rayaduras del cubreobjetos). El *stitching* es fundamental para obtener información del tejido a mesoescala y poder hacer un reconocimiento de patrones basado en las variaciones espaciales de sus características. Por otro lado, la segmentación de las imágenes de transmitancia nos permite poder separar las regiones que son de interés y analizarlas adecuadamente mediante métodos estadísticos. Las zonas que nos interesan son las que tienen mayor concentración melanocítica, las cuales se corresponden con un color marrón más oscuro. Por último, puesto que el cubreobjetos tiene pequeñas fisuras, probablemente ocasionadas por impactos en el manejo de las muestras, se crea un cambio brusco en el índice de refracción (aire/vidrio) en las zonas fracturadas, cambiando la polarización de la luz que se transmite a través de ellas (ver Figura 5.7). Esto afecta a las medidas polarimétricas del tejido y deben ser eliminadas para evitar una posible contaminación de las estadísticas que hagamos.

Para fusionar cada conjunto de imágenes, aprovechando que se había conseguido un buen *stitching* con las imágenes de transmitancia, utilizamos el algoritmo entrenado sobre estas y lo ejecutamos sobre las de polarizancia. En las Figuras 5.9 - 5.10 vemos los resultados para uno de los conjuntos clasificados como melanoma.

La segmentación de la zona de interés fue realizada utilizando el algoritmo 'K-Means', haciendo un *clustering* sobre el mapa de colores. Este nos permite enmascarar la transmitancia de muestra completa en dos clases distintas: tejido y fondo (ver Figura 5.11). Por comodidad, nos quedamos con la máscara de tejido.

Aplicamos algoritmos de morfología matemática a la máscara de tejido para diseñar una nueva que capture su forma general (ver Figura 5.12). Para conseguirla, utilizamos una serie de técnicas en el siguiente orden: erosión, dilatación, clausura y *fill holes* [58][59]. La combinación erosión-dilatación selecciona las zonas enmascaradas de mayor tamaño; clausura "encierra" las regiones separadas de tejido y *fill holes* rellena cada una de ellas. Juntando la máscara original y esta última, obtuvimos la máscara final de la Figura 5.13. Aplicándola sobre la transmitancia y la polarizancia del tejido, obtenemos los resultados de las Figuras 5.14 - 5.15.

Finalmente, para el filtrado de las fisuras del cubreobjetos usamos la clase de OpenCV llamada `LineSegmentDetector()` [60]. Se define un detector de líneas que se aplica a la imagen y se crea una máscara que excluya el conjunto de líneas detectadas. En las Figuras 5.16 podemos ver en rojo las líneas detectadas sobre la imagen original de polarizancia de tejido. Básicamente, LSD detecta contornos localmente rectos en las imágenes. A través del cálculo de gradiente, comienza detectando las líneas de nivel. Luego, mide cómo se distribuyen los ángulos de dichas líneas y produce un campo de vectores unitarios. Segmenta el campo de vectores en conjuntos candidatos a líneas y por último las valida en función de la geometría de la imagen original.

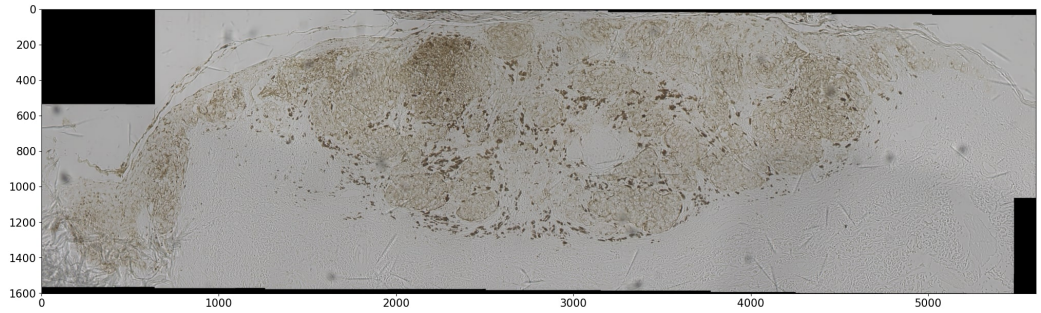

Figura 5.9: Transmitancia de melanoma de muestra completa.

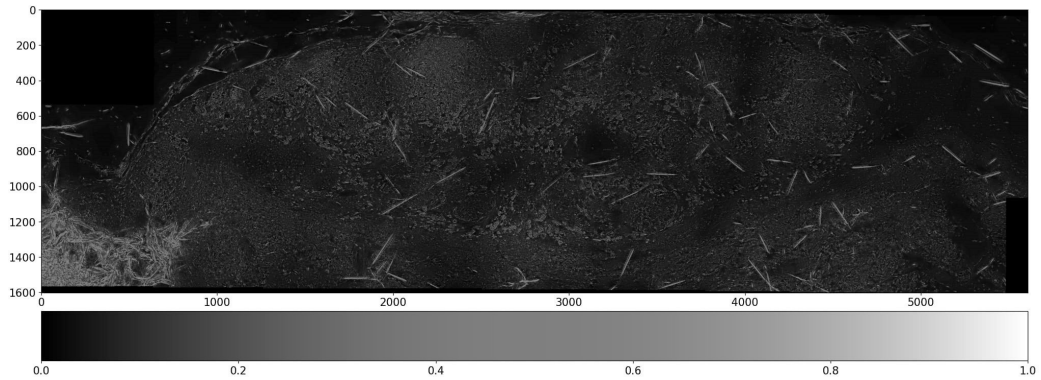

Figura 5.10: Polarizancia de melanoma de muestra completa.

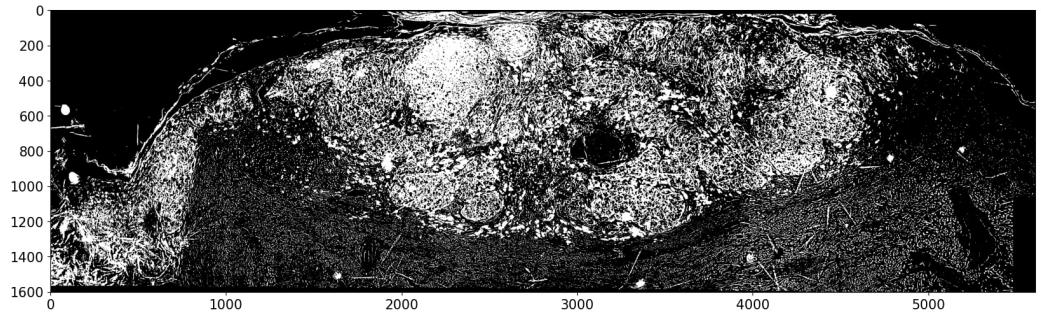

Figura 5.11: Máscara de forma detallada de tejido.

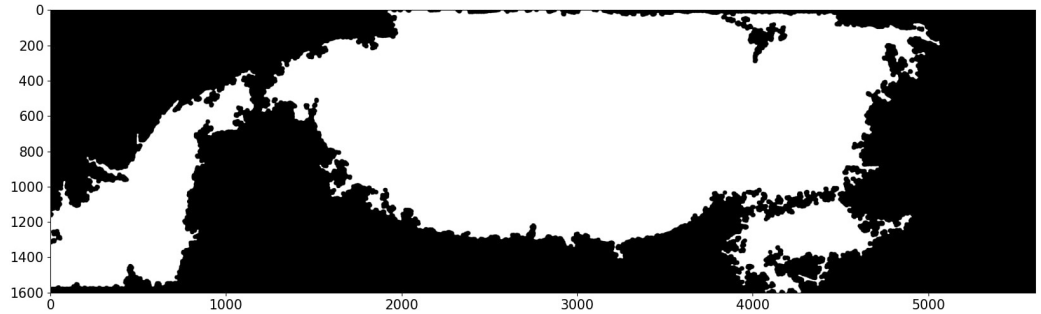

Figura 5.12: Máscara de forma general de tejido.

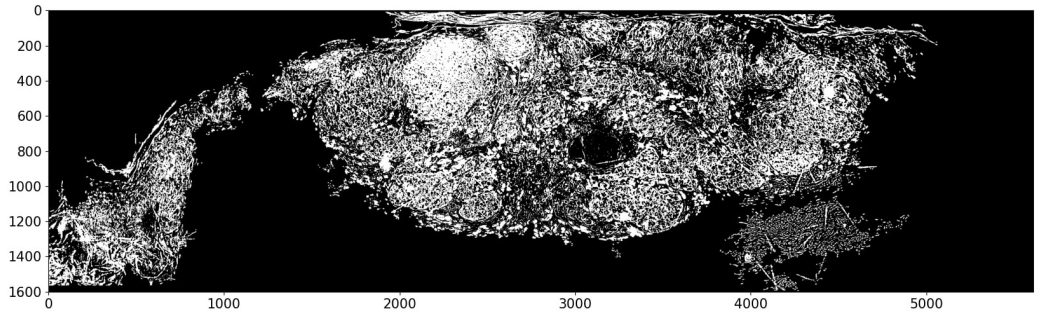

Figura 5.13: Máscara final de zona melanocítica.

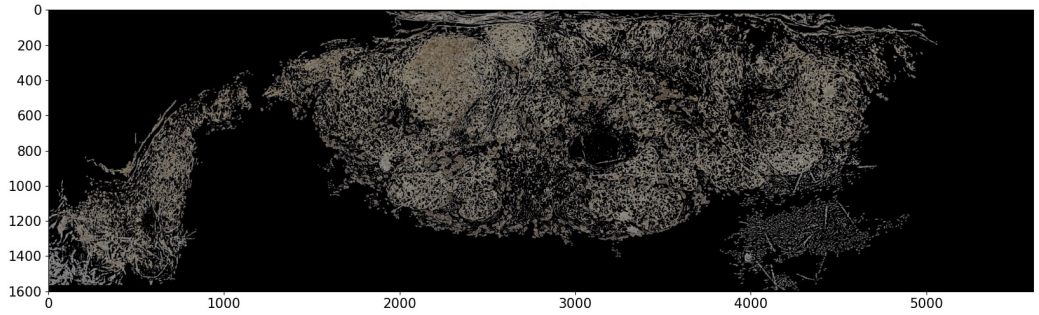

Figura 5.14: Transmitancia de tejido enmascarada.

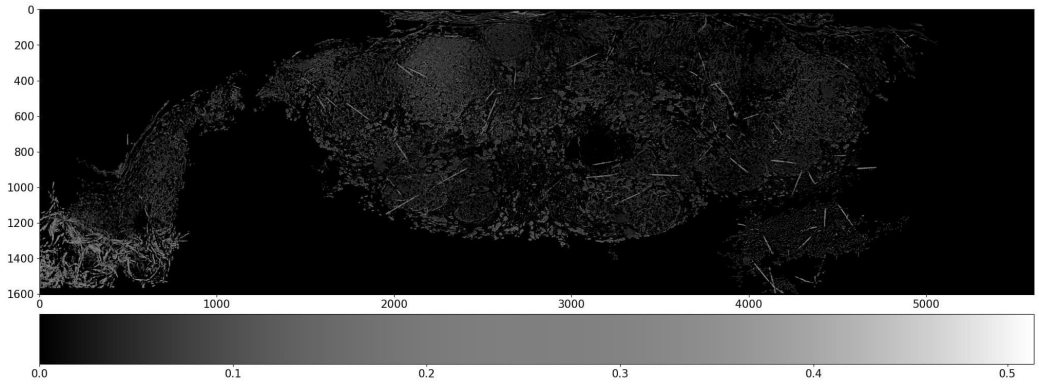

Figura 5.15: Polarizancia de tejido enmascarada.

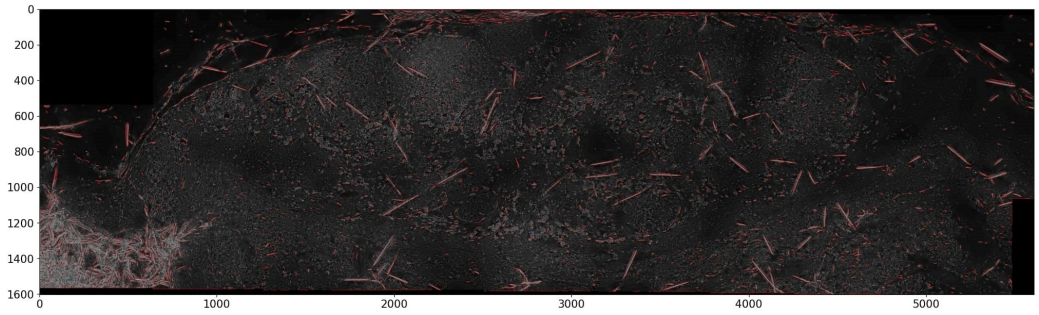

Figura 5.16: Polarizancia de melanoma de muestra completa junto a los segmentos de línea detectados utilizando LSD.

En las Figuras 5.17 - 5.18 vemos el resultado del filtrado de segmentos de líneas para muestras de melanoma y nevo. En ambos casos, recortamos la zona de mayor interés y ajustamos el contraste. Ambas imágenes presentan valores de polarizancia en el intervalo de 0 a 0,2. La profundidad de bits de la cámara, los leves desenfoques del sistema y los errores de precisión en el cálculo de Mueller, resultan en una variación de la medida que puede aproximarse en 0,01.

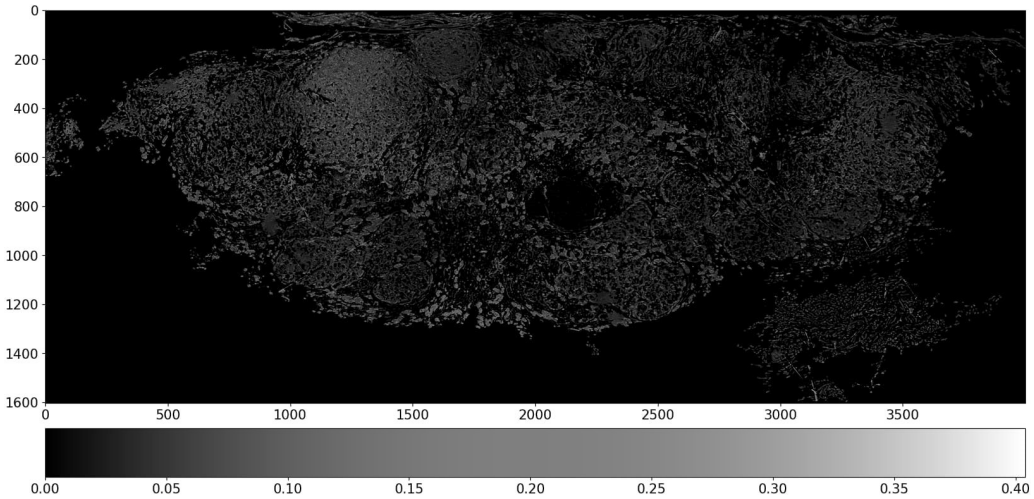

Figura 5.17: Polarizancia de melanoma enmascarado, filtrada y recortada.

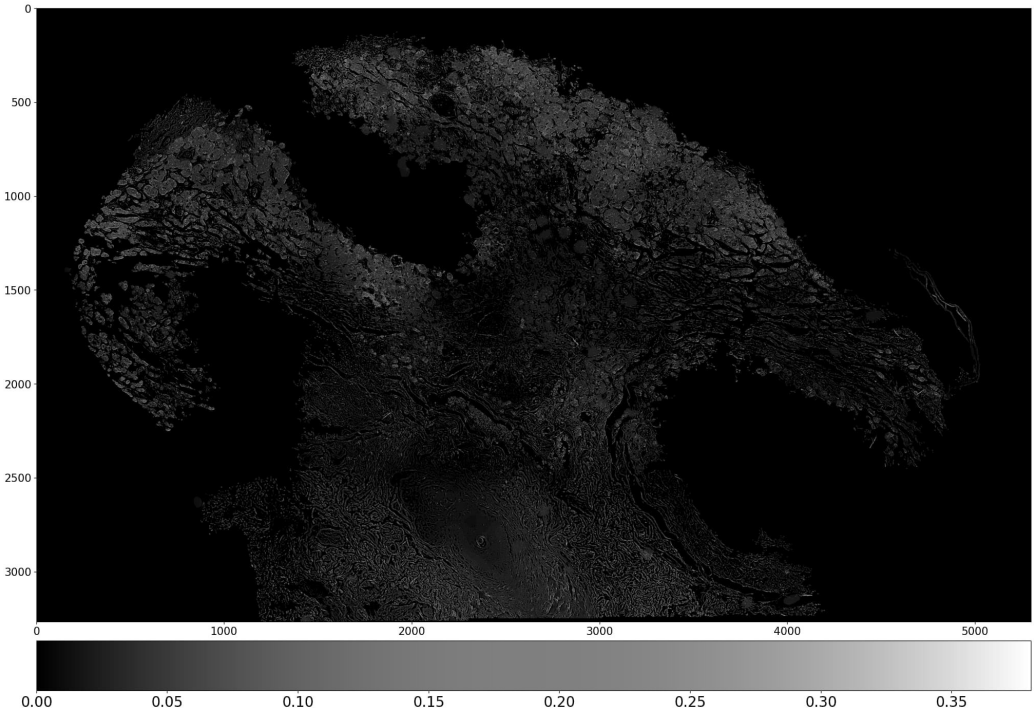

Figura 5.18: Polarizancia de nevo enmascarado, filtrada y recortada.

Si analizamos los histogramas de polarizancia en muestras de melanoma y nevo en la Figura 5.19, vemos que los valores de intensidad de cada píxel no son suficientes para discriminar entre un tipo de muestra y otra. Por lo tanto, debemos considerar en nuestras estadísticas la información estructural. Una forma de hacerlo es cuadriculando la imagen, dividiendo la imagen original en varios recortes y en cada uno de ellos calcular un vector de características que mida distintas propiedades de la región. Ver Figura 5.20.

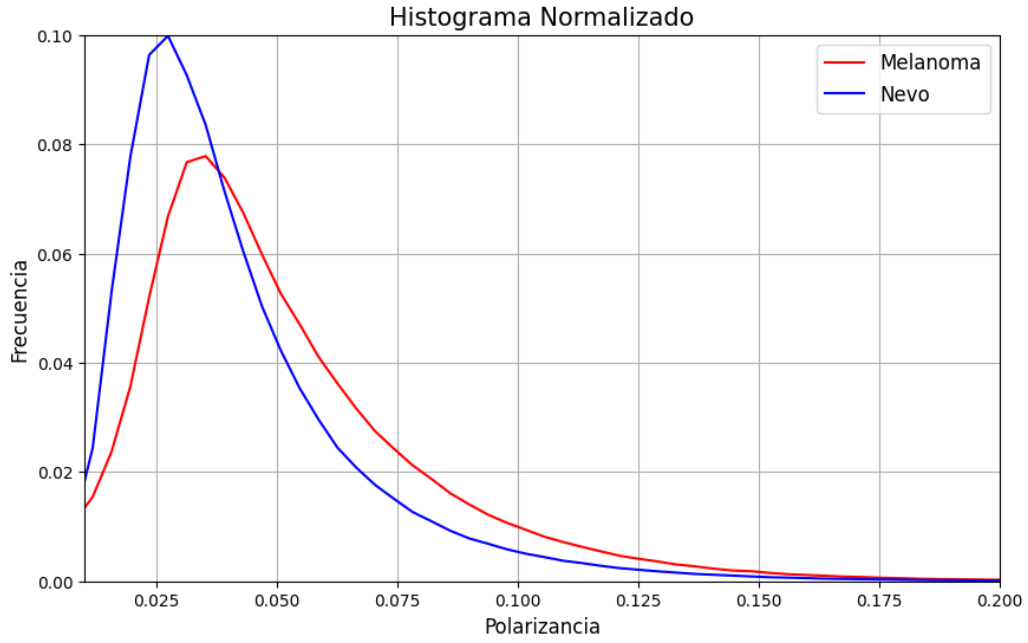

Figura 5.19: Histogramas de polarizancia en muestras de melanoma y nevo.

