UNIVERSIDAD NACIONAL DE COLOMBIA

Development of a Raman imaging system for the detection of phytopathogens

Juan Esteban Velez Alvarez

Universidad Nacional de Colombia
Facultad de Ciencias, Escuela de Física
Medellin, Colombia
2019

Desarrollo de un Sistema de Imágenes Raman Para Detección de Fitopatógenos [1]

Juan Esteban Velez Alvarez

Propuesta presentada para optar al titulo de:
Doctor en Ciencias Física

Director:
(Ph.D.) Alvaro Efrain Bastidas Gustin

Línea de Investigación:
Espectroscopia aplicada a la Biología
Grupo de Investigación:
Grupo de Láseres y Espectroscopia Óptica (GLEO)

Universidad Nacional de Colombia
Facultad de Ciencias, Escuela de Física
Medellin, Colombia
2019

[1] En este trabajo la prueba de concepto se realizó con fitopatógenos de origen fúngico

"I am the master of my failure... If I never fail how will I ever learn.".

C.V Raman

Agradecimientos

Gracias muy especiales a los miembros del grupo GLEO y al laboratorio de Microbiología del suelo que han contribuido al enriquecimiento de esta propuesta de tesis con sus consejos y conocimientos. Es de destacar que sin la ayuda de los profesores Hugo Restrepo, Amanda Lucia Mora, Maria del Socorro Yepes, Rafael Arango, este trabajo no hubiese sido posible, sin sus contribuciones tanto en lo personal como en el apoyo en el acceso a sus laboratorios y equipos. Así mismo es de destacar la ayuda que recibí de diversos miembros de la comunidad academica dentro de los cuales destaco a Alejandra Monsalve laboratorista del laboratorio de micotoxinas, Isabel Calle, joven investigadora del laboratorio de biologia molecular.
A la Dra Heather Allen, por su apoyo y asesoría en mi trabajo, lo cual permitió darle estándares mas elevados, además por su hospitalidad y abrirme las puertas de su grupo de investigación.
Al grupo Allen, donde todos sus miembros sin excepción hicieron mi estadía en los Estados Unidos más que memorable, especialmente Adel Tehseen, Mickey Rogers, Cat Shoeppner y Stephan Baumler.
Es necesario dar una mención especial a algunos funcionarios de la Universidad Nacional que me apoyaron desde lo logístico hasta en la adquisición de los diversos elementos necesarios para esta tesis, en especial Alonso López, Wilson Guerrero

A mi tutor Dr.Alvaro Bastidas, quien logro transmitirme mas allá de sus conocimientos parte de su pasión por hacer cosas, cosas trascendentes, sin miedo, ni limitantes. Destaco también su apoyo en lo personal, entendiendo siempre que mas allá del estudiante existe la persona.
Al profesor Juan Carlos Perez, quien me mostró el camino y la pasión por ïr hasta donde nunca nadie ha llegado", ese lema encierra toda una filosofía de vida la cual es bastante contagiosa. Destaco también el apoyo que brindo desde su laboratorio, haciendo posible el desarrollo de mi investigación.
A toda mi familia, que con sudor y esfuerzo allanaron el camino hasta este punto, sin su ayuda, ni su entusiasmo nada de esto hubiese tenido valor y sentido alguno, puedo afirmar que tome este camino en gran parte por la felicidad genuina y autentica que les generaba, cada vez que comenzaba un nuevo proceso. Mi padre por inculcarme el amor por la ciencia y aprender, mi madre su disciplina y mi hermano por sus ideas tan creativas que de alguna forma empujaban a las mías. Destaco a mis abuelas, quienes fallecieron durante este proceso formativo, las cuales me motivaron y educaron para que fuese posible llegar aquí, a ellas ni el mundo bastaría para agradecerles.
Elizabeth, gracias por tu apoyo incondicional, trajiste luz a las sombras e inundaste de nuevo mi vida con esperanza.

A la facultad de Ciencias por su apoyo económico para la adquisición de algunos elementos necesarios para la finalización de esta tesis de grado.
Finalmente agradezco a mis profesores durante el proceso formativo, haciendo una mención especial a Luis Alberto Sanchez, Jairo Marin y Alvaro Bastidas, quienes me mostraron que el camino en la física es la pasión, sin esta, se vuelve un camino tortuoso y estéril.