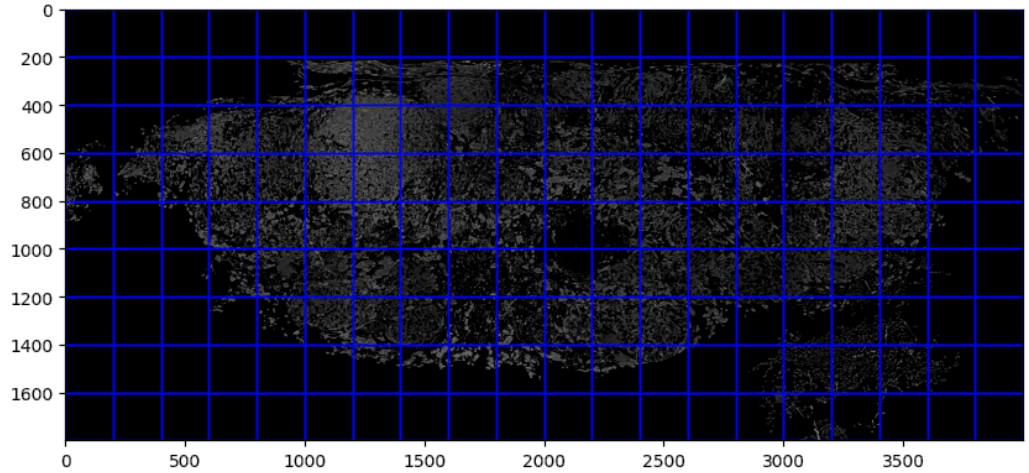

Figura 5.20: Cuadriculado de imagen de polarizancia de melanoma.

5.3. Reconocimiento de Patrones y Clasificación

El reconocimiento de patrones se refiere al proceso de identificar, interpretar y clasificar patrones y regularidades en conjuntos de datos. Esto puede implicar, como se verá a continuación, la extracción de características significativas de los datos, pero también la reducción de la dimensionalidad, y/o el descubrimiento de relaciones o estructuras entre distintos tipos de datos.

La clasificación, por otro lado, se refiere a la tarea de asignar etiquetas o categorías a los datos basándose en características observadas. Una manera usual de lograrlo es entrenando un modelo de aprendizaje automático utilizando ejemplos etiquetados (conjunto de entrenamiento) para aprender a predecir la clase de nuevos ejemplos no etiquetados (conjunto de test o prueba). Si el algoritmo tiene un buen desempeño en entrenamiento y también en test, decimos que generaliza bien.

Recortamos las imágenes de polarizancia en varios *patches* como se aprecia en la Figura 5.20, calculando en cada uno de ellos un vector de características X_i que mida: media, desviación estándar y entropía. En la Figura 5.21 vemos un diagrama de dispersión de los n vectores X_i calculados. Las características se guardan en una matriz X de tamaño $n \times 3$. El vector de etiquetas Y tiene tamaño $n \times 1$ y vale 1 si el vector es clase "Nevo" o 0 si es clase "Melanoma". Puesto que los valores muy bajos de media se confunden con el fondo, hicimos un filtrado por umbralización, dejando aquellos X_i cuya media queden por encima de 0,01. Recortando y filtrando de esta manera una imagen correspondiente a melanoma y otra a nevo obtuvimos $n = 552$ vectores. Además, como las medidas tienen diferencias notables en sus rangos, estandarizamos los X_i para que tengan media nula y desviación estándar unitaria. Por último, mezclamos la matriz X y el vector Y de forma aleatoria y los separamos en conjuntos de entrenamiento, validación y prueba.

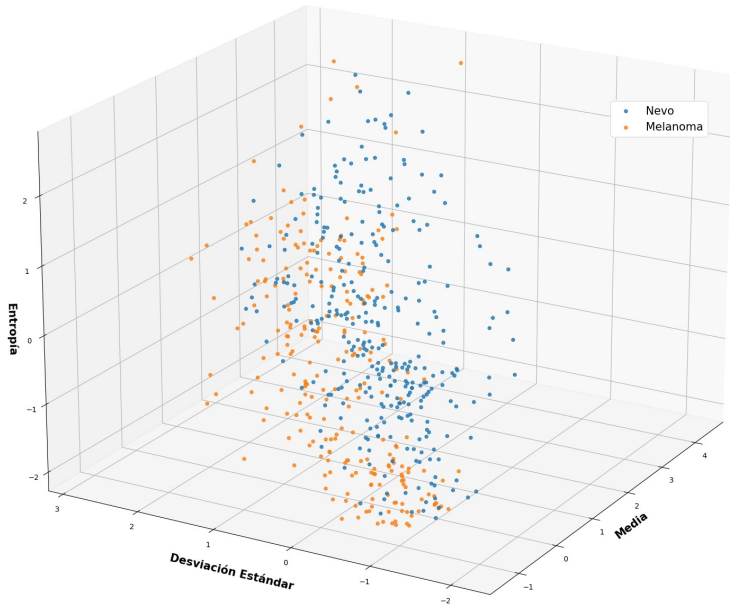

Figura 5.21: Diagrama de dispersión: media vs. desviación estándar vs. entropía.

Los datos están preparados para diseñar los clasificadores que veremos en las Secciones 5.3.1 - 5.3.3. Estos modelos trabajan sobre la matriz X y el vector Y y se caracterizan por ajustar una serie de parámetros que modelan una superficie que separa los conjuntos de datos según la clase a la que pertenezcan. Por otra parte, en la Sección 5.3.4 utilizaremos un modelo que trabaja directamente sobre las imágenes. Toma como conjunto de entrenamiento las imágenes etiquetadas, halla las características de forma automática y ajusta sus parámetros para clasificarlas.

5.3.1. *Support Vector Machine*

Las *Support Vector Machines* (SVM) son un tipo de algoritmo de aprendizaje supervisado utilizado para clasificación. La idea principal detrás de SVM es encontrar el hiperplano óptimo que mejor separa las clases en el espacio de características. Un hiperplano es una superficie de decisión que divide un espacio de características en dos regiones, una para cada clase. La "máxima separación" se refiere a encontrar el hiperplano que maximiza la distancia entre las clases, lo que ayuda a mejorar la capacidad de generalización del modelo.

En caso de que la división entre los datos no siga un modelo lineal, se puede optar por utilizar el modelo con un kernel no lineal, como el kernel RBF (Radial Basis Function), o uno polinomial. Así, el SVM es capaz de aprender fronteras de decisión no lineales en el espacio de características (ver Figura 5.22). El kernel no lineal transforma los datos de entrada en un espacio de características de dimensionalidad superior, donde es más probable que los datos sean linealmente separables. Esto permite al SVM encontrar un hiperplano de separación óptimo en este espacio de características transformado y luego aplicarlo a los datos en el espacio previo a la transformación.

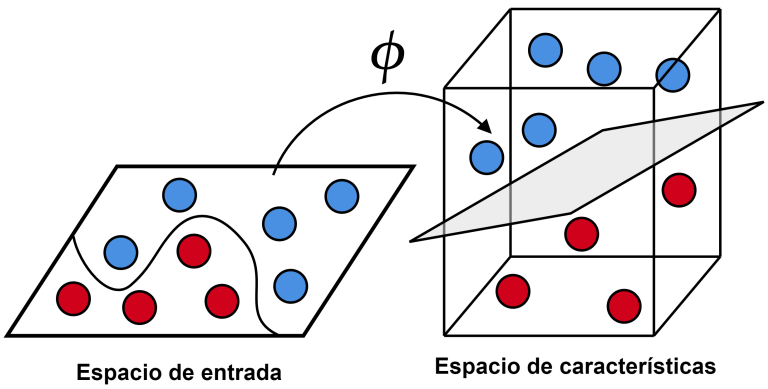

Figura 5.22: Ejemplo de Transformación del espacio de características. Realizado en Mathcha.

El SVC puede ser regularizado mediante el parámetro C, buscando el equilibrio entre la maximización del margen y la minimización del error de clasificación en el conjunto de entrenamiento. Este parámetro modifica la penalización por clasificar incorrectamente puntos de datos en el conjunto de entrenamiento, es decir, expresa cuánto se desea evitar clasificar erróneamente cada muestra de entrenamiento. Un

valor más bajo de C permite un margen más amplio y una mayor tolerancia a errores de clasificación en el conjunto de entrenamiento, lo que puede resultar en un modelo más suave y menos propenso al sobre ajuste. Por otro lado, un valor más alto de C penaliza más fuertemente los errores de clasificación en el conjunto de entrenamiento, lo que puede resultar en un margen más estrecho y un modelo más propenso al sobre ajuste. Debe buscarse un valor que optimice el rendimiento del modelo sin caer en estas desventajas.

5.3.2. *Random Forest*

Random Forest es también un algoritmo de aprendizaje supervisado utilizado para clasificación y regresión. Es una técnica de ensamblaje que combina múltiples árboles de decisión (*decision trees*) independientes, conocidos como "árboles de decisión de base", para tomar decisiones predictivas.

Los árboles de decisión se basan en la estructura de un árbol donde cada nodo interno representa una característica (o atributo), cada rama representa una regla de decisión basada en esa característica, y cada hoja representa el resultado de la clasificación o regresión. En un árbol de decisión para clasificación, cada nodo interno divide el conjunto de datos en función de una característica específica, eligiendo la característica que mejor separa las clases. Este proceso se repite recursivamente en cada nodo interno hasta que se alcanza un criterio de parada, como la pureza de las clases en las hojas o la profundidad máxima del árbol (ver Figura 5.23).

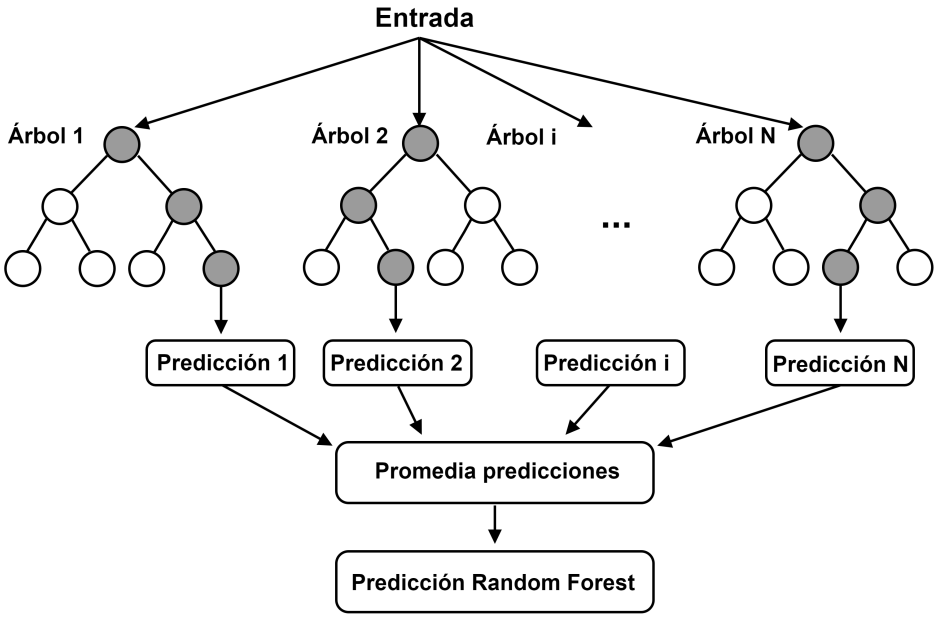

Figura 5.23: Diagrama de ejemplo de RF. Realizado en Mathcha.

La idea principal detrás de *Random Forest* es crear un conjunto diverso de árboles de decisión mediante el entrenamiento de cada árbol en una muestra alea-

toria de los datos y considerando un subconjunto aleatorio de las características en cada división de nodo. La predicción final se realiza promediando las predicciones de todos los árboles individuales (en el caso de la clasificación) o tomando el promedio ponderado (en el caso de la regresión). Se busca minimizar una medida de impureza como Gini o Entropía, que dan una idea de cuánto un nodo en un árbol de decisión está "mezclado" (contiene muestras de más de una clase) con respecto a las clases de destino. Un parámetro importante en el *Random Forest* es la profundidad del árbol, que refiere a la máxima cantidad de niveles (o capas) que puede tener. Aumentarlo permite que el árbol se adapte más a los datos de entrenamiento, pero puede llevar al sobre ajuste, mientras que disminuirlo puede llevar a un modelo menos complejo, e incluso sub ajuste.

5.3.3. *Multi-Layer Perceptron*

El *Multi-Layer Perceptron* (MLP) es un tipo de red neuronal que consta de múltiples capas de neuronas: una capa de entrada, una o varias capas ocultas y una capa de salida (ver Figura 5.24). Cada neurona en una capa está conectada a todos las neuronas de la capa siguiente. La información fluye hacia adelante a través de la red, desde la capa de entrada, pasando por las capas ocultas, hasta la capa de salida, donde se genera la predicción o salida deseada.

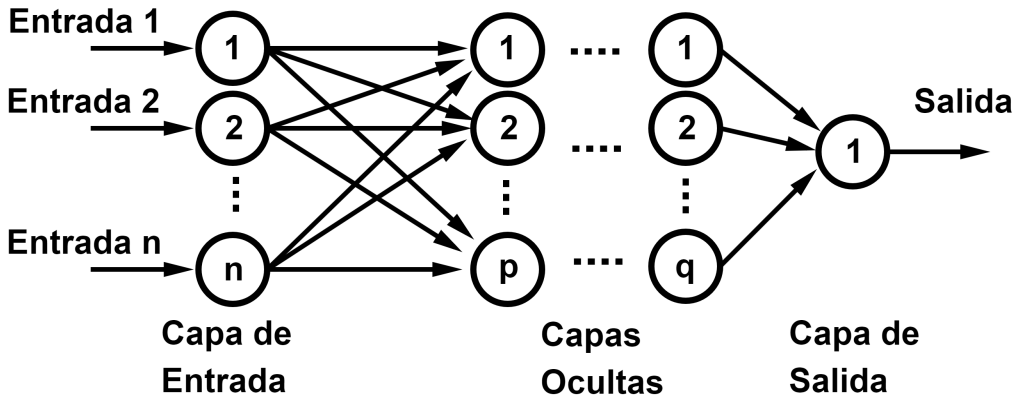

Figura 5.24: Diagrama de ejemplo de MLP. Realizado en Mathcha.

El perceptrón es la arquitectura más sencilla de todos. Las entradas de la capa de entrada, x_j, se conectan a cada neurona, j, de la capa posterior, i, y son ponderadas por una serie de pesos, w_{ij}. En general, a la capa de entrada se le agrega un término de *bias* que introduce un nuevo peso, b_j. Luego, se suman y se transforma el resultado mediante la función de activación, ϕ:

$$z_j = \phi(w_{1j}x_1 + w_{2j}x_2 + \dots w_{nj}x_n + b_j). \tag{5.1}$$

Una única neurona puede ser utilizada para una simple clasificación binaria. No obstante, si combinamos varias neuronas y las conectamos a todas las entradas, podemos hacer clasificación de múltiples clases. Cuando en una capa todas las

neuronas están conectadas a todas las neuronas de la capa anterior, decimos que es una capa densa o *fully connected* (ver Figura 5.25). Cuando conectamos varios perceptrones en un modelo de capas, tenemos una MLP como en la Figura 5.24.

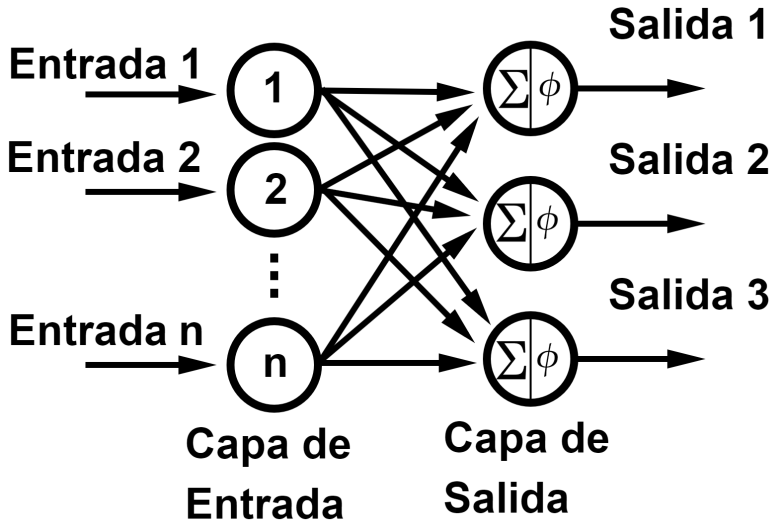

Figura 5.25: Diagrama de ejemplo de perceptrón. Realizado en Mathcha.

Las funciones de activación son componentes clave en las redes neuronales, las cuales se utilizan para introducir no linealidades en el modelo y permitir a la red aprender y representar relaciones más complejas en los datos. Es usual que la función de activación de la capa de salida dependa del tipo de salida que esperamos (binaria, múltiples clases, continua, etc.), mientras que las capas ocultas pueden variar dependiendo del problema y los resultados durante el entrenamiento del modelo específico. Algunas de las funciones de activación más comunes son:

- **ReLU** (*Rectified Linear Unit*):
  - Función matemática: $f(x) = \text{máx}(0, x)$.
  - Ventaja: Eficiente computacionalmente, no sufre del problema de desvanecimiento del gradiente, introduce no linealidad.
  - Desventaja: No es simétrica alrededor de cero, lo que puede llevar a la "muerte"de neuronas en la fase de entrenamiento.
- **Tanh** (Tangente Hiperbólica):
  - Función matemática: $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.
  - Ventaja: Produce una salida en el rango $(-1, 1)$, lo que puede ayudar a estabilizar el entrenamiento.
  - Desventaja: Sufre del problema de desvanecimiento del gradiente para entradas grandes o pequeñas.

- **Sigmoide**:
  - Función matemática: $f(x) = \frac{1}{1+e^{-x}}$.
  - Ventaja: Produce una salida en el rango $(0, 1)$, que puede interpretarse como la probabilidad de pertenecer a una clase, por lo que es usual verla en la capa de salida para problemas de clasificación binaria.
  - Desventaja: Sufre del problema de desvanecimiento del gradiente, menos común en capas ocultas debido a la saturación en los extremos.
- **Softmax**:
  - Función matemática: $f(x)j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$.
  - Ventaja: Produce una salida que representa una distribución de probabilidad sobre las clases, lo que facilita la interpretación y la toma de decisiones en problemas de clasificación multiclase.
  - Desventaja: Requiere un cálculo costoso debido a la suma de todas las puntuaciones exponenciales. Puede ser sensible a los valores atípicos en los datos de entrada.

Las MLP se constituyen de varias capas de perceptrones. La capa i-ésima es entrenada mediante una serie de pasos en función del error que comete, ajustando sus parámetros, w_{ij}, para que el error de predicción sea cada vez más chico. Más precisamente, se inyectan las entradas del primer paso, x_{1i}, con salidas ideales, y_{1i}. La capa i realiza una predicción con salida, $\hat{y}_{1i}$, y calcula el error de predicción $e_{1i} = f(y_{1i}, \hat{y}_{1i})$, siendo f una función de costo. Se ajustan los pesos, w_{ij}, para seguir reduciendo el error y repite el proceso con las entradas del paso siguiente, x_{2i}, y así sucesivamente. Una forma de actualizar los pesos es mediante el algoritmo de descenso por gradiente:

$$w_{ij}^{m+1} = w_{ij}^{m} - \eta \nabla_w f(y_{mi}, \hat{y}_{mi}), \tag{5.2}$$

donde:

- w_{ij}^{m} son los pesos de la capa i y neurona j en el paso m.
- y_{mi} son las salidas ideales de la capa i en el paso m.
- $\hat{y}_{mi}$ son las salidas calculadas de la capa i en el paso m.
- η es un parámetro de entrenamiento denominado *learning_rate*.