Resumen

Los fitopatógenos son microorganismos capaces de infectar plantas según sus características y entornos. Normalmente los fitopatógenos existen en forma de hongos, bacterias, protozoos, etc. que invaden diferentes partes de una planta como el fruto, hojas, tallos, entre otros, lo cual obligaría a un agricultor a una constante inspección por sintomatología para tomar medidas correctivas y oportunas en un cultivo. En este trabajo se abordó el tema de detección no convencional de dos especies de fitopatógenos fúngicos: *Mycosphaerella fijiensis* y *Colletotrichum gloeosporioides*, cuyo foco de infección son cultivos de alto valor comercial para la agro industria colombiana, principalmente banano y aguacate. Para tal fin, y teniendo en cuenta que un hongo altera la configuración energética de la estructura molecular de una planta, entonces se optó por desarrollar la técnica Raman, e incluso se diseñó y se configuró un equipo de muestreo para tal fin. El equipo construido incluye algunos dispositivos ópticos convencionales como lentes, prismas, espejos, y filtros pero también se complementó con otros sistemas de última tecnología como una cámara CCD PTGrey, y un espectrómetro B&W-Tech, obteniendo en definitiva un sistema de muestreo versátil y capaz de trabajar en cualquier ambiente. La detección y el registro de las señales Raman que aquí se obtuvieron corresponden a aquéllas provenientes de moléculas características de la pared celular de estos organismos, tales como quitina y β 1,3-glucano, y que se caracterizan por una eficiencia que perfectamente garantiza su utilidad en procesos de control de calidad tanto en las etapas de cosecha como en poscosecha, indicando al usuario la presencia de estos organismos en una fase temprana, a través de imágenes que hacen posible interpretar la información espectral sin necesidad de personal experto, logrando de esta forma brindar un margen de maniobra suficiente para adoptar medidas correctivas al agricultor, haciendo posible evitar la fumigación indiscriminada de los cultivos. Para lograr esto, el equipo fue sometido a un proceso de calibración con diferentes sustancias y en fases diferentes con el propósito de garantizar las condiciones óptimas de operatividad del equipo. Todo esto fue acompañado por otros desarrollos paralelos como configuraciones ópticas auxiliares, interfases electrónicas y de control, diseños mecánicos, sistemas termodinámicos de enfriamiento en el lugar de la muestra, y desarrollo de software. El resultado es un dispositivo capaz de tomar imágenes espectrales, que posteriormente se puede utilizar para entrenar una red neuronal, que en relación a la detección de fitipatógenos en plantas permite garantizar una estimación de más del 90 % de éxito en este proceso.

Palabras clave: Espectroscopia Raman, Imágenes, Fitopatógeno, Fúngico.

Abstract

Phytopathogens are microorganisms capable of infecting plants according to their characteristics and environments. Normally phytopathogens exist in the form of fungi, bacteria, protozoa, etc. that invade different parts of a plant such as fruit, leaves, stems, among others, which would force a farmer to a constant inspection for symptoms to take corrective and timely measures in a crop. In this work the topic of unconventional detection of two species of fungal phytopathogens was addressed: *Mycosphaerella fijiensis* and *Colletotrichum gloeosporioides*, whose focus of infection are crops of high commercial value for the Colombian agro industry, mainly bananas and avocado. For this purpose, and taking into account that a fungus alters the energy configuration of the molecular structure of a plant, then it was decided to develop the Raman technique; a sampling equipment was designed and configured for this purpose. The built equipment includes some conventional optical devices such as lenses, prisms, mirrors, and filters but it was also complemented with other state-of-the-art systems such as a PTGrey CCD camera, and a B & W-Tech spectrometer, ultimately obtaining a versatile sampling system and able to work in any environment. The detection and recording of the Raman signals obtained here correspond to those coming from molecules characteristic of the cell wall of these organisms, such as chitin and β-1,3-glucan, and which are characterized by an efficiency that perfectly guarantees its usefulness in quality control processes in both the harvest and post-harvest stages, indicating to the user the presence of these organisms at an early stage through images that make it possible to interpret the spectral information without the need for expert personnel, thus achieving a sufficient margin of maneuver to adopt corrective measures, making it possible to avoid the indiscriminate spraying of crops. To achieve this, the equipment was subjected to a calibration process with different substances and in different phases with the purpose of guaranteeing the optimum conditions of operation of the equipment. All this was accompanied by other parallel developments such as auxiliary optical configurations, electronics, control interfaces, mechanical designs, thermodynamic cooling systems at the sample site, and software development. The result is a device capable of taking spectral images, which can later be used to train a neural network, which in relation to the detection of phytopathogens in plants allows to guarantee a 90 % of success in this process.

Keywords: Raman spectroscopy, Imaging, Phytopathogen, Fungic, inelastic scattering

Tabla de Contenido

1 Introducción

La motivación para el diseño aquí propuesto fue proporcionar una solución tecnológica para uno de los sectores económicos más importantes de Colombia; La agricultura. Para ello, se identificaron dos cepas de hongos diferentes como problemáticas para algunos tipos de cultivos, en primer lugar *Mycosphaerella fijiensis* Morelet, causa de la enfermedad llamada tsigatoka negra, la cual produce estrías en las hojas, rendimiento reducido y una maduración prematura. El único control posible de la enfermedad es la aplicación frecuente de productos fungicidas, lo que produce una carga económica y ambiental, y con una eficiencia reducida si el tratamiento se aplica demasiado tarde (10-14) días después de la infección [1], por esa razón se necesita un sistema que pueda dar una alarma oportuna para reducir un poco los problemas mencionados anteriormente.

Una segunda cepa fitopatógena fue probada con nuestra configuración. *Colletotrichum gloeosporioides* es uno de los agentes patógenos más interesantes debido a su alta plasticidad genética y la capacidad de infectar diferentes cultivos que incluyen cereales, verduras, frutas y plantas ornamentales [2], por ejemplo, la pérdida en los cultivos de mora en Colombia puede estar alrededor de (53-73) % debido a la antracnosis [3]. La detección de la enfermedad se puede hacer por PCR (reacción en cadena de la polimerasa) en etapas tempranas, pero es una técnica costosa que necesita reactivos muy específicos (cebadores) [4], esto justifica la búsqueda de un método económico como la dispersión inelástica Raman. Para la detección de este tipo de organismos.
Este trabajo proporciona una herramienta tecnológica para los granjeros inspirados en el trabajo de CV Raman, Placzek, Smekal, Landsberg, Mandelstam y muchos otros, que fueron las primeras personas en imaginar y probar el análogo óptico del efecto Compton, el esparcimiento Raman [5]. Este fenómeno se caracteriza por un cambio en la longitud de onda incidente, lo que implica una variación en el "*momentum*"del fotón, causado por el intercambio de energía con las moléculas, esta interacción podría representar un aumento (desplazamiento azul) o una pérdida (desplazamiento rojo) de la energía del fotón incidente; los nombres técnicos son respectivamente esparcimiento Raman anti-Stokes y Stokes; a pesar de que ambos fenómenos ocurren al mismo tiempo, la probabilidad del desplazamiento Stokes es mayor porque las moléculas tienen más probabilidades de estar en un estado fundamental de energía que en un estado excitado, por eso la intensidad de la señal en este último es más fuerte que del lado anti-Stokes del espectro [6], sin embargo, es necesario decir que este es el caso de la mayoría de las moléculas porque se encuentran principalmente a temperatura

y presión normales (NTP) por lo tanto, en equilibrio termodinámico [7], al ser este último el caso de las aplicaciones presentadas en este trabajo, entonces los filtros seleccionados se seleccionaron para capturar la señal Stokes del espectro.

Se presentarán dos tipos de dispositivos, uno orientado al diagnóstico rápido y la detección de moléculas de interés y el otro orientado a la investigación, específicamente para estudiar la presencia de microorganismos en diferentes tipos de cultivos, que ha sido la razón principal detrás de este proyecto, como se indicó anteriormente. Es importante desarrollar una herramienta potente pero, al mismo tiempo, fácil de usar para el usuario final. El dispositivo portátil es una configuración clásica de dispersión Raman, equipada con un diodo láser a 405 nm y un conjunto de filtros, para acondicionar la iluminación, limpiar la radiación de Rayleigh y seleccionar el rango de números de onda de la región espectral de interés, para el dispositivo portátil se seleccionó la región entre (1300-1700) cm^{-1} debido a que muchos compuestos orgánicos están activos en esta zona, otra característica importante de este dispositivo es la posibilidad de usar el teléfono celular de la cámara como detector, para este desarrollo, se utilizó un Sony Exmor RS IMX400 CMOS, que es el estándar para el modelo XZ Premium, muy bueno en condiciones de poca luz. La calibración se realizó con diferentes tipos de ácidos grasos, ya que son muy activos en la región espectral mencionada anteriormente, para ellos se seleccionaron acido Láurico, acido palmitico y 1-estaoril-2-hidroxi-sn-glicero-3-fosfato, cada uno de ellos con aplicaciones de investigación muy interesantes en ciencias ambientales y cáncer, finalmente la estructura se imprimió en 3D para proporcionar una geometría fija y estable al sistema que permita la reproducibilidad de los experimentos. El dispositivo de banco es un microscopio estereoscópico modificado equipado con un diodo láser de 532 nm compuesto de un conjunto de filtros cuyas funciones son, acondicionar la línea láser, eliminar el esparcimiento Rayleigh y seleccionar la zona de trabajo. Para la aplicación propuesta se eligieron filtros paso de banda para obtener la señal de quitina y $\beta 1,3$ -glucano, dos moléculas muy comunes presentes en la pared celular de hongos [8], uno de ellos cubre la región espectral comprendida entre (788.78-939.85) cm^{-1} donde se encuentra la región de vibraciones de doblamiento ecuatorial y cubre los monómeros de carbohidratos de tipo α y β, en particular la banda de 893 cm^{-1} se considera un biomarcador de β-glucano; el segundo filtro fue seleccionado para la detección de quitina la cual tiene dos picos Raman en la región (1600-1700) cm^{-1} que corresponden al denominado grupo amida I, estos modos de vibración se deben al estiramiento C=O del enlace peptídico **1-1** [9].
De lo anterior es posible obtener una señal exclusiva del fitopatógeno compuesta por las intensidades de los picos en cada una de las regiones espectrales mencionadas, lo cual permite componer una imagen como una combinación lineal de la intensidad de los picos Raman en cada una de las frecuencias 3-2,.

$$\psi_{imagen}(\ omega) = \sum_i \phi_{pico}(\omega) \tag{1-1}$$

Esta señal exclusiva, hace las veces de una "huella dactilar" molecular única, lo que hace

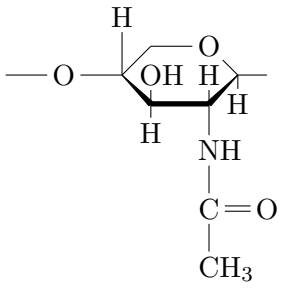


Figura 1-1: Estructura de Lewis de una Molécula de Quitina

posible ejecutar algoritmos de reconocimiento de imágenes, generalmente esta tarea se basa tradicionalmente en PCA (Análisis de componentes principales) [10], [11], pero en este trabajo la técnica seleccionada fue un algoritmo de aprendizaje profundo, llamado redes neuronales convolucionales, que se utiliza principalmente en el procesamiento de imágenes. La plataforma seleccionada para esa tarea fue Tensorflow implementado en Python, esto nos da acceso a uno de los sistemas de clasificación de imágenes más poderosos al entrenar solo la última capa de la red neuronal, esto es posible porque la mayoría de las capas internas de la red fueron preparadas previamente por google. El entrenamiento de la última capa se realizó utilizando PNASNET-5 (Búsqueda progresiva en la arquitectura neuronal), uno de los sistemas más recientes y rápidos para el reconocimiento de patrones, basado en una optimización secuencial al aumentar el nivel de complejidad cada vez [12] [13].