Existe distintas variantes de descenso por gradiente, de los cuales se destacan:

- **BGD** (*Batch Gradient Descent*):
  - Calcula el gradiente de la función de costo utilizando todo el conjunto de entrenamiento. Los pesos se actualizan como se aprecia en la ecuación (5.2).

  - Ventaja: Dirección de descenso precisa.
  - Desventaja: Entrenamiento lento.

- **SGD** (*Stochastic Gradient Descent*)
  - Elige un paso de forma aleatoria y estima el gradiente con los datos de entrenamiento en dicho paso.
  - Ventaja: Entrenamiento más rápido que BGD.
  - Desventaja: Dirección de descenso poco precisa.

- **RMSProp**:
  - Es una modificación de SGD que provee un *learning_rate* adaptativo. Acumula la información de los gradientes más recientes e incrementa el *learning_rate* cuando la superficie es muy empinada y lo decrece si está cerca de un punto crítico.
  - Ventaja: Entrenamiento más rápido y preciso que SGD.
  - Desventaja: El *learning_rate* inicial se ajustan manualmente.

- **Adam**:
  - Es una modificación de RMSProp que implementa el concepto físico de *momentum*. Los pesos descienden rápidamente si la superficie se vuelve empinada, pero alcanza una velocidad límite que depende de un coeficiente de fricción β.
  - Ventaja: Entrenamiento más rápido y preciso que RMSProp.
  - Desventaja: El *learning_rate* inicial y β se ajustan manualmente.

La función de costo es una medida de desempeño del entrenamiento. Sin embargo, un valor muy bajo no necesariamente implica que el modelo sea bueno. A medida que la red aprende sobre el conjunto de entrenamiento, corre el riesgo de ajustarse demasiado a los datos y perder poder de generalización, fenómeno conocido como sobre ajuste. Las funciones de costo más comunes son:

- **Entropía Binaria Cruzada** (*Binary Cross Entropy*):
  - Función matemática:

$$f(y_i, \hat{y}_i) = -\frac{1}{m}\sum_{i=1}^{m}\left[y_i \log\left(\hat{y}_i\right) + \left(1 - y_i\right)\log\left(1 - \hat{y}_i\right)\right]. \tag{5.3}$$

  - Ventaja: Es una función convexa, lo que asegura la convergencia del algoritmo de optimización.
  - Desventaja: Solo funciona para clases binarias.

- **Entropía Cruzada** (*Cross Entropy*):

- Función matemática:

$$f(y_i, \hat{y}_i) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} y_i^{(k)} \log (\hat{y}_i^{(k)}), \tag{5.4}$$

- Ventaja: Funciona para múltiples clases.
- Desventaja: Sensible a datos atípicos.

- **MSE** (*Mean Square Error*):
  - Función matemática: $f(y_i, \hat{y}_i) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$.
  - Ventaja: Sensible a los datos atípicos debido al exponente cuadrático, lo cual puede ser útil para detectarlos.
  - Desventaja: La misma sensibilidad a los datos atípicos puede resultar en un mal ajuste.
- **MAE** (*Mean Absolute Error*):
  - Función matemática: $f(y_i, \hat{y}_i) = \frac{1}{m} \sum_{i=1}^{m} |y_i - \hat{y}_i|$.
  - Ventaja: Robusto a los datos atípicos, lo que provee un mejor ajuste.
  - Desventaja: No es diferenciable en cero.

Al construir una MLP, es necesario tener en cuenta los distintos parámetros que podemos ajustar. Dependiendo del problema que nos enfrentemos, debemos elegir con cuidado los parámetros de la red para obtener un resultado óptimo.

5.3.4. *Convolutional Neural Network*

La *Convolutional Neural Network* (CNN) emerge de los estudios de la corteza visual, región del cerebro que procesa la información captada por la retina. Su arquitectura se basa en una serie de bloques con diversas funciones, de los cuales se destaca la capa convolucional. La red toma una imagen de entrada y, mediante una serie de filtros convolucionales, aprende sobre las principales características de la imagen. A diferencia de MLP, en la CNN cada neurona de la capa convolucional 1 se conecta solamente a un rectángulo de la imagen de entrada, denominado campo receptivo (ver Figura 5.26). Sucesivamente, cada neurona de la capa i-ésima se conecta a un rectángulo de la capa $(i-1)$-ésima. La arquitectura permite a la red concentrarse en las características de bajo nivel en las primeras entradas y las de alto nivel en las últimas. Este tipo de estructura jerarquizada es muy común en las imágenes reales, razón por la cual las CNN tienen una eficiencia tan buena.

Más precisamente, cada capa convolucional consta de múltiples filtros que devuelven un mapa de características por cada uno de ellos, donde todas las neuronas de un mapa de características comparten los mismos parámetros. Esto reduce el número de parámetros de la red y, a diferencia de MLP, una vez que la CNN aprende los patrones de una región de la imagen, los puede reconocer en cualquier

otra región de la misma. Por otro lado, el campo receptivo de cada neurona es el mismo que antes, solo que ahora se extiende a cada uno de los filtros, lo que permite detectar varias características en cada región de la imagen.

Para regularizar la red y reducir el tamaño de los filtros, se colocan capas de *pooling*. Funcionan de forma similar a las capas convolucionales: cada neurona de la capa de *pooling* se conecta a las neuronas de la región receptiva de la capa anterior. Sin embargo, este tipo de capas no tiene parámetros, sino que aplica operaciones sobre la región, como la media o el máximo.

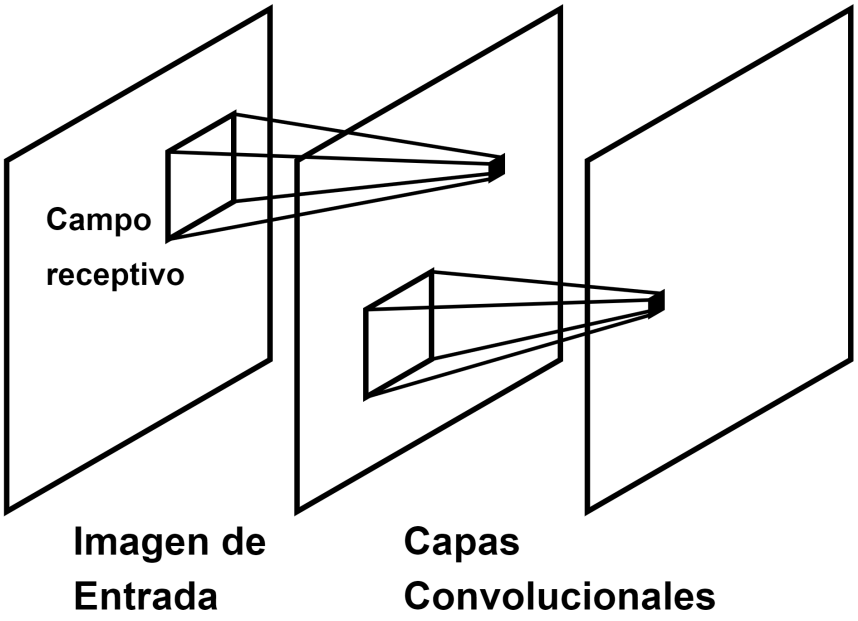

Figura 5.26: Capas convolucionales. Realizado en Mathcha.

La arquitectura típica de una red CNN consta de varios conjuntos de capas de convolución, capas de activación (generalmente, ReLU) y capas de *pooling*. Las capas se vuelven cada vez más chicas pero con características más profundas. Por último, en el encabezado de la red se coloca una capa de *flatten* para redimensionar el filtro bidimensional a un vector unidimensional. De esta forma se puede conectar una red MLP para que termine la tarea. En la Figura 5.27 vemos un esquema general de la arquitectura de una red CNN.

Con el correr de los años, han surgido arquitecturas más complejas. Un ejemplo es 'Xception', una red con decenas de capas de profundidad y entrenada con más de un millón de imágenes de la base de datos de 'ImageNet'. La base de datos cuenta con más de 1000 categorías de objetos convencionales (por ejemplo, animales, autos, paisajes, etc.). No obstante, esto no significa que no pueda ser utilizada en nuestro caso particular, puesto que los patrones de bajo nivel de las estructuras celulares pueden ser bastante comunes en otros tipos de objetos.

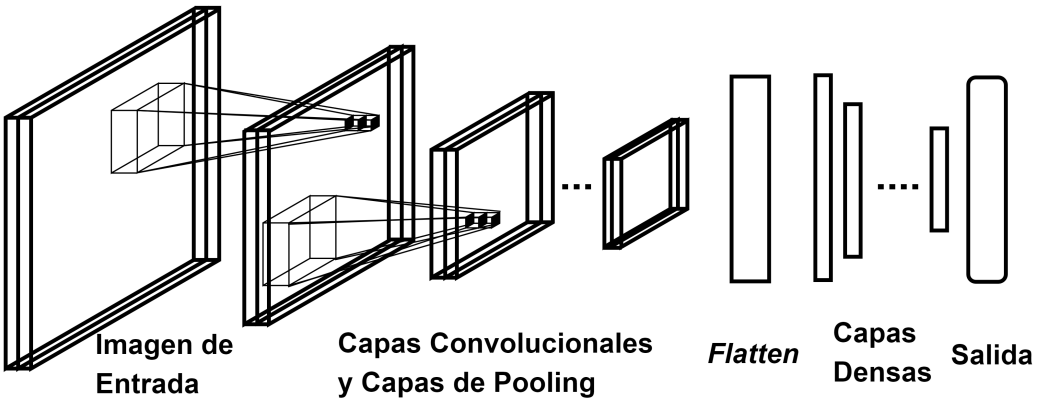

Figura 5.27: Diagrama de ejemplo de CNN. Realizado en Mathcha.

5.4. Clasificación de Tejido Humano

Después del filtrado, contamos con 552 muestras tridimensionales en total, lo que, en perspectiva con la complejidad del modelo abordado, parece configurar una disponibilidad limitada de datos. Por esta razón, se decidió dividir el conjunto en dos, dejando un 80 % para entrenamiento y 20 % para test, habiéndolos barajado para evitar sesgos en el orden; y tratar de validar los datos mediante validación cruzada. Esto nos deja con 411 muestras en entrenamiento y 111 en test, en los que se comprobó la proporción de clases, obteniendo 56,9/43,1 y 52,3/47,7 respectivamente, lo que indica un leve desbalance de clases.

A partir de estos datos se entrenan los tres primeros modelos presentados anteriormente, SVM, RF y MLP, ya que la CNN requiere otro tipo de datos y trabajo previo, que se explicará posteriormente. Para este primer grupo de modelos, se comenzó con una configuración inicial con parámetros en valores usuales, y luego tratando de mejorar su desempeño en validación cuanto fuera posible antes de probar el modelo en el conjunto de test. Para la validación cruzada se usaron $k = 5$ *folds*, utilizando la clase `KFold` de scikit-learn. Esta validación cruzada divide el conjunto de datos en pliegues (*folds*), barajando aleatoriamente los datos antes de dividirlos. Inicialmente, se obtuvieron valores de exactitud (en validación cruzada) cercanos al 60 % para los tres modelos, siendo *Random Forest* el más preciso, y SVC el menos, sin que haya diferencias significativas entre los puntajes ($\approx \pm 5\,\%$). En cada caso se realizó una búsqueda aleatoria de parámetros con `RandomizedSearchCV()` utilizando también 5 *folds*, lo que mejoró el desempeño en validación. Luego de algunas iteraciones, llevó a los modelos finales que se presentan a continuación, y que finalmente fueron ejecutados sobre los datos de test, arrojando los resultados presentados en la Tabla 5.1.

Para el SVC, se llegó al mejor resultado en validación con un kernel lineal y aplicando regularización con $C = 5$. Para RF, se consiguieron los siguientes parámetros:

- Criterio: Entropía.
- Profundidad Máxima: 9.

- Muestras Mínimas por Hoja: 9.
- Muestras Mínimas para Dividir el Nodo: 5.
- Número de Estimadores (Árboles): 260.

Este continuó siendo el modelo con mejor desempeño de los tres. En la Figura 5.28, se ve la frontera de decisión conseguida para una proyección 2D de los datos, habiendo reducido la dimensión de datos con PCA y ajustando el clasificador a esos datos.

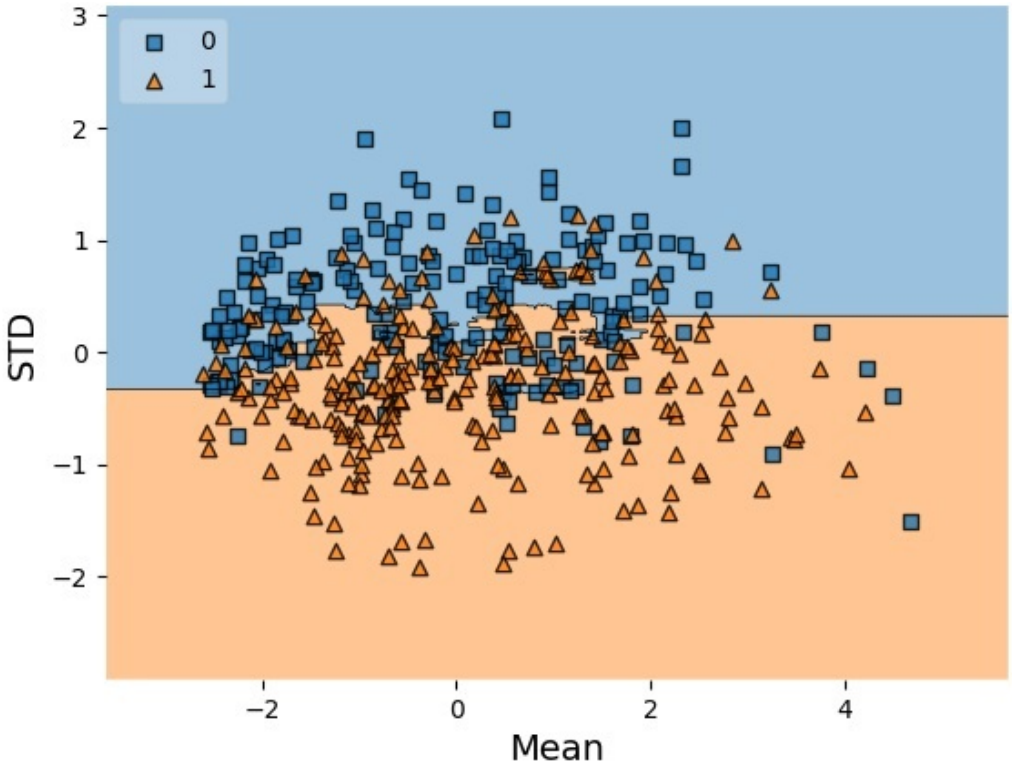

Figura 5.28: Proyección 2D con reducción PCA de la frontera de decisión del *Random Forest*.

Finalmente, el MLP se construyó con seis capas ocultas de seis neuronas, utiliza la función de activación ReLU en las capas ocultas, optimiza los pesos con el algoritmo Adam y aplica regularización L2 con un parámetro de regularización de 0,001, con una tasa de aprendizaje constante de 0,001. En la Figura 5.29 vemos el resultado luego de iterar un total de 150 veces.

La función de costo decae logarítmicamente, lo que indica que el modelo mejora rápidamente al principio y luego se va estancando, cesando su variación; que se detenga en este caso indica que se alcanzó el criterio de parada de tolerancia. En específico, para la clase usada (`MLPClassifier` de `sklearn.neural_network`) y según nuestra configuración, cuando la pérdida o puntaje no mejora en al menos una cantidad de $tol = 1 \times 10^{-4}$ durante `n_iter_no_change` $= 10$ iteraciones consecutivas, se considera que se ha alcanzado la convergencia y el entrenamiento se detiene.

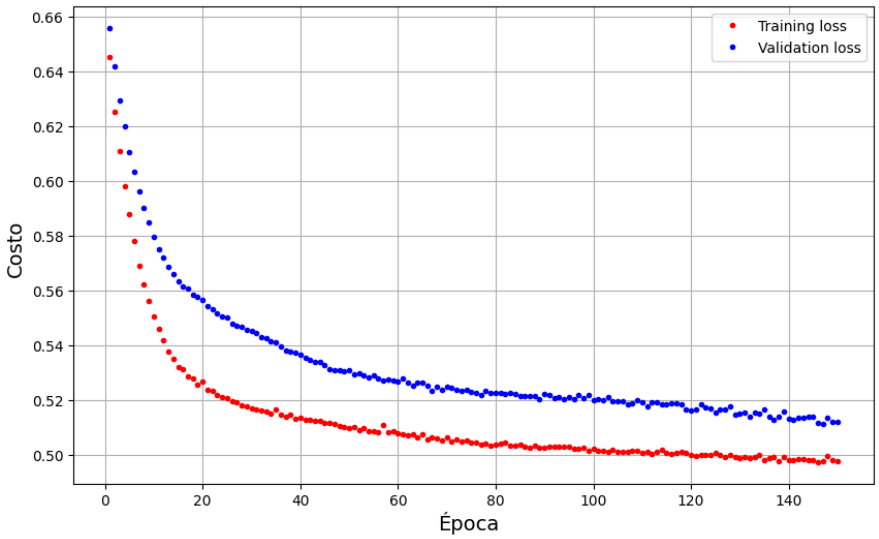

Figura 5.29: Evolución de la función de costo en entrenamiento y validación para el MLP.

Por otro lado, para el modelo de CNN, puesto que no contamos con suficientes datos para entrenar un modelo convolucional, decidimos utilizar una técnica de aumentado de datos. Aplicamos transformaciones de volteos a cada muestra completa, y los agregamos a nuestro conjunto de entrenamiento. Las imágenes de entrada son finalmente estandarizadas para evitar posibles inestabilidades numéricas durante el entrenamiento. En total contamos con 1739 recortes de imágenes (aproximadamente 50 % de ellas son nevo), 70 % para entrenar, 15 % para validar y 15 % para testear. Algunos de los recortes de imágenes sin estandarizar se pueden ver en la Figura 5.30.

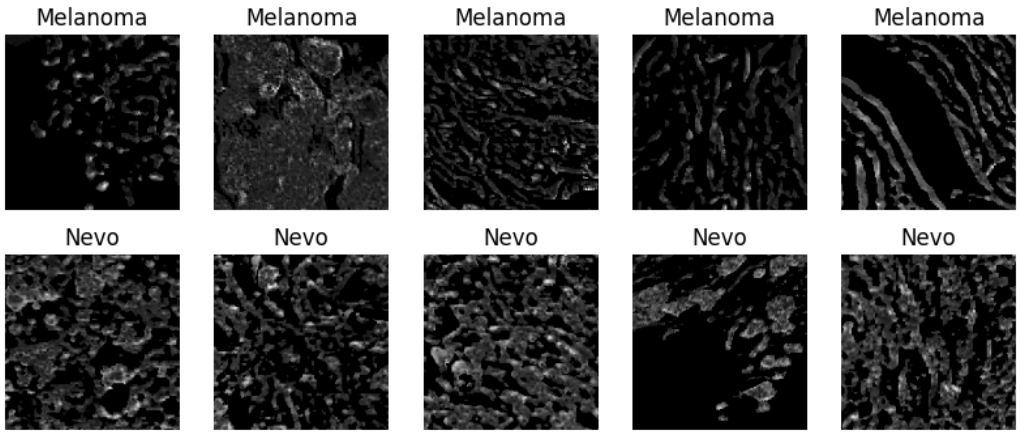

Figura 5.30: Recortes de melanoma y nevo utilizados como entrenamiento.