Para los datos de imagen y espectroscópicos, se creó un conjunto completo de scripts, algunos de ellos se utilizaron para mejorar el tiempo de exposición de la cámara, otros para dividir las imágenes en sus respectivos canales RGB.

Finalmente, se creó un sistema de control completo para establecer y mantener la temperatura del porta muestras utilizando una celda Peltier por medio de un controlador PID, seleccionar la potencia del láser a través de la modulación PWM y encender la cámara.
Todo lo anterior proporcionó una plataforma exitosa para la identificación de *Mycosphaerella fijiensis* y *Colletotrichum gloeosporioides* con porcentajes de exito superiores al 90 %, como se mostrará en este trabajo.

2 Objetivo General y Específicos

2.1. Objetivo General

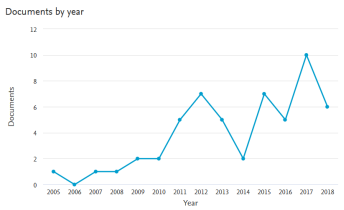
Desarrollar un sistema de registro de espectroscopia Raman útil para la detección de fitopatógenos de interés en la industria agrícola nacional.

2.1.1. Objetivos Específicos

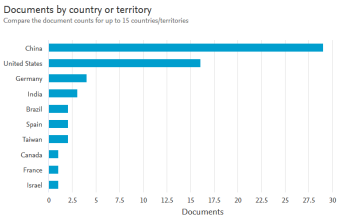
- Implementar un equipo de espectroscopia Raman para el muestreo de material biológico irradiado con luz láser.
- Equipar un porta-muestras con sensores y dispositivos de control de ajuste de parámetros biológicos necesarios para la supervivencia de las muestras.
- Realizar el registro programado de espectros Raman y mediante rutinas y software procesar la información para clasificar y analizar tales registros.

3 Hongos Fitopatógenos y Fundamentos de la Técnica Raman

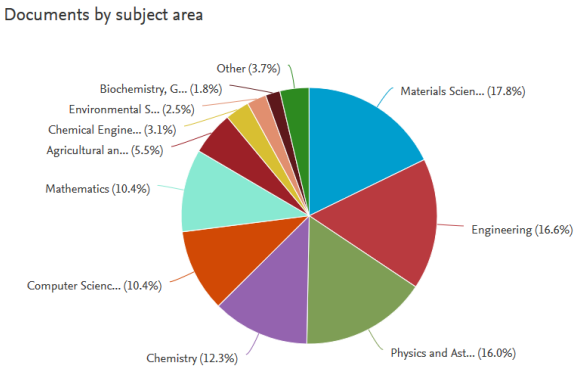
El interés por el esparcimiento Raman aumenta día a día en diferentes campos de investigación, principalmente porque la reducción de costes de las fuentes láser y de los fotodetectores de alta sensibilidad facilita el acceso a este tipo de instrumentos, como prueba de ello, la figura **3-1** muestra, la tendencia creciente año a año de la aplicación de esta técnica al campo de la investigación agrícola actualizada a 2018, (clusters).



(a) Documents by Year



(b) Documents by Region



(c) Documents by Subject Area

Figura 3-1: Analysis of Raman Scattering Relevance in Agricultural Research

Es importante destacar los documentos por región como un indicador de la relevancia que cada país da al desarrollo de este sector estratégico y muestra un poco la situación en la región norte de Suramérica en términos de inversión en tecnología o su exitosa aplicación.

Sólo Brasil tiene una participación significativa en el uso de técnicas ópticas para el desarrollo de plataformas tecnológicas para las tierras de cultivo, los demás países a pesar de la relevancia de la agricultura para sus economías están ausentes en este indicador, lo que significa que existe la necesidad de crear nuevas tecnologías para mejorar la productividad en estas regiones, permitiendo a los agricultores tomar el control de sus cultivos a través del monitoreo de variables clave, como la nutrición, la humedad del suelo, la radiación, y en este caso un sistema de diagnóstico fitosanitario que es crucial para la exportación a países o comunidades con altos estándares de calidad como Estados Unidos y la Unión Europea, ambos principales consumidores de productos agrícolas colombianos [14].

3.1. Estructura de la pared celular de los hongos

En este trabajo el conocimiento sobre la pared celular de los hongos, es fundamental para saber de antemano que moléculas químicas son aptas para identificar de forma precisa la presencia del hongo sobre la planta. Por tal motivo y en aras de tener éxito en esta investigación es necesario profundizar en este aspecto, ya que al ser la parte más expuesta de la célula es la más susceptible para la obtención de información espectroscópica usando radiación láser como fuente de excitación. La obtención de imágenes Raman, se hará respecto de las moléculas más comunes de la pared celular de los fitopatógenos seleccionados para este trabajo.

A continuación se presenta una revisión bibliográfica enfocada en la pared celular de hongos de diferentes tipos y los desplazamientos Raman asociados a las moléculas presentes en la pared.