Para entrenar con una red convolucional, utilizamos 'Keras' de la librería TensorFlow. Cargamos el modelo prediseñado 'Xception' con los pesos descargados de

'ImageNet' y sin encabezado. Le conectamos al final de la red un modelo MLP compuesto por tres capas con activación ReLU de (256,128,64) neuronas y una neurona final con activación 'sigmoid'. Para tener un buen control del sobre ajuste, utilizamos SGD como optimizador con *learning_rate* igual a 1×10^{-4}. Como el problema es de clasificación binaria y las clases están balanceadas, la función de costo elegida fue entropía binaria cruzada. El entrenamiento duró 35 épocas y las curvas de aprendizaje se pueden apreciar en la Figura 5.31.

Existen dos métricas que miden la calidad del algoritmo de predicción: precisión y recuperación. La precisión mide la capacidad de no etiquetar un melanoma como nevo. La recuperación mide la habilidad de detectar nevos. Ajustamos el umbral de decisión para que ambas métricas sean lo más altas posible. Según las curvas de precisión y recuperación en el conjunto validación de la Figura 5.32, un umbral de aproximadamente 0,5 implicaría casi un 90 % de precisión y recuperación, siendo este el umbral elegido.

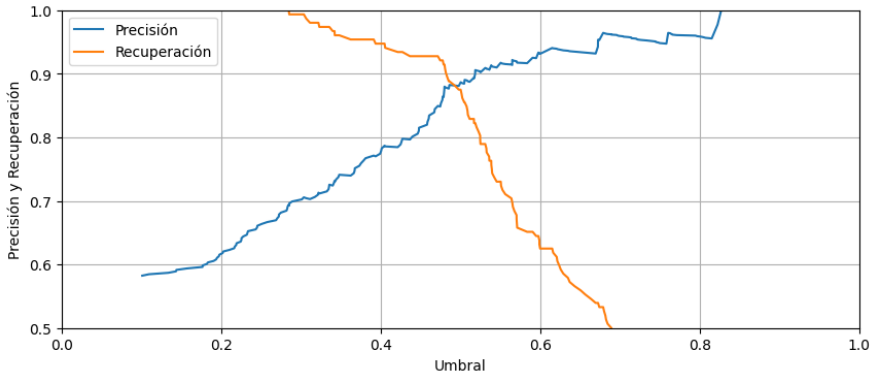

Figura 5.31: Curva de aprendizaje.

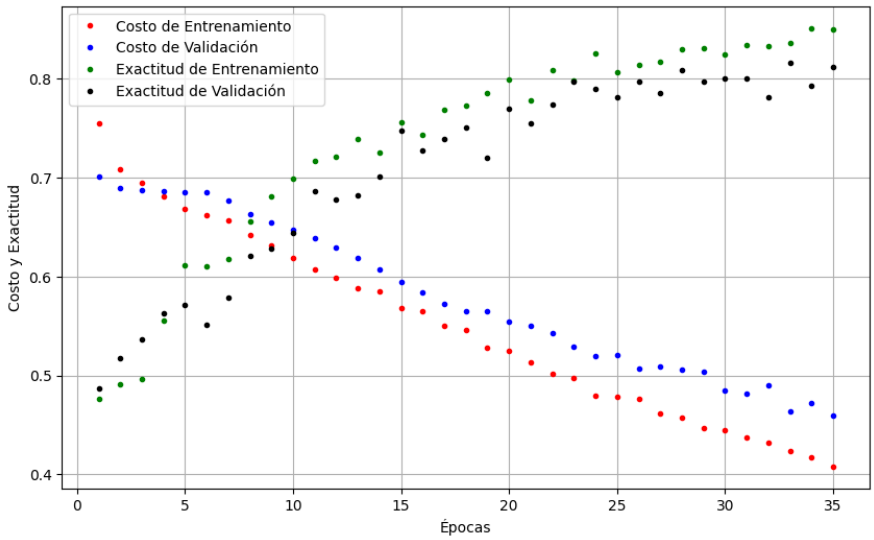

Figura 5.32: Curvas de precisión y recuperación en validación.

Para terminar, elegimos como modelo final la CNN. En las cuatro imágenes de muestra completa que disponemos, predecimos la clase de cada recorte y los coloreamos de color rojo si el modelo le asigna la clase melanoma o verde si le asigna la clase nevo. Los resultados de las predicciones en cada cuadrante, junto al porcentaje de acierto, se observan en las Figuras 5.34 - 5.36. En general, observamos como la mayor parte de tejido se predice correctamente en cada uno de los casos. En particular, en la Figura 5.36 vemos la predicción con exactitud más baja. Notamos como al modelo le cuesta clasificar aquellas zonas del tejido que tienen poca melanina (Ver Figura B.4).

En la Tabla 5.1 vemos los resultados obtenidos para los conjuntos originales, sin aumentado de datos. El entrenamiento logra resultados por encima del 70 % de exactitud, destacando el modelo CNN con un 86 % en prueba.

Exactitud	**SVC**	**RF**	**MLP**	**CNN**
Entrenamiento	0.7506	0.8299	0.7460	0.8661
Validación	0.7392	0.7233	0.6870	0.8084
Prueba	0.7207	0.7568	0.7207	0.8621

Tabla 5.1: Resultados del entrenamiento con los algoritmos elegidos.

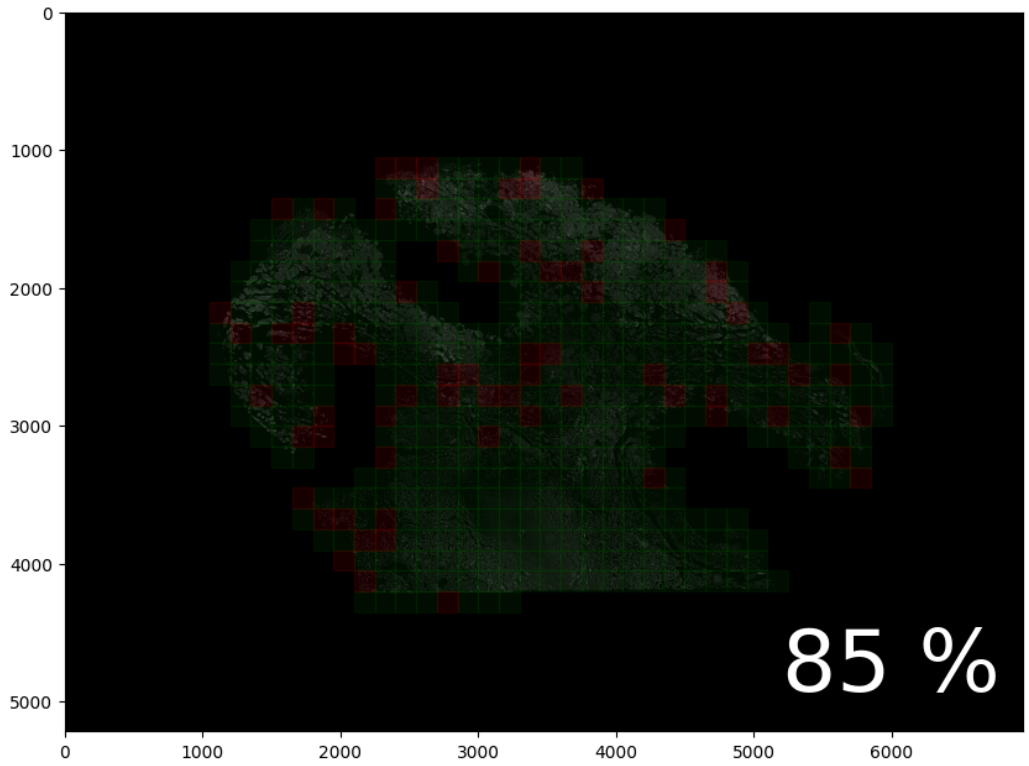

Figura 5.33: Predicción sobre muestra de nevo 1.

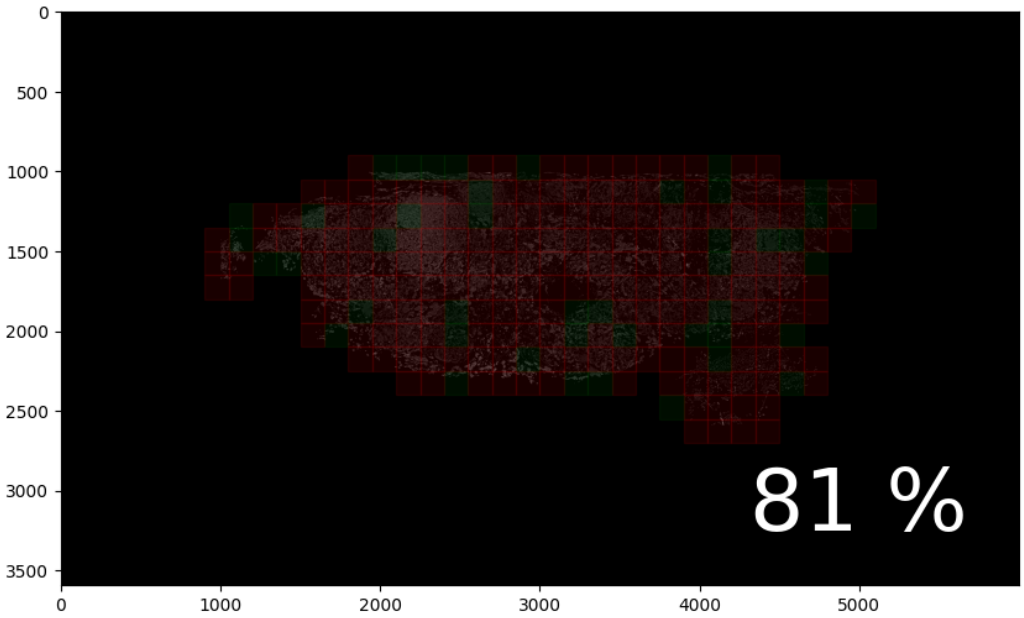

Figura 5.34: Predicción sobre muestra de melanoma 1.

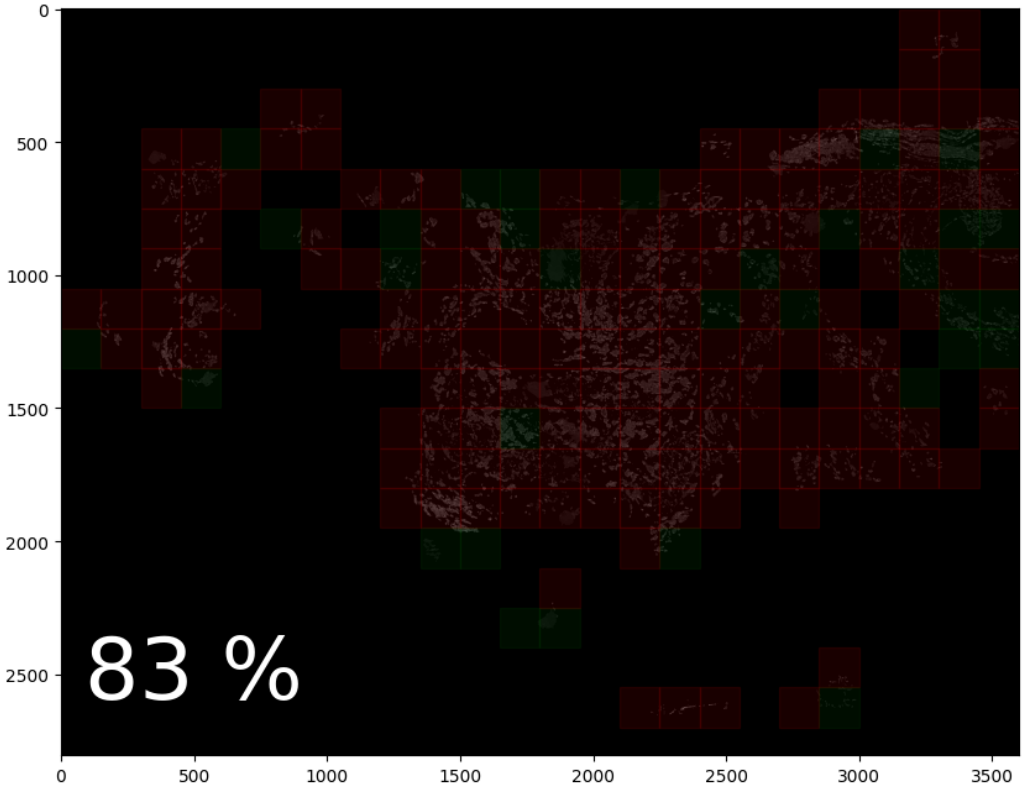

Figura 5.35: Predicción sobre muestra de melanoma 2.

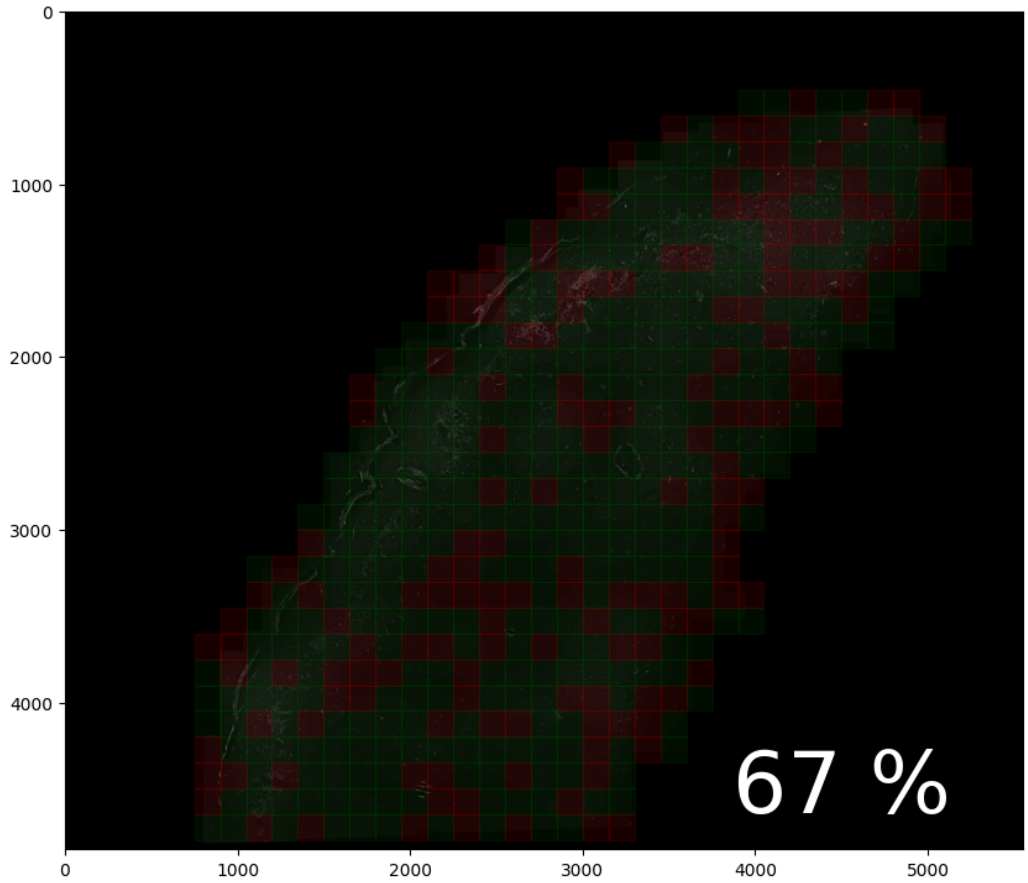

Figura 5.36: Predicción sobre muestra de nevo 2.

Conclusiones y Trabajo a Futuro

Realizaremos una evaluación global del trabajo realizado, considerando los resultados obtenidos y los objetivos planteados. Adicionalmente, proponemos algunas posibles líneas de trabajo que podrían ser implementadas a partir del sistema desarrollado y que resultan de interés a la luz de las conclusiones halladas.

5.5. Conclusiones

Durante el transcurso de este trabajo, logramos distintos objetivos con la finalidad de conseguir un sistema completo y funcional de microscopía de luz polarizada de muestra completa. Utilizamos el sistema para dos aplicaciones específicas: *stitching* de matrices de Mueller y diseño de un clasificador de lesiones melanocíticas: melanoma y nevo.

En primer lugar, desarrollamos un microscopio de matrices de Mueller con FoV limitado para caracterizar las principales propiedades polarimétricas de tejidos biológicos. El desempeño de este microscopio probó ser acertado, permitiendo calcular matrices con valores similares a los teóricos para elementos ópticos cuyas matrices son conocidas.

Para poder registrar imágenes de muestra completa, implementamos un mecanismo motorizado. El microscopio lo diseñamos prácticamente desde cero, lo que incluye: el modelado digital, las estructuras fabricadas en impresora 3D, la instalación de la electrónica adquirida localmente, y el *software* correspondiente para manipularlo. El sistema es capaz de recorrer y fotografiar muestras de hasta 1 cm^2. En otras palabras, fabricamos un dispositivo para realizar Imaginería de matrices de Mueller de Muestra Completa, que recorre, fotografía, y guarda las imágenes necesarias para obtener las propiedades físicas del tejido entero, con resultados fotográficamente adecuados.

Para las imágenes recabadas por el sistema, implementamos en Python el *software* necesario para fusionarlas en una sola escena final con el método conocido como *Stitching*. A partir de una librería *open-source*, acondicionamos el algoritmo para nuestro caso particular. Además, conseguimos reducir el tiempo de ejecución necesario para completar la fusión a partir de nuevas funciones que retienen los parámetros conseguidos a partir de un *set* de imágenes y evitan tener que recalcularlos en cada ejecución.

Creamos una base de datos con imágenes de muestras de tejido de piel humana suministradas por el Instituto Pasteur, clasificadas por un experto como nevo y melanoma. Armamos las correspondientes imágenes de muestra completa a partir del método de stitching, lo que contiene un valor intrínseco más allá del estudio concreto que realizamos. Además, logramos utilizar algoritmos de segmentación y morfología matemática para extraer la zona de interés del tejido y poder trabajar más adecuadamente.

Finalmente, construimos varios modelos de Aprendizaje Automático entrenados con la polarizancia del tejido, que lograron arrojar resultados prometedores en cuanto a la posible clasificación de tejido Entre ellos se destaca el modelo de red neuronal convolucional, que alcanzó un valor de exactitud por encima del 80 % en el conjunto de prueba.

5.6. Trabajo a Futuro

Proponemos tres posibles sugerencias que extenderían el alcance del trabajo, tanto a nivel de *hardware* como de *software*.

La primera consiste en extender la polarimetría a matrices de Mueller 3×4 mediante la incorporación de una lámina de retardo lineal al sistema de iluminación. Esto permitiría el cálculo de tres nuevas componentes matriciales, con las cuales sería posible calcular la diatenuación circular del tejido y obtener medidas más precisas de la birrefringencia y la despolarizancia. La metodología computacional sería similar a la expuesta en este trabajo, con la única excepción de que el parámetro de Stokes de entrada S_3 debería ser modelado teóricamente.

La segunda, radica en una mejora del algoritmo de *stitching* de matrices de Mueller. En lugar de usar una métrica de distancia escalar en cada componente, se podría definir una métrica de distancia entre matrices. De esta manera, se podrían hallar las correspondencias entre los puntos clave utilizando toda la información disponible y de forma más robusta. Esto involucraría la modificación del proceso de *feature detection & feature matching* del algoritmo de *stitching*. Más específicamente, la clase `FeatureDetector()` debería considerar las características de todas las componentes matriciales y la clase `FeatureMatcher()` debería emparejarlas utilizando la nueva métrica de distancia matricial.

La tercera reside en la posibilidad de fusionar la microscopía de matrices de Mueller y la microscopía hiperespectral de fluorescencia. La ingeniería de la iluminación podría combinar elementos de visión múltiple para obtener matrices de Mueller y filtros con modulación seno/coseno en la transmitancia para obtener información espectral sin necesidad de obtener el espectro. La integración de estas técnicas presentaría un nuevo enfoque en histopatología para una evaluación cuantitativa y basada en la caracterización estructural así como molecular de las enfermedades, proporcionando además un mejor entendimiento de los procesos patológicos subyacentes.

Apéndice A

Cálculos Adicionales

A.1. Ecuación de Ondas en los Medios Materiales

Para modelar las ecuaciones de Maxwell en un medio material, debemos hacer explícita la interacción de la materia con los nuevos términos fuente (ρ, $\boldsymbol{J}$) a través de la carga ligada ρ_{ligada} y la densidad de corriente eléctrica ligada $\boldsymbol{J}_{ligada}$. En este caso debemos suponer que la longitud de onda de las ondas que atraviesa el medio es lo suficientemente grande como para despreciar la distancia inter-atómica del material. De esta forma podemos definir dos campos que nos serán de ayuda: el campo de polarización $\boldsymbol{P}$ y el campo de magnetización $\boldsymbol{M}$.

La teoría electromagnética prueba el siguiente resultado:

$$\rho_{ligada} = -\boldsymbol{\nabla} \cdot \boldsymbol{P}, \quad \boldsymbol{J}_{ligada} = \frac{\partial \boldsymbol{P}}{\partial t} + \boldsymbol{\nabla} \times \boldsymbol{M}. \tag{A.1}$$

Entonces, bajo un delicado tratamiento, las ecuaciones de Maxwell se pueden reescribir en función de cantidades macroscópicas:

$$\boldsymbol{\nabla} \cdot \boldsymbol{D} = \frac{\rho_{libre}}{\epsilon_0} , \tag{A.2}$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{B} = 0 , \tag{A.3}$$

$$\boldsymbol{\nabla} \times \boldsymbol{E} = -\frac{\partial \boldsymbol{B}}{\partial t} , \tag{A.4}$$

$$\boldsymbol{\nabla} \times \boldsymbol{H} = \boldsymbol{J}_{libre} + \frac{\partial \boldsymbol{D}}{\partial t} . \tag{A.5}$$

Las cantidades $\rho = \rho_{libre} + \rho_{ligada}$ y $\boldsymbol{J} = \boldsymbol{J}_{libre} + \boldsymbol{J}_{ligada}$ son los términos fuentes. Los campos auxiliares $\boldsymbol{D}$ y $\boldsymbol{H}$ son conocidos como campo de desplazamiento eléctrico y campo de intensidad magnética, definidos de la siguiente forma:

$$\boldsymbol{D} = \epsilon_0 \boldsymbol{E} + \boldsymbol{P}, \quad \boldsymbol{H} = \frac{1}{\mu_0} \boldsymbol{B} - \boldsymbol{M}. \tag{A.6}$$

Para los medios lineales, no dispersivos y no magnéticos, se cumplen las siguientes relaciones constitucionales:

$$\boldsymbol{P} = \epsilon_0 \chi_e(\boldsymbol{r}) \boldsymbol{E}, \quad \boldsymbol{M} = \chi_m(\boldsymbol{r}) \boldsymbol{H}, \tag{A.7}$$

dónde χ_e es el tensor de susceptibilidad eléctrica y χ_m es el tensor de susceptibilidad magnética. Si suponemos además que el medio es homogéneo e isótropo, los tensores se reducen a simples constantes, concluyendo en las siguientes relaciones lineales:

$$\boldsymbol{D} = \epsilon \boldsymbol{E}, \quad \boldsymbol{H} = \frac{1}{\mu}\boldsymbol{B}, \tag{A.8}$$

siendo $\epsilon = \epsilon_0(1 + \chi_e)$ y $\mu = \mu_0(1 + \chi_m)$. Supongamos esta vez que los términos fuentes ρ_{libre} y $\boldsymbol{J}_{libre}$ son nulos. Si tomamos rotor en la ecuación (A.4):

$$\boldsymbol{\nabla} \times (\boldsymbol{\nabla} \times \boldsymbol{E}) = -\boldsymbol{\nabla}^2 \boldsymbol{E} + \boldsymbol{\nabla}(\underbrace{\boldsymbol{\nabla} \cdot \boldsymbol{E}}_{=0}) = -\frac{\partial(\boldsymbol{\nabla} \times \boldsymbol{B})}{\partial t} = -\epsilon\mu \frac{\partial^2 \boldsymbol{E}}{\partial^2 t}. \tag{A.9}$$

Despejando, hayamos una ecuación de ondas en el campo eléctrico:

$$\boldsymbol{\nabla}^2 \boldsymbol{E} - \frac{n^2}{\epsilon_0 \mu_0} \frac{\partial^2 \boldsymbol{E}}{\partial^2 t} = 0, \tag{A.10}$$

siendo el índice de refracción del medio una constante definido cómo:

$$n^2 = \frac{\mu\epsilon}{\mu_0 \epsilon_0} \simeq \frac{\epsilon}{\epsilon_0}. \tag{A.11}$$

Análogamente para el campo magnético:

$$\boldsymbol{\nabla} \times (\boldsymbol{\nabla} \times \boldsymbol{H}) = -\boldsymbol{\nabla}^2 \boldsymbol{H} + \boldsymbol{\nabla}(\underbrace{\boldsymbol{\nabla} \cdot \boldsymbol{H}}_{=0}) = -\epsilon \frac{\partial(\boldsymbol{\nabla} \times \boldsymbol{E})}{\partial t} = -\epsilon \frac{\partial^2 \boldsymbol{B}}{\partial^2 t}. \tag{A.12}$$

Entonces:

$$\boldsymbol{\nabla}^2 \boldsymbol{B} - \frac{n^2}{\epsilon_0 \mu_0} \frac{\partial^2 \boldsymbol{B}}{\partial^2 t} = 0. \tag{A.13}$$

Las ecuaciones (A.10) y (A.13) modelan la propagación de una onda electromagnética en un medio con índice de refracción n homogéneo. El caso inhomogéneo se estudia particularmente en la óptica geométrica, cuando la longitud de onda es despreciable frente a los objetos que rodean al rayo de luz. Cuando el medio es anisotrópico, la relación entre $\boldsymbol{D}$ y $\boldsymbol{E}$ es tensorial, lo que nos permite modelar fenómenos como la birrefringencia y la actividad óptica.

Buscaremos a continuación una solución particular al problema de propagación de ondas electromagnéticas polarizadas en los medios lineales, isótropos, homogéneos, no dispersivos y no magnéticos, en coordenadas cartesianas. En este sistema de coordenadas la ecuación de ondas se reescribe en tres ecuaciones escalares:

$$\boldsymbol{\nabla}^2 u_i(\boldsymbol{r}, t) - \frac{1}{c^2} \frac{\partial^2 u_i(\boldsymbol{r}, t)}{\partial t^2} = 0, \quad i = 1, 2, 3 \tag{A.14}$$

Con respecto a la variable espacial $\boldsymbol{r}$, tenemos dos tipos de ondas particulares: planas y esféricas. En lo que se refiere a la variable temporal t, existen dos sumamente útiles: monocromáticas y cuasi-monocromáticas.

A.2. Solución de Onda Plana

Sea $\boldsymbol{r}$ una posición de cualquier punto del espacio y $\hat{\boldsymbol{k}}$ una dirección unitaria cualquiera, diremos que $u_i(\boldsymbol{r}, t)$ es una solución de onda escalar plana si:

$$u_i(\boldsymbol{r}, t) = u_i(\hat{\boldsymbol{k}} \cdot \boldsymbol{r}, t). \tag{A.15}$$

Es decir, $u_i(\boldsymbol{r}, t)$ toma un valor constante en un plano de la forma:

$$\zeta(\boldsymbol{r}) = \boldsymbol{k} \cdot \boldsymbol{r} = k_x x + k_y y + k_z z = cte. \tag{A.16}$$

Observemos además lo siguiente:

$$\frac{\partial}{\partial x} = k_x \frac{\partial}{\partial \zeta}, \quad \frac{\partial}{\partial y} = k_y \frac{\partial}{\partial \zeta}, \quad \frac{\partial}{\partial z} = k_z \frac{\partial}{\partial \zeta}. \tag{A.17}$$

Sabemos que $|\boldsymbol{k}| = k$, entonces la ecuación (A.14) la podemos reescribir:

$$\frac{\partial^2 u_i}{\partial \zeta^2} - \frac{1}{k^2 c^2} \frac{\partial^2 u_i}{\partial t^2} = 0, \quad i = 1, 2, 3. \tag{A.18}$$

Aplicando la Transformada de Fourier temporal a ambos lados y utilizando la relación de dispersión $\omega = kc$:

$$\frac{\partial^2 U_i}{\partial \zeta^2} + U_i = 0, \quad i = 1, 2, 3. \tag{A.19}$$

Obtenemos una ecuación diferencial ordinaria de segundo orden, cuya solución se calcula:

$$U_i(\zeta, \omega) = A_i(\omega) e^{i\zeta} + B_i(\omega) e^{-i\zeta}, \quad i = 1, 2, 3. \tag{A.20}$$

Antitransformando:

$$u_i(\zeta, t) = a_i\left(\zeta, t - \frac{\zeta}{\omega}\right) + b_i\left(\zeta, t + \frac{\zeta}{\omega}\right), \quad i = 1, 2, 3. \tag{A.21}$$

La ecuación (A.21) nos muestra que la solución de onda plana es una combinación lineal de dos funciones arbitrarias que viajan a la misma velocidad y en sentidos opuestos a través de los argumentos $t + \frac{\zeta}{\omega}$ y $t - \frac{\zeta}{\omega}$.

Las ondas monocromáticas son aquellas cuya variación temporal es periódica, sinusoidal. Matemáticamente, estas ondas representan las componentes principales de Fourier de cualquier onda más compleja. Mirando la ecuación (A.21), la solución monocromática más sencilla se puede expresar:

$$u_i(\boldsymbol{k}, t) = A_i \cos(\omega t - \boldsymbol{k} \cdot \boldsymbol{r} + \delta_i) + B_i \cos(\omega t + \boldsymbol{k} \cdot \boldsymbol{r} + \delta_i), \quad i = 1, 2, 3. \tag{A.22}$$

Supongamos el caso particular de la onda se propaga en la dirección z, es decir, $\boldsymbol{k} = (0, 0, k)^t$. Volviendo a la notación de campo eléctrico, la solución en coordenadas cartesianas se puede desarrollar:

$$\begin{cases} E_x(z, t) = E_{0x} \cos(\omega t - kz + \delta_x), \\ E_y(z, t) = E_{0y} \cos(\omega t - kz + \delta_y), \\ E_z(z, t) = E_{0z} \cos(\omega t - kz + \delta_z). \end{cases} \tag{A.23}$$

Es cómodo representar la ecuación (A.23) en su notación compleja de la siguiente manera:

$$\boldsymbol{E}(z,t) = \Re\{\hat{\boldsymbol{E}}_{\mathbf{0}} e^{i(\omega t - kz + \delta_x)}\}. \tag{A.24}$$

Tomando la divergencia del campo eléctrico y utilizando Ley de Gauss:

$$\boldsymbol{\nabla} \cdot \boldsymbol{E} = \boldsymbol{k} \cdot \boldsymbol{E} = 0. \tag{A.25}$$

Entonces, el campo eléctrico oscila transversalmente a su dirección de propagación, por lo que $E_{0z} = 0$ en la ecuación (A.23). Reescribiéndola en dos dimensiones:

$$\begin{cases} E_x(z,t) = E_{0x}\cos(\omega t - kz + \delta_x), \\ E_y(z,t) = E_{0y}\cos(\omega t - kz + \delta_y). \end{cases} \tag{A.26}$$

De igual forma podemos hacer con el campo magnético:

$$\boldsymbol{B}(z,t) = \Re\{\hat{\boldsymbol{B}}_{\mathbf{0}} e^{i(\omega t - kz + \delta_x)}\}. \tag{A.27}$$

Tomando rotor en $\boldsymbol{E}$ y usando la Ley de Faraday:

$$\boldsymbol{\nabla} \times \boldsymbol{E} = -\frac{\partial}{\partial t}(\boldsymbol{\nabla} \times \boldsymbol{B}) = \omega \boldsymbol{B} = \boldsymbol{k} \times \boldsymbol{E}. \tag{A.28}$$

La ecuación (A.28) nos demuestra como $\boldsymbol{E}$, $\boldsymbol{B}$ y $\boldsymbol{k}$ forman un triedro directo. Utilizando la relación de dispersión, podemos hallar explícitamente la forma del campo magnético:

$$\boldsymbol{B} = \frac{1}{c}(\hat{\boldsymbol{k}} \times \boldsymbol{E}). \tag{A.29}$$

Finalmente, nos interesa conocer la forma del vector de Poynting, que por definición se calcula:

$$\boldsymbol{S} = \frac{1}{\mu_0}\boldsymbol{E} \times \boldsymbol{B} = \frac{1}{c\mu_0}|\boldsymbol{E}|^2\hat{\boldsymbol{k}} = c\epsilon_0|\boldsymbol{E}|^2\hat{\boldsymbol{k}}. \tag{A.30}$$

Es decir, en una onda plana la energía se propaga en la misma dirección en que la onda viaja. Además, la intensidad instantánea de la onda es proporcional al cuadrado del módulo del campo eléctrico.

A.3. Parámetros de Stokes para Luz Monocromática

Supongamos un haz de luz monocromático propagándose en el eje $\hat{z}$ y tomemos $z = 0$ en las dos primeras ecuaciones:

$$E_x(t) = E_{0x}\cos(\omega t + \delta_x), \tag{A.31}$$

$$E_y(t) = E_{0y}\cos(\omega t + \delta_y). \tag{A.32}$$

Utilizando propiedades trigonométricas, podemos reescribir:

$$\frac{E_x(t)}{E_{0x}} = \cos(\omega t)\cos(\delta_x) - \sin(\omega t)\sin(\delta_x), \tag{A.33}$$

$$\frac{E_y(t)}{E_{0y}} = \cos(\omega t)\cos(\delta_y) - \sin(\omega t)\sin(\delta_y). \tag{A.34}$$

Operando un poco:

$$\frac{E_x(t)}{E_{0x}}\sin(\delta_y) - \frac{E_y(t)}{E_{0y}}\sin(\delta_x) = \cos(\omega t)\sin(\delta_y - \delta_x), \tag{A.35}$$

$$\frac{E_x(t)}{E_{0x}}\cos(\delta_y) - \frac{E_y(t)}{E_{0y}}\cos(\delta_x) = \sin(\omega t)\sin(\delta_y - \delta_x). \tag{A.36}$$

Sumando los cuadrados de ambos lados:

$$\frac{E_x^2(t)}{E_{0x}^2} + \frac{E_y^2(t)}{E_{0y}} - 2\frac{E_x(t)}{E_{0x}}\frac{E_y(t)}{E_{0xy}}\cos(\delta) = \sin^2(\delta). \tag{A.37}$$

La ecuación (A.37) representa una elipse para cada tiempo fijo t. Para reescribirla en función de una cantidad medible debemos tomar el promedio sobre un periodo de observación T, que consideraremos mucho mayor al período de oscilación de la onda. Podemos suponer T infinito y considerar los siguientes cálculos:

$$\langle E_i(t)E_j(t)\rangle = \lim_{T\to\infty}\frac{1}{T}\int_0^T E_i(t)E_j(t)dt. \tag{A.38}$$

Promediando ambos lados de la ecuación (A.37):

$$\frac{\langle E_x^2(t)\rangle}{E_{0x}^2} + \frac{\langle E_y^2(t)\rangle}{E_{0y}^2} - 2\frac{\langle E_x(t)E_y(t)\rangle}{E_{0x}E_{0y}}\cos(\delta) = \sin^2(\delta). \tag{A.39}$$

Los promedios de interés se calculan:

$$\langle E_x^2(t)\rangle = \lim_{T\to\infty}\frac{1}{T}\int_0^T E_{0x}^2\cos^2(\omega t + \delta_x)dt = \frac{E_{0x}^2}{2}, \tag{A.40}$$

$$\langle E_y^2(t)\rangle = \lim_{T\to\infty}\frac{1}{T}\int_0^T E_{0y}^2\cos^2(\omega t + \delta_y)dt = \frac{E_{0y}^2}{2}, \tag{A.41}$$

$$\begin{aligned}\langle E_x(t)E_y(t)\rangle &= \lim_{T\to\infty}\frac{1}{T}\int_0^T E_{0x}E_{0y}\cos(\omega t + \delta_x)\cos(\omega t + \delta_y)dt \\ &= \frac{1}{2}E_{0x}E_{0y}\cos(\delta).\end{aligned} \tag{A.42}$$

Sustituyendo los promedios en la ecuación (A.39):

$$\frac{E_{0x}^2}{2E_{0x}^2} + \frac{E_{0y}^2}{2E_{0y}^2} - \frac{E_{0x}E_{0y}\cos(\delta)}{E_{0x}E_{0y}}\cos(\delta) = \sin^2(\delta). \tag{A.43}$$

Multiplicando por $4E_{0x}^2E_{0y}^2$ a ambos lados de la ecuación y completando cuadrados, podemos llegar al siguiente resultado:

$$(E_{0x}^2 + E_{0y}^2)^2 - (E_{0x}^2 - E_{0y}^2)^2 - (2E_{0x}E_{0y}\cos(\delta))^2 = (2E_{0x}E_{0y}\sin(\delta))^2. \tag{A.44}$$

Los términos dentro de los paréntesis son los parámetros de Stokes:

$$s_0 = E_{0x}^2 + E_{0y}^2, \tag{A.45}$$

$$s_1 = E_{0x}^2 - E_{0y}^2, \tag{A.46}$$

$$s_2 = 2E_{0x}E_{0y}\cos(\delta), \tag{A.47}$$

$$s_3 = 2E_{0x}E_{0y}\sin(\delta). \tag{A.48}$$

La ecuación (A.44) se puede reescribir:

$$s_0^2 = s_1^2 + s_2^2 + s_3^2. \tag{A.49}$$

Recordando la notación fasorial de las componentes de campo eléctrico:

$$E_x = E_{0x}e^{i\delta_x}, \quad E_y = E_{0y}e^{i\delta_y}. \tag{A.50}$$

Podemos reescribir los parámetros de Stokes en notación fasorial:

$$s_0 = E_{0x}E_{0x}^* + E_{0y}E_{0y}^*, \tag{A.51}$$
$$s_1 = E_{0x}E_{0x}^* - E_{0y}E_{0y}^*, \tag{A.52}$$
$$s_2 = E_{0x}E_{0y}^* + E_{0y}E_{0x}^*, \tag{A.53}$$
$$s_3 = i(E_{0x}E_{0y}^* - E_{0y}E_{0x}^*). \tag{A.54}$$

A.4. Elipse de Polarización

La ecuación (A.37) representa una elipse para cada tiempo fijo t. Para verificarlo, definamos el propagador $\tau = \omega t - kz$ y reescribamos la ecuación (A.26):

$$E_x = E_{0_x}\cos(\tau + \delta_x) = E_{0_x}\cos(\tau)\cos(\delta_x) - E_{0_x}\sin(\tau)\sin(\delta_x), \tag{A.55}$$
$$E_y = E_{0_y}\cos(\tau + \delta_y) = E_{0_y}\cos(\tau)\cos(\delta_y) - E_{0_y}\sin(\tau)\sin(\delta_y). \tag{A.56}$$