3.1.1. Composición química de la pared celular de hongos

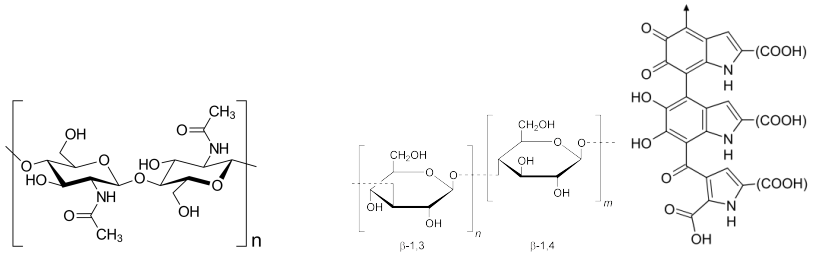
Las paredes son elementos funcionales muy importantes en las células fúngicas, ya que desempeña actividades fundamentales para la viabilidad de la célula tales como, transporte de nutrientes, comunicación celular, metabolismo de sustratos no permeables etc. Se compone principalmente de material fibrilar unido por proteínas y azucares principalmente [15].

Una de las más recientes revisiones bibliográficas sobre el tema de la pared celular de hongos fue publicado en 2013 por Stephen J. Free de la universidad estatal de New York, en el cual compara 5 especies de diferentes de hongos; *Saccharomyces cerevisiae, Candida albicans, Aspergillus fumigatus, Schizosaccharomyces pombe, Neurospora crassa, y Cryptococcus neoformans*, de los cuales entra a analizar las diferencias y similitudes en su pared celular, de allí concluye que los principales componentes comunes a estas especies y quizás aplicable a otras son los siguientes: Quitina, quitosán, β 1,3-glucanos, β 1,6-glucanos, combinaciones

Tabla 3-1: Desplazamientos Raman y longitudes de onda de las principales moléculas encontradas en pared celular [16], [17]

Molécula	**Desplazamiento Raman** cm^{-1}	$\lambda(\eta m)$
Quitina	1638	582.78
β glucanos	1392	574.55
Proteínas	1686	584

de β 1,3/1,4 glucanos, α 1,3 glucano, melanina y glucoproteinas (ver figura **3-2**[1].



(a) Quitina (b) β-1,3 y 1,4 glucano (c) Melanina

Figura 3-2: Principales Moléculas de la Pared Celular de Hongos

Lo anterior es importante conocerlo dado que sobre algunos de estos compuestos se han llevado a cabo experimentos espectroscópicos de obtención de sus espectros Raman, así como también la deducción teórica de las lineas Raman mediante teoría de funcionales de densidad (*"DFT"*); ejemplo de esto, es el trabajo de (Cheol et al, 2013), en el cual mediante cálculos teóricos usando el paquete de software para química teórica Gaussian 09 logran establecer en que rango se encuentran las emisiones de las moléculas de β 1,3- glucano, β 1,6-glucano y quitina, presentes en la pared celular de hongos para posteriormente establecer el grado de acierto mediante la toma de espectros Raman de las moléculas mencionadas.

Teniendo en cuenta lo anterior se puede comenzar a explorar el sistema de imágenes Raman propuesto en este trabajo partiendo de la información recopilada por estos investigadores respecto a las lineas de espectro Raman activas para los compuestos más comunes de la pared; en la tabla **3-1** se amplia lo aquí expuesto .

3.1.2. Algunos hongos fitopatógenos de interés

Se sugieren las siguientes especies de hongos fitopatógenos por su incidencia en cultivos de interés nacional, así como por la facilidad para su aislamiento y manipulación. Cabe aclarar

[1]Las imágenes utilizadas son de licencia Creative Commons

que de todos los organismos nombrados a continuación solo se elegirá uno de ellos como organismo modelo para el trabajo final, así mismo el factor común de todas estas especies, es su incidencia en poscosecha y un corto periodo de incubación.

Colletotrichum Spp.

Se trata de un hongo fitopatógeno que ataca un gran numero de cultivos y plantas ornamentales, causando la enfermedad conocida como antracnosis. Algunas especies producen de forma frecuente su forma teleomorfica *Glomerella cingulata*, a las cuales se les atribuye una serie de enfermedades conocidas como enfermedades de Glomerella. Estas especies causan ademas chancros y muerte regresiva en algunos frutales.

Algunos de los cultivos más afectados anualmente por esta enfermedad son el algodón, fresas, fríjol, cebolla entre otros [18]. Es considerada como el octavo más importante genero de hongo fitopatógeno en el mundo, debido a su influencia en factores tanto económicos como científicos, esto, según una encuesta realizada a 495 personas de la comunidad internacional [19].
Se ha podido comprobar que *Colletotrichum graminicola* durante su fase necrotrofica incrementa de forma significativa la producción de β 1,3 - glucanos, mientras que en su etapa biotrofica su concentración es baja, lo cual indicaría que se trata de un patrón molecular asociado a microbios (MAMP) que permitiría a la planta reconocerlo y activar mecanismos de defensa ante la infección [20]. Sin embargo como el interés es un detección temprana, lo que se buscaría es un punto de inflexión en la aparición de esta sustancia, ese cambio marcaría el comienzo de la enfermedad. En la figura **3-3** se pueden apreciar las lesiones producidas en un fruto de aguacate.
El banano es uno de los cultivos más importantes en la economía colombiana; solo en 2017 el valor de las exportaciones fue de US $ 850 millones [21], pero se necesita una gran cantidad de recursos para controlar la enfermedad, según [22] US $ 550 millones se gastan anualmente para controlar la Sigatoka negra en todo el mundo, por esta razón, se necesita un método para detectar la enfermedad para reducir los costos totales de mantenimiento de cultivos.

Un hecho interesante sobre *Mycosphaerella* está relacionado con el período de infección, que toma 22 días antes de que aparezca una mancha negra, esto es importante porque el dispositivo podría detectar la presencia del fitopatógeno durante las primeras etapas de la infección y de esta manera permitir un cierto control de la progresión de la enfermedad en **3-4** la figura muestra las lesiones causadas por el hongo en una etapa muy tardía, la idea es detectarla en una temprana para aumentar la eficiencia de la lucha contra la enfermedad, ya que la aplicación de un producto antifúngico, según (Lazo, et al., 2012) [23] debe realizarse

Figura 3-3: Antracnosis del aguacate

antes de los 15 días del inicio de la infección. Por lo tanto, un sistema capaz de detectar unos pocos grupos de moléculas ayudará a reducir las pérdidas y a la vez, disminuirá la cantidad de producto necesaria para detener la enfermedad, mejorando de esta manera la calidad y la seguridad del producto cosechado.

3.2. Imágenes Raman

Para esta sección se realizará una revisión bibliográfica de los últimos adelantos en imágenes Raman. Se busca mostrar una variedad de aplicaciones de tal forma que se muestre el valor científico de la técnica en el estudio de sistemas biológicos de forma general, esto con el fin de dar a entender la relevancia que pueden llegar a tener en el caso concreto de la agricultura.

3.2.1. Avances en el campo de imágenes diagnosticas

La idea de usar imágenes Raman para diagnosticar enfermedades no es nueva, de hecho el primer trabajo relacionado que se pudo rastrear data de 1989, este, buscaba comparar imágenes obtenidas mediante Raman resonante de plasma sanguíneo obtenido de personas sanas y compararlo con el de personas enfermas, con el fin de desarrollar un herramienta diagnostica para la detección de algún tipo de mal, pese a que Surrendra Verma no logro finalmente que se le otorgue la patente[24], la idea si perduro hasta el día de hoy. Por ejemplo

Figura 3-4: Marcas Causadas por *Mycosphaerella*

el trabajo de Harmsen y colaboradores [25], busca mediante el desarrollo de un nuevo tipo de nanopartícula de oro con forma de estrella y Raman estimulado por superficie (SERS) detectar lesiones tumorales alejadas del tumor principal, las cuales muchas veces son indetectables mediante los métodos diagnósticos tradicionales.

Uno de los más interesantes avances en el diagnostico de enfermedades se puede leer en (Abramczyk et al, 2016) en donde mediante biomarcadores como carotenoides, esfingomielina, ácido palmítico y otros. Se estudian los ductos en el tejido mamario en pacientes con cáncer de seno con el fin de entender mejor la ubicación y propagación de ciertas moléculas para así comprender la relación bioquímica entre estos y el desarrollo de la enfermedad.
Otro de los puntos fuertes de las imágenes Raman, es como ayuda diagnostica en las neurocirugías asociadas a tumores malignos como el glioma, en [26] se muestra como mediante las imágenes Raman obtenidas mediante Raman estimulado pueden servir para diferenciar en tiempo real, tejido sano de enfermo e inclusive encontrar estructuras que mediante métodos tradicionales no son susceptibles de ser encontradas. Esto lo lograron concluir mediante el análisis de 41 tejidos enfermos provenientes de 12 pacientes. Es evidente el avance de esta técnica desde las primeras ideas en el ya remoto 1989, hasta la actualidad, donde la técnica ya es considerada una ayuda diagnostica relevante y que por motivos técnicos y de costos aun no se ha popularizado, igual queda la sensación de que es cuestión de tiempo para verla como una herramienta diagnostica convencional en un futuro cercano y cuando la legislación

medica así lo permita.

3.2.2. Imágenes Raman aplicadas a la agricultura

En la actualidad las imágenes Raman gozan de cierta acogida en el campo agrícola, principalmente en la identificación de compuestos de gran valor comercial y en un segundo plano en la identificación de fitopatógenos. El trabajo realizado por Roman y colaboradores en 2015 [27], es una muestra de lo antes mencionado, ya que ellos logran identificar diferentes concentraciones de cristales de carotenoides presentes en células de la raíz de una planta de zanahorias . Analizaron la homogeneidad en la distribución de estos cristales empleando láseres en longitudes de onda de 532nm y 488nm, en donde el segundo permitía la identificación de otro compuesto como la luteína. Una de las ventajas que destacan los investigadores, es la posibilidad de realizar estos ensayos sin dañar la planta lo cual representa una mejora respecto de otras técnicas químicas empleadas para este fin, en donde resulta necesario la extracción de algunos tejidos.

El articulo titulado Raman chemical imaging of the rhizosphere bacterium Pantoea sp. YR343 and its co-culture with Arabidopsis thaliana"publicado en 2016, se acerca un poco a lo que se busca realizar en esta tesis doctoral, básicamente logran probar que la técnica de imágenes Raman acopladas a microscopia permiten hacer un estudio de la rizosfera con muy baja intervención, que les permitió analizar los comportamientos de colonias bacterianas y su interacción con otros organismos, usando como modelo la planta *Arabidopsis thaliana* e inoculada con *Pantoea sp.*. Uno de los elementos técnicos de este articulo que lo hacen interesante es el análisis por componentes principales (PCA), el cual les permite diferenciar las señales espectrales provenientes de la planta de las asociadas a las células bacterianas [28].