Consideremos ahora las variables auxiliares E'_x, E'_y definidas como una rotación de ángulo ψ de las variables E_x, E_y:

$$E'_x = E_x\cos(\psi) + E_y\sin(\psi), \tag{A.57}$$
$$E'_y = -E_x\sin(\psi) + E_y\cos(\psi). \tag{A.58}$$

Impongamos que E'_x, E'_y describan una elipse no rotada:

$$E'_x = a\cos(\tau + \delta') = a\cos(\tau)\cos(\delta') - a\sin(\tau)\sin(\delta'), \tag{A.59}$$
$$E'_y = b\cos(\tau + \delta') = b\cos(\tau)\cos(\delta') - b\sin(\tau)\sin(\delta'). \tag{A.60}$$

Sustituyendo las definiciones de E_x, E_y, E'_x y E'_y en (A.57) y (A.58):

$$\begin{aligned} a(\cos(\tau)\cos(\delta') - \sin(\tau)\sin(\delta')) = {} & E_{0_x}(\cos(\tau)\cos(\delta_x) - \sin(\tau)\sin(\delta_x))\cos(\psi) \\ & + E_{0_y}(\cos(\tau)\cos(\delta_y) - \sin(\tau)\sin(\delta_y))\sin(\psi). \end{aligned} \tag{A.61}$$

$$\begin{aligned} b(\sin(\tau)\cos(\delta') - \cos(\tau)\sin(\delta')) = {} & -E_{0_x}(\cos(\tau)\cos(\delta_x) - \sin(\tau)\sin(\delta_x))\sin(\psi) \\ & + E_{0_y}(\cos(\tau)\cos(\delta_y) - \sin(\tau)\sin(\delta_y))\cos(\psi). \end{aligned} \tag{A.62}$$

Igualando las componentes que acompañan a $\cos(\tau)$ y $\sin(\tau)$:

$$a\cos(\delta') = E_{0_x}\cos(\delta_x)\cos(\psi) + E_{0_y}\cos(\delta_y)\sin(\psi), \tag{A.63}$$
$$a\sin(\delta') = E_{0_x}\sin(\delta_x)\cos(\psi) + E_{0_y}\sin(\delta_y)\sin(\psi), \tag{A.64}$$
$$b\cos(\delta') = E_{0_x}\sin(\delta_x)\sin(\psi) - E_{0_y}\sin(\delta_y)\cos(\psi), \tag{A.65}$$
$$b\sin(\delta') = E_{0_x}\cos(\delta_x)\sin(\psi) - E_{0_y}\cos(\delta_y)\cos(\psi). \tag{A.66}$$

Elevando al cuadrado las ecuaciones (A.63) - (A.64) y sumando:

$$a^2 = E_{0_x}^2 \cos^2(\psi) + E_{0_y}^2 \sin^2(\psi) + 2E_{0_x}E_{0_y}\cos(\psi)\sin(\psi)\cos(\delta). \tag{A.67}$$

Análogamente con (A.65) - (A.66):

$$b^2 = E_{0_x}^2 \sin^2(\psi) + E_{0_y}^2 \cos^2(\psi) - 2E_{0_x}E_{0_y}\cos(\psi)\sin(\psi)\cos(\delta). \tag{A.68}$$

Si sumamos (A.67) y (A.68):

$$a^2 + b^2 = E_{0_x}^2 + E_{0_y}^2. \tag{A.69}$$

Multiplicando (A.63) por (A.64), (A.65) por (A.66) y sumando:

$$ab = E_{0_x}E_{0_y}\sin(\delta). \tag{A.70}$$

Elevando al cuadrado (A.70) e igualando con el producto de (A.67) por (A.68), podemos llegar a la siguiente relación:

$$(E_{0_x}^2 - E_{0_y}^2)\sin(2\psi) = 2E_{0_x}E_{0_y}\cos(2\psi)\cos(\delta). \tag{A.71}$$

O dicho de otra forma:

$$\tan(2\psi) = \frac{2E_{0_x}E_{0_y}\cos(\delta)}{E_{0_x}^2 - E_{0_y}^2} = \frac{s_2}{s_1}. \tag{A.72}$$

Hallando así una relación entre el ángulo de polarización ψ y las características de la onda.

Otro parámetro de interés es la elipticidad χ, definida como:

$$\tan(\chi) = \frac{\sin(\chi)}{\cos(\chi)} = \frac{b}{a}. \tag{A.73}$$

Dividiendo el doble de (A.70) por (A.69):

$$\frac{2ab}{a^2 + b^2} = \frac{2E_{0_x}E_{0_y}\sin(\delta)}{E_{0_x}^2 + E_{0_y}^2}. \tag{A.74}$$

Utilizando la definición de χ, llegamos a otro importante resultado:

$$\sin(2\chi) = \frac{2E_{0_x}E_{0_y}\sin(\delta)}{E_{0_x}^2 + E_{0_y}^2} = \frac{s_3}{s_0}. \tag{A.75}$$

A.5. Cálculo Matemático de Matrices de Mueller

Dado un campo incidente $\boldsymbol{E}$ que incide sobre un medio con matriz de Mueller $\boldsymbol{M}$, consideremos el campo transmitido $\boldsymbol{E'}$ que se transforma en función de las propiedades del medio:

$$\boldsymbol{E} = E_x\hat{x} + E_y\hat{y}, \tag{A.76}$$

$$\boldsymbol{E'} = E'_x\hat{x} + E'_y\hat{y}. \tag{A.77}$$

En esta sección analizaremos los principales elementos polarizadores y cómo se describen sus respectivas matrices de Mueller.

A.5.1. Rotador

En este caso, las componentes ortogonales del campo transmitido se rotan un ángulo θ respecto a las componentes ortogonales del campo incidente, es decir:

$$E'_x = \cos(\theta)E_x + \sin(\theta)E_y, \tag{A.78}$$
$$E'_y = -\sin(\theta)E_x + \cos(\theta)E_y. \tag{A.79}$$

Aplicando las ecuaciones (A.51)-(A.54) sobre el campo transmitido:

$$E'^*_x E'_x = \cos^2(\theta)E^*_x E_x + \sin^2(\theta)E^*_y E_y + \sin(\theta)\cos(\theta)(E^*_y E_x + E^*_x E_y), \tag{A.80}$$
$$E'^*_y E'_y = \cos^2(\theta)E^*_y E_y + \sin^2(\theta)E^*_x E_x - \sin(\theta)\cos(\theta)(E^*_x E_y + E^*_y E_x), \tag{A.81}$$
$$E'^*_x E'_y = \cos^2(\theta)E^*_x E_y - \sin^2(\theta)E_x E^*_y - \sin(\theta)\cos(\theta)(E^*_x E_x - E^*_y E_y), \tag{A.82}$$
$$E'^*_y E'_x = \cos^2(\theta)E^*_y E_x - \sin^2(\theta)E_y E^*_x + \sin(\theta)\cos(\theta)(E^*_y E_y - E^*_x E_x). \tag{A.83}$$

Los parámetros de Stokes de la onda transmitida se calculan:

$$S'_0 = E'^*_x E'_x + E'^*_y E'_y = E^*_x E_x + E^*_y E_y = S_0, \tag{A.84}$$
$$\begin{aligned} S'_1 &= E'^*_x E'_x - E'^*_y E'_y = \\ &= \cos(2\theta)(E^*_x E_x - E^*_y E_y) + \sin(2\theta)(E^*_x E_y + E^*_y E_x) = \\ &= \cos(2\theta)S_1 + \sin(2\theta)S_2, \end{aligned} \tag{A.85}$$
$$\begin{aligned} S'_2 &= E'^*_x E'_y + E'^*_y E'_x = \\ &= \cos(2\theta)(E^*_x E_y + E^*_y E_x) - \sin(2\theta)(E^*_x E_x - E^*_y E_y) = \\ &= \cos(2\theta)S_2 - \sin(2\theta)S_1, \end{aligned} \tag{A.86}$$
$$S'_3 = i(E'_x E'^*_y - E'_y E'^*_x) = i(E^*_x E_y - E^*_y E_x) = S_3. \tag{A.87}$$

Entonces, la relación entre los parámetros de Stokes queda:

$$\boldsymbol{S'} = \begin{pmatrix} S'_0 \\ S'_1 \\ S'_2 \\ S'_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(2\theta) & \sin(2\theta) & 0 \\ 0 & -\sin(2\theta) & \cos(2\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix} = \boldsymbol{M}_{Rot}(-2\theta)\boldsymbol{S}. \tag{A.88}$$

La matriz de Mueller del rotador es útil para hallar las matrices de Mueller de los demás elementos polarizadores rotados. Si tomamos un rayo de luz con vector de Stokes $\boldsymbol{S}$ y lo rotamos un ángulo θ hacia el eje $\hat{x}$:

$$\boldsymbol{S'} = \boldsymbol{M}_{Rot}(-2\theta)\boldsymbol{S}. \tag{A.89}$$

Si el rayo con vector de Stokes $\boldsymbol{S'}$ atraviesa un medio con matriz de Mueller $\boldsymbol{M}$, la operación se escribe:

$$\boldsymbol{S''} = \boldsymbol{M}\boldsymbol{S'} = \boldsymbol{M}\boldsymbol{M}_{Rot}(-2\theta)\boldsymbol{S}. \tag{A.90}$$

Finalmente, rotamos en sentido opuesto a la primera rotación para obtener la relación matricial entre los vectores de Stokes en las coordenadas originales:

$$\boldsymbol{S'''} = \boldsymbol{M}_{Rot}(2\theta)\boldsymbol{M}\boldsymbol{S'} = \boldsymbol{M}_{Rot}(2\theta)\boldsymbol{M}\boldsymbol{M}_{Rot}(-2\theta)\boldsymbol{S}. \tag{A.91}$$

Entonces, la matriz de Mueller $\boldsymbol{M'}$ de un elemento polarizador con matriz de Mueller $\boldsymbol{M}$ rotado un ángulo θ será:

$$\boldsymbol{M'} = \boldsymbol{M}_{Rot}(2\theta)\boldsymbol{M}\boldsymbol{M}_{Rot}(-2\theta). \tag{A.92}$$

A.5.2. Polarizador Lineal

En este caso, las componentes ortogonales del campo transmitido se atenúan según los factores de atenuación p_x y p_y asociados a los ejes $\hat{x}$ e $\hat{y}$ respectivamente:

$$E'_x = p_x E_x, \tag{A.93}$$
$$E'_y = p_y E_y. \tag{A.94}$$

Aplicando las ecuaciones (A.51)-(A.54) sobre el campo transmitido:

$$E'^*_x E'_x = p_x^2 E^*_x E_x, \tag{A.95}$$
$$E'^*_y E'_y = p_y^2 E^*_y E_y, \tag{A.96}$$
$$E'^*_x E'_y = p_x p_y E^*_x E_y, \tag{A.97}$$
$$E'^*_y E'_x = p_x p_y E_x E^*_y. \tag{A.98}$$

Los parámetros de Stokes de la onda transmitida se calculan:

$$S'_0 = E'^*_x E'_x + E'^*_y E'_y = p_x^2 E^*_x E_x + p_y^2 E^*_y E_y = \frac{p_x^2}{2}(S_0 + S_1) + \frac{p_y^2}{2}(S_0 - S_1) = \\ = \frac{p_x^2 + p_y^2}{2} S_0 + \frac{p_x^2 - p_y^2}{2} S_1, \tag{A.99}$$

$$S'_1 = E'^*_x E'_x - E'^*_y E'_y = p_x^2 E^*_x E_x - p_y^2 E^*_y E_y = \frac{p_x^2}{2}(S_0 + S_1) - \frac{p_y^2}{2}(S_0 - S_1) = \\ = \frac{p_x^2 - p_y^2}{2} S_0 + \frac{p_x^2 + p_y^2}{2} S_1, \tag{A.100}$$

$$S'_2 = E'^*_x E'_y + E'^*_y E'_x = p_x p_y (E^*_x E_y + E_x E^*_y) = p_x p_y S_2, \tag{A.101}$$

$$S'_3 = i(E'_x E'^*_y - E'_y E'^*_x) = p_x p_y i (E^*_x E_y - E_x E^*_y) = p_x p_y S_3. \tag{A.102}$$

Entonces, la relación entre los parámetros de Stokes queda:

$$\boldsymbol{S'} = \begin{pmatrix} S'_0 \\ S'_1 \\ S'_2 \\ S'_3 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} p_x^2 + p_y^2 & p_x^2 - p_y^2 & 0 & 0 \\ p_x^2 - p_y^2 & p_x^2 + p_y^2 & 0 & 0 \\ 0 & 0 & 2p_x p_y & 0 \\ 0 & 0 & 0 & 2p_x p_y \end{pmatrix} \begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix} = \boldsymbol{M}_D \boldsymbol{S}. \tag{A.103}$$

La **diatenuación lineal** es una cantidad entre 0 y 1 definida:

$$D_L = \frac{p_x^2 - p_y^2}{p_x^2 + p_y^2}, \;\; p_x > p_y. \tag{A.104}$$

La matriz de Mueller del polarizador se puede reescribir:

$$\boldsymbol{M}_D = \frac{p_x^2 + p_y^2}{2} \begin{pmatrix} 1 & D_L & 0 & 0 \\ D_L & 1 & 0 & 0 \\ 0 & 0 & \sqrt{1 - D_L^2} & 0 \\ 0 & 0 & 0 & \sqrt{1 - D_L^2} \end{pmatrix}. \tag{A.105}$$

La matriz de Mueller del polarizador lineal rotado un ángulo θ queda:

$$\boldsymbol{M}_D = \frac{T}{2} \begin{pmatrix} 1 & D_L \cos(2\theta) & D_L \sin(2\theta) & 0 \\ D_L \cos(2\theta) & \alpha + (1-\alpha)\cos^2(2\theta) & (1-\alpha)\sin(2\theta)\cos(2\theta) & 0 \\ D_L \sin(2\theta) & (1-\alpha)\sin(2\theta)\cos(2\theta) & \alpha + (1-\alpha)\sin^2(2\theta) & 0 \\ 0 & 0 & 0 & \alpha \end{pmatrix}, \tag{A.106}$$

siendo T la transmitancia del polarizador:

$$T = p_x^2 + p_y^2. \tag{A.107}$$

A.5.3. Retardador Lineal

El retardador lineal es un dispositivo que cambia la fase del campo incidente, introduciendo un desfasaje ϕ entre sus componentes ortogonales. En este caso, las componentes del campo se transforman de la siguiente forma:

$$E'_x = E_x, \tag{A.108}$$

$$E'_y = e^{-i\phi} E_y. \tag{A.109}$$

Aplicando las ecuaciones (A.51)-(A.54) sobre el campo transmitido:

$$E'^*_x E'_x = E^*_x E_x, \tag{A.110}$$

$$E'^*_y E'_y = E^*_y E_y, \tag{A.111}$$

$$E'^*_x E'_y = e^{-i\phi} E^*_x E_y, \tag{A.112}$$

$$E'^*_y E'_x = e^{i\phi} E_x E^*_y. \tag{A.113}$$

Los parámetros de Stokes de la onda transmitida se calculan:

$$S'_0 = E'^*_x E'_x + E'^*_y E'_y = E^*_x E_x + E^*_y E_y = S_0, \tag{A.114}$$

$$S'_1 = E'^*_x E'_x - E'^*_y E'_y = E^*_x E_x - E^*_y E_y = S_1, \tag{A.115}$$

$$\begin{aligned} S'_2 &= E'^*_x E'_y + E'^*_y E'_x = e^{-i\phi} E^*_x E_y + e^{i\phi} E_x E^*_y = \\ &= (\cos(\phi) + i\sin(\phi)) E^*_x E_y + (\cos(\phi) - i\sin(\phi)) E_x E^*_y = \\ &= \cos(\phi)(E^*_x E_y + E_x E^*_y) + \sin(\phi) i (E^*_x E_y - E_x E^*_y) = \\ &= \cos(\phi) S_2 + \sin(\phi) S_3, \end{aligned} \tag{A.116}$$

$$\begin{aligned} S'_3 &= i(E'_x E'^*_y - E'_y E'^*_x) = i(e^{-i\phi} E^*_x E_y - e^{i\phi} E_x E^*_y) = \\ &= \cos(\phi) i (E^*_x E_y - E_x E^*_y) - \sin(\phi)(E^*_x E_y + E_x E^*_y) = \\ &= \cos(\phi) S_3 - \sin(\phi) S_2. \end{aligned} \tag{A.117}$$

(A.118)

Entonces, la relación entre los parámetros de Stokes queda:

$$\boldsymbol{S}' = \begin{pmatrix} S_0' \\ S_1' \\ S_2' \\ S_3' \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\phi) & \sin(\phi) \\ 0 & 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix} \begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix} = \boldsymbol{M}_R \boldsymbol{S}. \tag{A.119}$$

La matriz de Mueller del retardador lineal con eje rápido en $\hat{x}$ rotado un ángulo θ se calcula:

$$\boldsymbol{M}_R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & (1-\cos(\phi))\cos^2(\theta)+\cos(\phi) & (1-\cos(\phi))\sin(2\theta)\cos(2\theta) & -\sin(\phi)\sin(2\theta) \\ 0 & (1-\cos(\phi))\sin(2\theta)\cos(2\theta) & (1-\cos(\phi))\sin^2(2\theta)+\cos(\phi) & \sin(\phi)\cos(2\theta) \\ 0 & \sin(\phi)\sin(2\theta) & -\sin(\phi)\cos(2\theta) & \cos(\phi) \end{pmatrix}. \tag{A.120}$$

A.6. Descomposición de Lu-Chipman

Si $\boldsymbol{M}$ es una matriz de Mueller no singular, existe una única descomposición que tiene la siguiente forma:

$$\boldsymbol{M} = m_{00} \boldsymbol{M_\Delta} \boldsymbol{M_R} \boldsymbol{M_D}, \tag{A.121}$$

dónde m_{00} es la transmitancia bajo luz despolarizada, $\boldsymbol{M_\Delta}$ es la matriz de despolarizancia, $\boldsymbol{M_R}$ la matriz de retardancia y $\boldsymbol{M_D}$ la matriz de diatenuación.

Para probarlo, definamos una matriz auxiliar $\boldsymbol{M}'$ como:

$$\boldsymbol{M}' = \boldsymbol{M}\boldsymbol{M}_{\boldsymbol{D}}^{-1}. \tag{A.122}$$

Como $\boldsymbol{M}$ es no singular, podemos invertir la matriz de diatenuación:

$$\boldsymbol{M}_D^{-1} = \frac{1}{1-D^2} \begin{pmatrix} 1 & -\boldsymbol{D}^T \\ -\boldsymbol{D} & \sqrt{1-D^2}\boldsymbol{I} + \frac{1}{(1+\sqrt{1-D^2})}(\boldsymbol{D}\otimes\boldsymbol{D}) \end{pmatrix}, \tag{A.123}$$

siendo $\boldsymbol{D}$ el vector de diatenuación:

$$\boldsymbol{D} = \frac{1}{m_{00}} \begin{pmatrix} m_{11} \\ m_{22} \\ m_{33} \end{pmatrix}. \tag{A.124}$$

Multiplicando $\boldsymbol{M}_\Delta$ y $\boldsymbol{M}_R$ por un lado:

$$\boldsymbol{M}_\Delta \boldsymbol{M}_R = \begin{pmatrix} 1 & \boldsymbol{0}^T \\ \boldsymbol{P}_\Delta & \boldsymbol{m}_\Delta \end{pmatrix} \begin{pmatrix} 1 & \boldsymbol{0}^T \\ \boldsymbol{0} & \boldsymbol{m}_R \end{pmatrix} = \begin{pmatrix} 1 & \boldsymbol{0}^T \\ \boldsymbol{P}_\Delta & \boldsymbol{m}_\Delta \boldsymbol{m}_R \end{pmatrix}. \tag{A.125}$$

Multiplicando $\boldsymbol{M}$ y $\boldsymbol{M}_D^{-1}$ por otra parte:

$$\begin{aligned} \boldsymbol{M}\boldsymbol{M}_D^{-1} &= \frac{m_{00}}{1-D^2} \begin{pmatrix} 1 & \boldsymbol{D}^T \\ \boldsymbol{P} & \boldsymbol{m} \end{pmatrix} \begin{pmatrix} 1 & -\boldsymbol{D}^T \\ -\boldsymbol{D} & \sqrt{1-D^2}\boldsymbol{I} + \frac{1}{(1+\sqrt{1-D^2})}(\boldsymbol{D}\otimes\boldsymbol{D}) \end{pmatrix} \\ &= m_{00} \begin{pmatrix} 1 & \boldsymbol{0}^T \\ \frac{\boldsymbol{P}-\boldsymbol{m}\boldsymbol{D}}{1-D^2} & \boldsymbol{m}' \end{pmatrix}. \end{aligned} \tag{A.126}$$

Imponiendo Lu-Chipman, se deben cumplir las igualdades:

$$\boldsymbol{P}_\Delta = \frac{\boldsymbol{P} - \boldsymbol{m}\boldsymbol{D}}{1 - D^2}, \quad \boldsymbol{m}' = \boldsymbol{m}_\Delta \boldsymbol{m}_R. \tag{A.127}$$

Definamos una matriz auxiliar $\boldsymbol{n}$ de la siguiente forma:

$$\boldsymbol{n} = (\boldsymbol{m}')^T \boldsymbol{m}' = \boldsymbol{m}_\Delta \boldsymbol{m}_R \boldsymbol{m}_R^T \boldsymbol{m}_\Delta^T = \boldsymbol{m}_\Delta^2. \tag{A.128}$$

Como $\boldsymbol{m}_\Delta$ es simétrica, $\boldsymbol{n}$ es diagonalizable y tiene valores propios reales positivos:

$$\boldsymbol{n} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1}. \tag{A.129}$$

Si λ_1, λ_2 y λ_3 son los valores propios de $\boldsymbol{n}$, entonces:

$$\boldsymbol{m}_\Delta = \boldsymbol{n}^{\frac{1}{2}} = \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}\boldsymbol{U}^{-1}. \tag{A.130}$$

Solo queda determinar $\boldsymbol{m}_R$:

$$\boldsymbol{m}_R = \boldsymbol{m}_\Delta^{-1} \boldsymbol{m}'. \tag{A.131}$$

Apéndice B

Transmitancia de Muestras Escogidas

B.1. Muestras de Entrenamiento

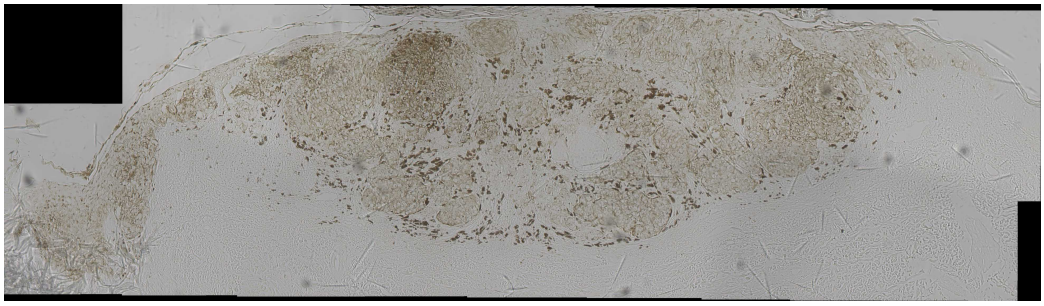

Figura B.1: Melanoma 1 a muestra completa.

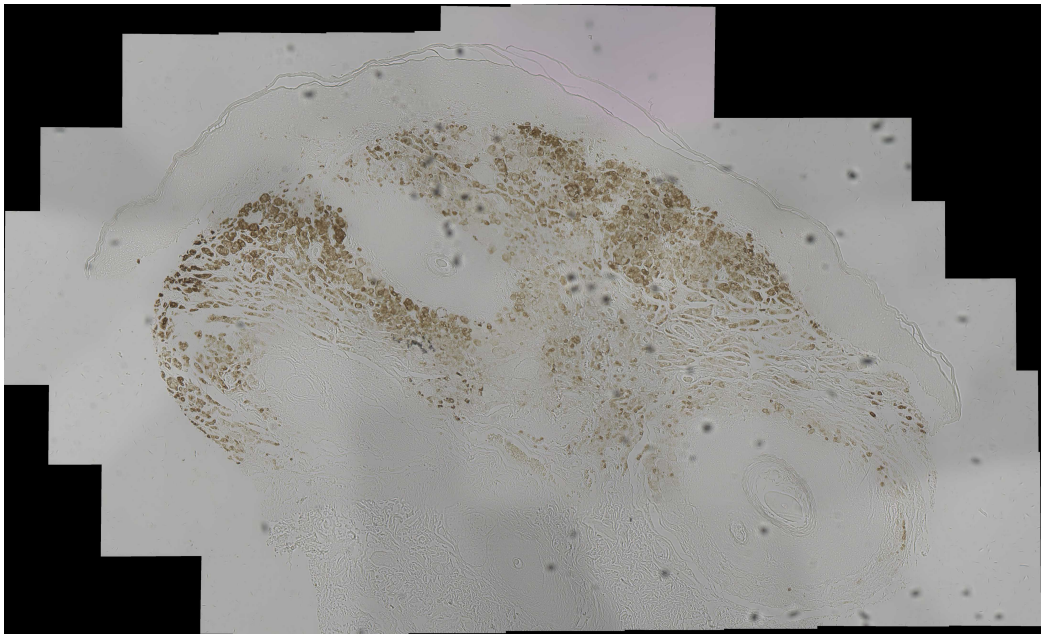

Figura B.2: Nevo 1 a muestra completa.

B.2. Muestras Adicionales

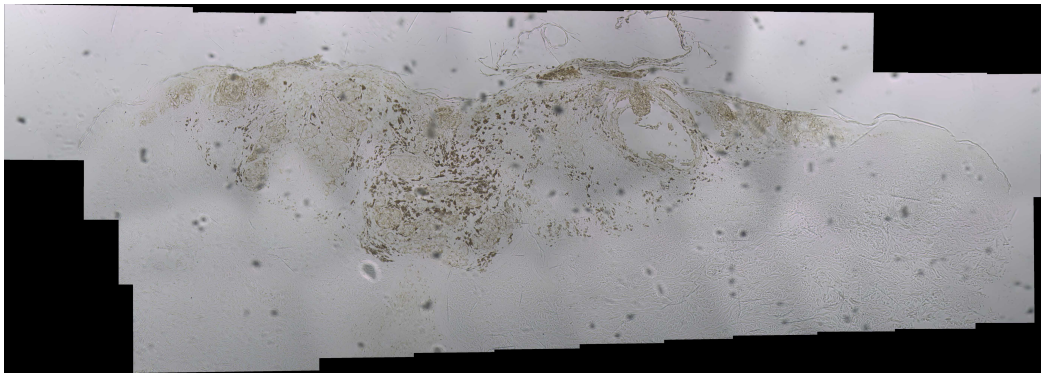

Figura B.3: Melanoma 2 a muestra completa.

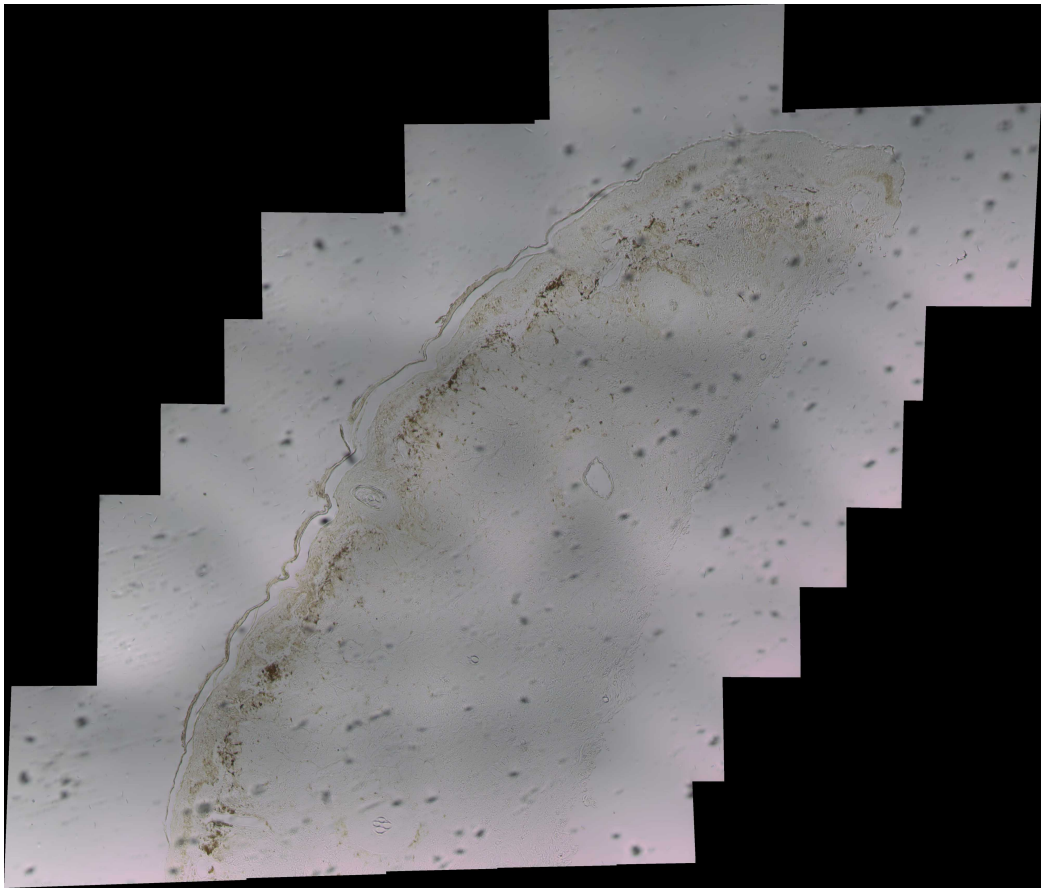

Figura B.4: Nevo 2 a muestra completa.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice C

Algoritmos de *Stitching*

```
def compose_panoramic(cameras, images, medium_imgs, low_imgs, final_imgs,
                      crop = True, plots = False, warper_type = 'Plane'):
    """
    Compone una imagen panorámica con imágenes y parámetros de entrada.

    Parámetros:
    - cameras: Lista de objetos de cámaras estimadas para cada imagen.
    - images: Objeto de imágenes de entrada.
    - medium_imgs: Lista de imágenes de resolución media.
    - low_imgs: Lista de imágenes de baja resolución.
    - final_imgs: Lista de imágenes de resolución final.
    - crop: Booleano para recortar imagen panorámica (por defecto, True).
    - plots: Booleano para gráficos de visualización (por defecto, False).
    - warper_type: Tipo de warper a utilizar (por defecto, 'Plane').

    Devuelve:
    - panorama: Imagen panorámica compuesta.
    """

    #(...) Alineamiento

    #(...) Recortado

    #(...) Costuras

    #(...) Fusión

    return panorama
```

```
def registration(img_list, stitching_config, indices = None, train = True,
                 plots = False):
  """
   Recibe imágenes de entrada y realiza proceso de Registro, devolviendo
   imágenes y parámetros necesarios para la Composición.

    Parámetros:
    - img_list: Lista de imágenes de entrada.
    - stitching_config: Configuración de parámetros para el stitching.
    - indices: Índices de imágenes seleccionadas  (por defecto, None).
    - train: Booleano para entrenamiento (por defecto, True).
    - plots: Booleano para gráficos  (por defecto, False).

    Devuelve:
    - cameras: Lista de objetos de cámaras estimadas.
    - indices: Índices de las imágenes seleccionadas por el subsetter.
    - images: Objeto de imágenes de entrada actualizado.
    - medium_imgs: Lista de imágenes de resolución media.
    - low_imgs: Lista de imágenes de baja resolución.
    - final_imgs: Lista de imágenes de resolución final.
  """

  #(...) Preprocesamiento

  #Entrenamiento
  if train:

    #(...) Detección de Características

    #(...) Emparejamiento de Características

    #(...) Selección de Subconjunto

    #(...) Estimación de Homografía

    return cameras, indices, images, medium_imgs, low_imgs, final_imgs

  #Ya entrenado, con cameras e indices guardados previamente
  else:

    #(...) Selección de Subconjunto

    return images, medium_imgs, low_imgs, final_imgs
```

```
def load_cameras(cam_list):
    """
    Carga los parámetros de las cámaras desde una lista serializable.

    Parámetros:
    - cam_list: Lista de diccionarios con los parámetros de las cámaras
    en formato serializable.
    Devuelve:
    - Lista de objetos CameraParams.
    """

    #Lista de cámaras
    loaded_cams = []

    #Para cada cámara en la lista, carga parámetros
    for serialized_camera in cam_list:
        loaded_camera = cv2.detail.CameraParams()
        loaded_camera.focal = serialized_camera['focal']
        loaded_camera.aspect = serialized_camera['aspect']
        loaded_camera.ppx = serialized_camera['ppx']
        loaded_camera.ppy = serialized_camera['ppy']
        loaded_camera.R = serialized_camera['R']
        loaded_camera.t = serialized_camera['t']
        loaded_camera.K = np.array(serialized_camera['K']) if
                          serialized_camera['K'] is not None else None

        #Anexa cámara en la lista
        loaded_cams.append(loaded_camera)

    return loaded_cams
```

```
def save_cameras(cameras):
    """
    Guarda los parámetros de las cámaras en un formato serializable.

    Parámetros:
    - cameras: Lista de objetos CameraParams.
    Devuelve:
    - Lista de diccionarios con los parámetros de las cámaras
    en formato serializable.
    """

  # Lista de cámaras
  serializable_cameras = []

  # Para cada cámara en la lista, guarda parámetros
  for camera in cameras:
    serializable_camera = {
        'focal': camera.focal,
        'aspect': camera.aspect,
        'ppx': camera.ppx,
        'ppy': camera.ppy,
        'R': camera.R,
        't': camera.t,
        'K': camera.K().tolist() if isinstance(camera.K(), np.ndarray)
                                 else None
    }

    #Anexa cámara en la lista
    serializable_cameras.append(serializable_camera)

  return serializable_cameras
```

Apéndice D

Lista de Materiales

D.1. Electrónica y Mecánica

Nombre	Cantidad
Raspberry Pi 2	1
Acople Motor 5×5	1
Motor Nema17	2
Motor BYJ48	1
Fuente DC de $5V$ micro USB	1
Fuente DC de $5V$ CCTV	1
Kit cables Jumper H-H	1
Kit cables Jumper M-H	1
Conector CCTV Macho	1
Driver ULN2003	1
Driver L298N	2
Tarjeta Micro SD 8GB	1
Zapatilla de 3 entradas	1
Impresora Ultimaker 3	1

Tabla D.1: Lista de materiales: Electrónica.

Nombre	Cantidad
Barra Hilada 6 mm (1 metro)	1
Barra Lisa 4 mm (1 metro)	1
Filamento PLA 3D	1
Tornillo, arandela y tuerca	Varios
Anillo para objetivo	1
Mesa metálica con agujeros	1

Tabla D.2: Lista de materiales: Mecánica.

D.2. Modelos STL

Nombre Archivo (.stl)	Cantidad	Descripción
Esquina_Union	4	Diseñado en Blender.
Pata	4	Diseñado en Blender.
Acople5_4mm	1	Diseñado en Blender.
Soporte_Guias	1	Diseñado en Blender.
PortaMuestras	1	Diseñado en Blender.
DeslizadorH	1	Diseñado en Blender.
DeslizadorM	1	Diseñado en Blender.
PortaObjetivo	2	Diseñado en Blender.
PortaEspejo_Cover	1	Diseñado en Blender.
Iluminador_Base	1	Diseñado en Blender.
Iluminador_Condenser_Cover	1	Diseñado en Blender.
Iluminado_Colector_Cover	1	Diseñado en Blender.
Raspberry_Pi_Union	1	Diseñado en Fusion360.
ULN2003_Union	1	Diseñado en Fusion360.
PortaCamara	1	Diseñado en Blender.
PortaEspejo_Base	1	Diseñado en Blender.
PortaMotorX	1	Acondicionado en Blender.
PortaMotorY	1	Acondicionado en Blender.
Soporte_Perfil	1	Diseño descargado.
ULN2003_Base	1	Diseño descargado.
ULN2003_Cover	1	Diseño descargado.
L298N_Double_Case_Base	1	Diseño descargado.
L298N_Double_Case_Cover	1	Diseño descargado.
Raspberry_Pi_Case_Bottom	1	Diseño descargado.
Raspberry_Pi_Case_Cover	1	Diseño descargado.

Tabla D.3: Lista de materiales: Impresiones.

Nombre Archivo (.stl)	Descripción
NEMA_17	Diseño descargado.
stepper	Diseño descargado.
condenser	Diseñado con OptiCore.
colector	Diseñado con OptiCore.
tube_lens	Diseñado con OptiCore.

Tabla D.4: Lista de materiales: Otros modelos utilizados, no impresos.

D.3. Óptica

Nombre	Cantidad	Código
Sensor Polarizado BFS-U3-51S5P-C	1	Edmund Optics #13-928
Doblete Acromático 25 × 200 mm	1	Edmund Optics #47-645
Doblete Acromático 25 × 25 mm	1	Edmund Optics #65-553
Objetivo Olympus UPLFLN 10X	1	Edmund Optics #86-818
Target de Calibración	1	Edmund Optics #37-539
Kit de Tubos de extensión	1	Thor Labs CML-KIT
Base Magnética	1	Thor Labs PHM1
Poste metálico	2	Thor Labs TR4
Módulo LED	1	Thor Labs M405LP1
Ángulo 90°	1	Thor Labs RA90
Espejo	1	Thor Labs PFR10-P01

Tabla D.5: Lista de materiales: Óptica.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice E

Descripción de Funciones

```
I0, I45, I90, I135 = polarization_full_dec_array(A, interpolador)
```

Convierte la información cruda de la cámara en una una imagen de cuatro canales de polarización y tres canales de color. Toma como argumento un arreglo A de 2048 x 2448 enteros sin signo de 8 bits y un string interpolación, que puede tomar como valor: 'vecinos' o 'bilineal'. Retorna los cuatro medibles I90, I45, I135, I0 como arreglos de 1024 x 1224 x 3 enteros de 8 bits sin signo y codificación de color BGR.

```
S0, S1, S2 = calcular_stokes(I90, I45, I135, I0, interpolador)
```

Calcula los parametros de Stokes. Toma como argumento los cuatro medibles I90, I45, I135, I0 y la técnica de interpolacion. Retorna los parámetros de Stokes S0, S1 y S2 como arreglos 1024 x 1224 x 3 de enteros de 16 bits con signo y codificación de color BGR.

```
M = calcular_mueller_inv(S_in_stat_inv, S_out_stat)
```

Calcula matriz de Mueller. Toma como argumento la Matriz de Stokes de entrada invertida S_in_stat_inv y la Matriz de Stokes de salida S_out_stat. Retorna M como un arreglo de 1024 x 1224 x 3 x 3 x 3 [pxy, pxx, color, row, col] puntos flotantes de 64 bits y codificación de color BGR.

```
M_norm = normalizar_mueller(M)
```

Normaliza la Matriz de Mueller M. Toma como argmuento la Matriz de Mueller M. Retorna la Matriz de Mueller normalizada M_norm.

```
M_show = acoplar_mueller(M)
```

Convierte una Matriz de Mueller 1024 x 1224 x 3 x 3 x 3 en un arreglo 3072 x 3672 x 3. Toma como argumento la matriz de Mueller M. Retorna una imagen compuesta por las nueve componentes de la Matriz mostradas de forma simultánea y codificación de color BGR.

```
M = desacoplar_mueller(M_show)
```

Es la inversa de la función acoplar_mueller. Toma como argumento la imagen en formato .png. Retorna un arreglo representando la matriz de Mueller como punto flotante y codificación de color BGR.

```
output = ejecutar_comando_ssh(comando)
```

Guarda en la Raspberry un archivo ejecutable con los comandos de los motores y ejecuta un comando a elección de el usuario. Toma como argumento el comando. Retorna la salida de el ejecutable.

```
S = take_stokes(cam_config)
```

Captura el Vector de Stokes visto por la cámara. Toma como argumento el tiempo de exposición y el número de imágenes promediadas. Retorna el Vector de Stokes en formato tensorial como punto flotante.

```
M = take_mueller(S_in_stat_inv, cam_config, path, thetas_list)
```

Captura la Matriz de Mueller vista por la cámara. Toma como argumento la Matriz de Stokes de entrada invertida, el tiempo de exposición, el número de imágenes promediadas y una lista con los ángulos de rotación de el polarizador. Retorna la matriz de Mueller como punto flotante.