Otra metodología abordada para el uso de la espectroscopia Raman aplicada al agro, es la que presentan (Vitek et al, 2017) en la cual acompañan la toma de espectros Raman secuenciales (mapeo) con imágenes de fluorescencia por clorofila, lo cual les permite establecer los daños asociados a los herbicidas clonazone y diflufenican. En este estudio se pudo establecer la disminución de carotenoides en *Helianthus annuus* asociados a la presencia de los compuestos anteriormente mencionados, para esto se empleo un diodo láser de 785nm con una potencia 30mW, realizando 5 escaneos de 3 segundos cada uno. En el articulo se presenta una tabla de gran utilidad, en donde se dan a conocer las principales bandas Raman presentes en la hoja de girasol [29], este dato facilitara la toma de imágenes Raman ya que se puede inferir el tipo de señal perteneciente a la hoja, partiendo del hecho de que el modelo biológico de planta seleccionado tenga una composición química similar en la hoja de girasol.

También es posible encontrar trabajos enfocados en la identificación y clasificación de patógenos, por ejemplo el articulo titulado Ïnvestigation of the antimicrobial activity of soy peptides by developing a high throughput drug screening assay", trata precisamente de establecer la resistencia a antibióticos, en este caso se busco tolerancia a péptidos de la solla. Se emplearon dos cepas; *Pseudomonas aeruginosa y Listeria monocytogenes*, el uso de la espectroscopia Raman en este caso fue para identificar cada una de las cepas, para este ensayo emplearon la técnica SERS (Raman intensificado por superficie) y un láser a 745nm; la principal ventaja de haber utilizado espectroscopia Raman, fue que los tiempos para apreciar la interacción del patógeno con los péptidos se redujo a unas cuantas horas, cosa que una incubación tradicional jamas permitiría realizar [30].
La siguiente tabla muestra las diferentes variaciones de la espectroscopia Raman aplicada a la investigación biológica, las diferentes variantes se utilizan para evitar la fluorescencia, aumentar la intensidad natural de la dispersión Raman u obtener información adicional de los grupos moleculares en la muestra, como la posición de los grupos funcionales o la quiralidad.

Técnica	Descripción	Autores
Raman Estimulado (SRS)	En esta técnica se emplean dos láser de frecuencias ω_1 y ω_2, de tal forma que si $\Delta\omega = \omega_1 - \omega_2$ está cerca de una transición molecular vibracional, la señal Raman sera amplificada. Actualmente es una de las técnicas mas empleadas para generación de imágenes moleculares.	[31]
Raman Intensificado por Superficie (SERS)	En esta técnica, la amplificación de la dispersión Raman se produce por la interacción con una superficie metálica, la cual genera tres tipos de resonancia, plasmon superficie, transferencia de carga y resonancia molecular en el rango de excitación	[32]
Raman por Transmisión	Esta técnica se caracteriza por dejar pasar los fotones a través de la muestra, en lugar de capturar la reflexión inelastica, se captura la transmisión, dejando apreciar la composición o el comportamiento del total de la muestra, dando más detalles cuando de sistemas complejos se trata, por ejemplo sistemas biológicos	[33]
Raman Actividad Óptica (ROA)	Permite obtener la quiralidad de las moléculas y ver pequeñas diferencias en la polarización de la luz a izquierda y derecha, de la radiación dispersada. Es útil para la identificación de pequeñas moléculas orgánicas e inclusive se ha logrado trabajar con muestras de virus intactas.	[34]

Tabla 3-2: Algunas técnicas Raman aplicadas en Biología

3.2.3. Principios Fisicoquímicos y Revisión Histórica

La dispersión Raman-Smekal fue teorizada por Adolf Smekal en 1923, y llamó a esta obra *"Zur Quantentheorie der Dispersion"*. [35] en la cual afirma que la luz monocromática tiene un comportamiento, que solo puede ser explicado desde la mecánica cuántica, en el cual, debido a la interacción con la materia, se produce un cambio a longitudes de onda mayores y menores respecto de la original. Esa afirmación fue más una hipótesis que una demostración del fenómeno, más adelante entre 1924-1925 Werner Heisemberg y Hendrik A. Kramers junto con P.A.M Dirac, dieron una explicación satisfactoria del proceso de esparcimiento desde el punto de vista de la mecánica cuántica, proporcionando una relación que explica la probabilidad de emisión de fotones a través de un ángulo sólido después de la excitación del sistema con radiación monocromática de frecuencia ω_k [36][37].

$$\frac{d^2\sigma}{d\Omega_{k'}d(\hbar\omega'_k)} = \frac{\omega'_k}{\omega_k}\sum_{|f\rangle}\left|\sum_{|n\rangle}\frac{\langle f|\,T^\dagger\,|n\rangle\,\langle n|\,T\,|i\rangle}{E_i - E_n + \hbar\omega_k + i\dfrac{\Gamma_n}{2}}\right|^2 \delta(E_i - E_f + \hbar\omega_k - \hbar\omega'_k) \tag{3-1}$$

Esta ecuación 3-1 expresó por primera vez la posibilidad de que un fotón esparcido pudiera tener mayor energía que el incidente e inspiró el trabajo de Grigorii Samuilovich Landsberg y Leonid Isaakovich Mandelstam, que en mayo de 1928 descubrió el cambio en la longitud de onda de luz incidente sobre un cristal, este trabajo se publicó en *"Die Naturwissenschaften"* [38], mientras que Raman y Krishnan reportaron la disminución de la frecuencia con respecto a la radiación incidente para 60 tipos diferentes de líquidos y vapores, pero en marzo del mismo año, es por eso que esta técnica se llama espectroscopia Raman [39]. Como dato curioso, la Unión Soviética nunca aceptó el nombre de espectroscopía Raman para este fenómeno, en lugar de eso lo llamaron esparcimiento combinado porque pensaron que Mandelstam y Ladnsberg merecían más mérito por su trabajo [40].