S = take_mueller_stokes(cam_config, thetas_list)

Captura la Matriz de Stokes visto por la cámara. Toma como argumento el tiempo de exposición, el número de imágenes promediadas y una lista con los ángulos de rotación de el polarizador. Retorna los tres vectores de Stokes en formato tensorial como punto flotante.

A_dig = digitalizar(A_analog, medida)

Digitaliza una medida física para poder ser representada en una imagen. Toma como argumento la medida en un arreglo A y el tipo de medida en formato de string. Medidas compatibles: 'S0', 'S1', 'S2', 'DoLP', 'AoLP', 'M8' (Mueller de 8 bits) y 'M16' (Mueller de 16 bits). Retorna la imagen lista para ser guardada.

A_analog = analogizar(A_dig, medida)

Es la inversa de la función digitalizar. Toma como argumento la medida en un arreglo A y el tipo de medida en formato de string. Medidas compatibles: 'S0', 'S1', 'S2', 'DoLP', 'AoLP', 'M8' (Mueller de 8 bits) y 'M16' (Mueller de 16 bits). Retorna un arreglo representando la medida físicamente.

guardar_mueller(M, path, name)

Guarda la matriz de Mueller en formato .png con una precisión de 16 bits. Además, guarda un colormap de cada canal de color con su respectiva barra. Toma como argumento la matriz de Mueller M, un string indicando la ruta dónde se van a guardar los archivos y un string indicando el nombre de la muestra.

M = png2mueller(M_show, medida)

Es la inversa de la función guardar_mueller. Toma como argumento la foto en formato .png y el tipo de medida que corresponda. Retorna un arreglo representando físicamente la Matriz de Mueller.

`motor_control(string_motor, string_movimiento)`

Controla los motores desde la Raspberry Pi. Recibe como argumentos un string indicando el motor y otro string indicando el tipo de movimiento. Los motores disponibles son 'X', 'Y' y 'T'. Los movimientos posibles son 'F' y 'B'. No retorna nada.

`motor_control_ssh(string_motor, string_movimiento)`

Controla la Raspberry desde la PC. Envía los comandos mediante SSH. No retorna nada.

`capturar_muestra(cam_config, S)`

Captura la muestra entera a visualizar. Toma como argumento la configuración de la cámara y el número S, dónde S^2 es el número de fotos que toma para hacer el escaneo. No retorna nada.

`simple_photo(cam_config)`

Toma el Vector de Stokes de lo que está observando la cámara. Toma como argumento la configuración de la cámara.

`simple_stokes(cam_config)`

Captura la Matriz de Stokes de entrada. La muestra no debe estar en la platina. Toma como argumento la configuración de la cámara. No retorna nada.

`MDelta, MR, MD, m00 = lu_chipman(M)`

Descompone una matriz de Mueller según la Descomposición de Lu-Chipman-Suami. Toma como argumento la Matriz de Mueller M. Retorna las matrices de Mueller MDelta, MR y MD, todas como punto flotante y codificación de color BGR.

`turn_off`

Apaga la Raspberry Pi.

Bibliografía

[1] Dennis H. Goldstein. *Polarized Light*. Florida, Estados Unidos: CRC Press, 2010. ISBN: 978-1-4398-3041-3.

[2] Valery V Tuchin. "Polarized light interaction with tissues". En: *Journal of biomedical optics* 21.7 (2016), pág. 071114. DOI: 10.1117/1.JBO.21.7.071114.

[3] Tatiana Novikova et al. "Polarimetric Imaging for Cancer Diagnosis and Staging". En: *Opt. Photon. News* 23.10 (2012), págs. 26-33. DOI: 10.1364/OPN.23.10.000026.

[4] J. Qi y D. S. Elson. "Mueller polarimetric imaging for surgical and diagnostic applications: a review." En: *Journal of biophotonics* 10.8 (2017), págs. 950-982. DOI: 10.1002/jbio.201600152.

[5] Russell A. Chipman y Shih-Yau Lu. "Decomposition of Mueller matrices". En: *Wideband Interferometric Sensing and Imaging Polarimetry*. Vol. 3120. International Society for Optics y Photonics. SPIE, 1997, págs. 385-396. DOI: 10.1117/12.283849.

[6] Leanne E Iannucci et al. "Effect of matrix properties on transmission and reflectance mode division-of-focal-plane Stokes polarimetry". En: *Journal of Biomedical Optics* 28.10 (2023), págs. 102902-102902. DOI: 10.1117/1.JBO.28.10.102902.

[7] Shengkui Gao y Viktor Gruev. "Bilinear and bicubic interpolation methods for division of focal plane polarimeters". En: *Opt. Express* 19.27 (2011), págs. 26161-26173. DOI: 10.1364/OE.19.026161.

[8] Davis LE, Shalin SC y Tackett AJ. "Current state of melanoma diagnosis and treatment". En: *Cancer biology and therapy* 20.11 (2019), págs. 1366-1379. DOI: 10.1080/15384047.2019.1640032.

[9] Anat Levin et al. *Seamless Image Stitching in the Gradient Domain*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, págs. 377-389. ISBN: 978-3-540-24673-2.

[10] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2019. ISBN: 1492032646.

[11] Ngan Luu et al. "Characterization of Mueller matrix elements for classifying human skin cancer utilizing random forest algorithm". En: *Journal of biomedical optics* 26 (2021). https://doi.org/10.1117/1.JBO.26.7.075001.

[12] Manuel J. Cabrera. *Óptica Electromagnética*. Madrid, España: Addison-Wesley, 1993. ISBN: 0-201-60132-X.

[13] Matcha Team. *Mathcha Editor*. URL: https://www.mathcha.io/.

[14] John Freudenthal. *Intuitive interpretation of Mueller matrices of transmission*. Inf. téc. 2018. URL: https://www.hindsinstruments.com/wp-content/uploads/Intuitive-interpretation-of-Mueller-matrices-of-transmission-J-Freudenthal-January-18-2018.pdf.

[15] Nirmalya Ghosh, Michael F. G. Wood e I. Alex Vitkin. "Mueller matrix decomposition for extraction of individual polarization parameters from complex turbid media exhibiting multiple scattering, optical activity, and linear birefringence". En: *Journal of Biomedical Optics* 13.4 (2008), págs. 044036-044036. DOI: 10.1117/1.2960934.

[16] juh. *Nema17 Case*. https://www.thingiverse.com/thing:3269804.

[17] huczekdesign. *Raspberry Pi Case*. https://www.printables.com/es/model/219727-raspberry-pi-case.

[18] ShmilCat. *L298N driver case*. https://www.printables.com/es/model/472016-double-l298n-case-with-zip-tie-closure-and-screw-h.

[19] VincentM. *ULN2003 Case*. https://www.thingiverse.com/thing:2317662.

[20] Blender Foundation. *Blender*. URL: https://www.blender.org/.

[21] EA3D. *Perfil Aluminio Model*. https://www.printables.com/es/model/303508-aluminium-profile-20x20/files.

[22] sale723. *Nema17 Model*. https://www.thingiverse.com/thing:4681240.

[23] PawArmy. *28BYJ-48 Model*. https://www.thingiverse.com/thing:4031757.

[24] Teledyne FLIR. *Manual Cámara BFS-U3-51S5*. URL: http://softwareservices.flir.com/BFS-U3-51S5/latest/Model/readme.html.

[25] Roman Demczylo y Diego Silva Piedra. *Repositorio MWSI*. 2023. URL: https://github.com/MWSI2023/MWSI.

[26] Dustin Kleckner. *An easy-to-use Pythonic wrapper for the FLIR PySpin Library*. 2019. URL: https://pypi.org/project/simple-pyspin/.

[27] R. Demczylo. *Informe de Pasantía*. Inf. téc. 2022. URL: https://www.pedeciba.edu.uy/admin/uploads/estatico/Fisica/Informe_Roman_Demczylo.pdf.

[28] Technologie Manufaktur. *Target de resolución TC-RT01*. URL: https://www.technologie-manufaktur.de/en/products/test-targets/#TC-RT01.

[29] MK Swami et al. "Polar decomposition of 3×3 Mueller matrix: a tool for quantitative tissue polarimetry". En: *Optics express* 14.20 (2006), págs. 9324-9337. DOI: 10.1364/oe.14.009324.

[30] *Muestras Biológicas de AmScope*. URL: https://amscope.com/products/ps25p.

[31] *Photoshop*. URL: https://www.adobe.com/la/products/photoshop.html.

[32] Riverbank Computing Limited. *Python bindings for the Qt cross platform application toolkit*. 2023. URL: https://pypi.org/project/PyQt5/.

[33] Ebtsam Adel, Mohammed Elmogy y Hazem El-Bakry. "Image Stitching based on Feature Extraction Techniques: A Survey". En: *International Journal of Computer Applications* 99 (ago. de 2014), págs. 1-8. DOI: 10.5120/17374-7818.

[34] David Benson. *drawio*. URL: https://app.diagrams.net/.

[35] Marsha J. Hannah. *Computer Matching of Areas in Stereo Images*. 2019.

[36] D. Capel y A. Zisserman. "Automated mosaicing with super-resolution zoom". En: *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*. 1998, págs. 885-891. DOI: 10.1109/CVPR.1998.698709.

[37] Brown y Lowe. "Recognising panoramas". En: *Proceedings Ninth IEEE International Conference on Computer Vision*. 2003, 1218-1225 vol.2. DOI: 10.1109/ICCV.2003.1238630.

[38] Quoc Do, Peter Lozo y Lakhmi Jain. "A Fast Visual Search and Recognition Mechanism for Real-Time Robotics Applications". En: vol. 3339. Dic. de 2004, págs. 937-942. ISBN: 978-3-540-24059-4. DOI: 10.1007/978-3-540-30549-1_82.

[39] Richard Szeliski. "Image Alignment and Stitching: A Tutorial". En: *Foundations and Trends in Computer Graphics and Vision* 2 (ene. de 2006). DOI: 10.1561/0600000009.

[40] Ertuğrul Bayraktar y Pınar Boyraz. "Analysis of feature detector and descriptor combinations with a localization experiment for various performance metrics". En: *Turkish Journal of Electrical Engineering and Computer Sciences* 25 (2017), págs. 2444-2454. DOI: 10.3906/elk-1602-225.

[41] Antti Hietanen et al. "A comparison of feature detectors and descriptors for object class matching". En: *Neurocomputing* 184 (2016), págs. 3-12. DOI: 10.1016/j.neucom.2015.08.106.

[42] Frazer K. Noble. "Comparison of OpenCV's feature detectors and feature matchers". En: *2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. 2016, págs. 1-6. DOI: 10.1109/M2VIP.2016.7827292.

[43] Ives Rey Otero y Mauricio Delbracio. "Anatomy of the SIFT Method". En: *Image Processing On Line* 4 (2014). https://doi.org/10.5201/ipol.2014.82, págs. 370-396.

[44] Tony Lindeberg. *Scale-Space Theory in Computer Vision*. Dic. de 1993. ISBN: 0-7923-9418-6. DOI: 10.1007/978-1-4757-6465-9.

[45] Alexei A. Efros y William T. Freeman. "Image Quilting for Texture Synthesis and Transfer". En: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 1.ª ed. New York, NY, USA: Association for Computing Machinery, 2023. ISBN: 9798400708978. URL: https://doi.org/10.1145/3596711.3596771.

[46] Shai Avidan y Ariel Shamir. "Seam Carving for Content-Aware Image Resizing". En: *SIGGRAPH* 26 (jul. de 2007). DOI: 10.1145/1276377.1276390.

[47] Vladan Rankov et al. "An Algorithm for image stitching and blending". En: *Three-Dimensional and Multidimensional Microscopy: Image Acquisition and Processing XII*. Vol. 5701. International Society for Optics y Photonics. SPIE, 2005, págs. 190-199. DOI: 10.1117/12.590536.

[48] Lukas Weber. *Open Stitching*. 2023. URL: https://github.com/OpenStitching/stitching.

[49] Open Source Computer Vision. *Images stitching*. URL: https://docs.opencv.org/3.4/d1/d46/group__stitching.html.

[50] Pavan Chennagiri Madhusudana y Rajiv Soundararajan. "Subjective and Objective Quality Assessment of Stitched Images for Virtual Reality". En: *IEEE Transactions on Image Processing* 28.11 (2019), págs. 5620-5635. DOI: 10.1109/TIP.2019.2921858.

[51] Vipula Dissanayake et al. "Quantitative and Qualitative Evaluation of Performance and Robustness of Image Stitching Algorithms". En: *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. 2015, págs. 1-6. DOI: 10.1109/DICTA.2015.7371297.

[52] Gene Cheung et al. "A Content-Aware Metric for Stitched Panoramic Image Quality Assessment". En: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2017, págs. 2487-2494. DOI: 10.1109/ICCVW.2017.293.

[53] R. Marzotto, A. Fusiello y V. Murino. "High resolution video mosaicing with global alignment". En: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. 2004, págs. I-I. DOI: 10.1109/CVPR.2004.1315099.

[54] H.S. Qureshi et al. "Quantitative quality assessment of stitched panoramic images". En: *Image Processing, IET* 6 (dic. de 2012), págs. 1348-1358. DOI: 10.1049/iet-ipr.2011.0641.

[55] Alex Brollo. *Imagen a Muestra Completa de Adenocarcinoma en Próstata*. URL: https://upload.wikimedia.org/wikipedia/commons/5/57/Prostate_adenocarcinoma_whole_slide.jpg.

[56] Joana Lopes et al. "Melanoma Management: From Epidemiology to Treatment and Latest Advances". En: *Cancers* 14 (2022), pág. 4652. DOI: 10.3390/cancers14194652.

[57] Armaghan Vahidnia et al. "Quantitative polarimetry Mueller matrix decomposition approach for diagnosing melanoma and non-melanoma human skin cancer". En: *OSA Continuum* 4.11 (2021), págs. 2862-2874. DOI: 10.1364/OSAC.425373.

[58] Robert M. Haralick, Stanley R. Sternberg y Xinhua Zhuang. "Image Analysis Using Mathematical Morphology". En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9.4 (1987), págs. 532-550. DOI: 10.1109/TPAMI.1987.4767941.

[59] *Frontmatter*. John Wiley y Sons, Ltd, 2013. ISBN: 9781118600788. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118600788.fmatter.

[60] Rafael Grompone von Gioi et al. "LSD: a Line Segment Detector". En: *Image Processing On Line* 2 (2012). https://doi.org/10.5201/ipol.2012.gjmr-lsd, págs. 35-55.

Esta página ha sido intencionalmente dejada en blanco.

Glosario

AoP Angle of Polarization. 1, 7, 35, 49

ARP Address Resolution Protocol. 30

BGD Batch Gradient Descent. 98, 99

CL Condenser Lens. 22

CNN Convolutional Neural Network. 3, 86, 100–102, 104, 106, 154

CO Colector Lens. 22

CPU Central Processing Unit. 27, 29

DoFP Division of Focal Plane. 1, 13, 18, 19, 34

DoG Difference of Gaussian. 59

DoLP Degree of Linear Polarization. 14, 49

DoP Degree of Polarization. 1, 8, 14

DoTP Division of Time Polarimetry. 13

FAST Features from Accelerated Segment Test. 56

FLANN Fast Library for Approximate Nearest Neighbors. 56

FoV Field of View. 51, 73, 81, 109

FPA Focal Plane Array. 18, 19

GPIO General Pins Input Output. 27, 46

GPU Graphics Processing Unit. 27, 29

GUI Graphic User Interface. 2, 48, 152

IFFI Instituto de Física de Facultad de Ingeniería. 23

IMR Intensity Magnitude Ratio. 74

IP Internet Protocol. 30

LED Light Emission Diode. 7, 22, 34, 35, 40

LoG Laplacian of Gaussian. 57

LSD Line Segment Detector. 88, 90, 154

MAC Medium Access Control. 30

MAE Mean Absolute Error. 100

MLP Multi-Layer Perceptron. 3, 86, 96–98, 100–103, 105, 106, 154

MO Microscopic Objective. 23, 75, 81

MSE Mean Square Error. 75, 77, 100

MSSIM Mean Structural Similarity. 74

MWSI Mueller Whole-Slide Imaging. 81

ORB Oriented FAST and Rotated BRIEF. 53, 56

OS Operative System. 27

PC Personal Computer. 21, 34

PCA Principal Component Analysis. 103, 154

PLA Polylactic acid. 22

PNG Portable Network Graphics. 38

PSNR Peak signal-to-noise ratio. 74–79

RAM Random Access Memory. 29

RANSAC Random Sample Consensus. 56, 62

RBF Radial Base Function. 94

ReLU Rectified Linear Unit. 97, 101, 103, 105

RF Random Forest. 3, 86, 95, 102, 106, 154

RMSE Root Mean Square Error. 74, 76, 78, 79

RMSProp Root Mean Square Prop. 99

SAM Spectral Angle Mapper. 74

SBC Single Board Computer. 27

SD Secure Digital. 27

SDK Software Development Kit. 29

SGD Stochastic Gradient Descent. 99, 105

SIFT Scale-Invariant Feature Transform. 53, 56–58, 60, 152

SoP State of Polarization. 1, 6, 9, 11, 13, 15, 23

SSD Sum of Squared Differences. 57, 65, 66

SSH Secure Shell. 27, 30

SSIM Structural similarity index measure. 74–76, 78

SURF Features from Accelerated Segment Test. 53, 56

SVC Support Vector Classifier. 94, 102, 106

SVM Support Vector Machine. 3, 86, 94, 102

Tanh Tangente Hiperbólica. 97

TL Tube Lens. 23

TPI Threads Per Inch. 23

USB Universal Serial Bus. 46

WSI Whole-Slide Imaging. 1, 74

Esta página ha sido intencionalmente dejada en blanco.

Índice de Tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de Figuras

Esta es la última página.
Compilado el martes 21 mayo, 2024.
http://iie.fing.edu.uy/