Uno de los avances más importantes del esparcimiento Raman surge accidentalmente después del descubrimiento de T.H. Maiman, con su dispositivo de radiación estimulada producida por un Rubí finalmente llamado láser [41], al cual se le conoce como Raman estimulado. Todo comenzó cuando los investigadores colocaron accidentalmente una celda de cuarzo con nitrobenceno dentro de una cavidad de un láser de Rubí y observaron una fuerte emisión diferente de la central (694.3nm), la explicación de este incidente se produjo tres años después con el documento *"Theory of la dispersión de Brillouin y Raman "* escrita por Shen et al., (1965) [42], quienes explican de manera satisfactoria los resultados obtenidos por los experimentos de aquella época [43].

Teoría Clásica

Para los propósitos de este trabajo, se necesita una breve revisión de los conceptos de la teoría de Raman clásica y cuántica, esta parte mostrará el origen del desplazamiento Raman cuando un campo eléctrico interactúa con un grupo de moléculas, luego para comprender cuáles son las causas detrás de este cambio en el momento de los fotones y más importante aún la diferencia en la intensidad entre los efectos Stokes y anti-Stokes se dará una explicación cuántica.

Si un campo eléctrico E_0 interactúa con una molécula, producirá un movimiento de electrones a posiciones opuestas generando un dipolo temporal si además el campo eléctrico no es lo suficientemente fuerte como para modificar permanentemente la molécula, en este caso, la relación entre el momento dipolar inducido y el campo eléctrico incidente están dados por:

$$\mu = \alpha \mathbf{E} \tag{3-2}$$

El escalar α es una constante de proporcionalidad y se conoce como *"polarizabilidad"*, la cual es una propiedad característica de la molécula. En la figura **3-5** se muestra el problema general.

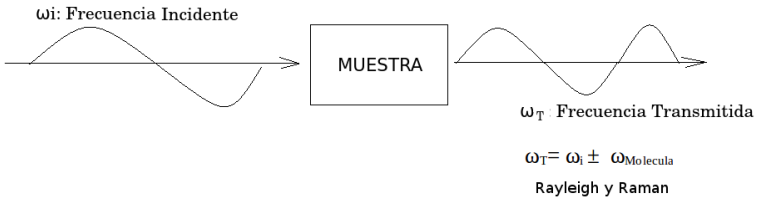


Figura 3-5: Diagrama de Esparcimiento Raman

La intensidad de la luz esparcida es proporcional a la potencia al cuadrado del momento dipolar oscilatorio inducido en la molécula, por lo cual es posible que aparezcan nuevas frecuencias debido a la resonancia entre la radiación y la muestra. [44].

$$\alpha = \alpha_0 + \alpha_1 cos(\omega t) \tag{3-3}$$

Reemplazando 3-3 en 3-2, suponiendo que la radiación es monocromatica (láser), entonces:

$$\mathbf{E} = E_0 cos(\omega_i) \tag{3-4}$$

using the trigonometric expression $cos(\alpha)cos(\beta) = \frac{cos(\alpha+\beta)+cos(\alpha-\beta)}{2}$ we got

$$\mu = \alpha_0 E_0 cos(\omega_i t) + \frac{E_0\alpha_1}{2} cos(\omega + \omega_i) + \frac{E_0\alpha_1}{2} cos(\omega_i - \omega) \tag{3-5}$$

De 3-5 sabemos que el primer término del lado derecho es la dispersión Rayleigh porque no hay cambio en la frecuencia, el segundo término es la dispersión Raman anti-Stokes en la cual la frecuencia se desplaza hacia valores más altos, esto significa que la molécula ya estaba en un estado excitado, entregando parte de su energía de vibración al fotón, finalmente el desplazamiento Stokes, tercer término de la ecuación, la frecuencia se desplaza a valores más bajos, por lo que la molécula obtiene algo de energía del campo electromagnético para poder oscilar. El diagrama **3-6** resume los tres procesos.

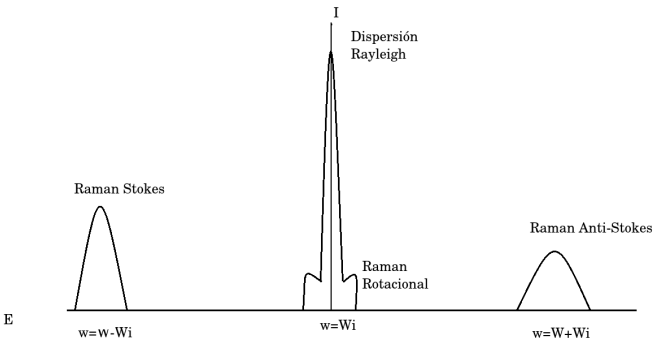


Figura 3-6: Comportamiento de las intensidades y frecuencias durante el esparcimiento Raman

La figura **3-6** muestra una diferencia de las intensidades de los tres procesos, pero en el enfoque clásico no es posible notar esto en la ecuación 3-5, a pesar de esto el modelo si fue exitoso en explicar el desplazamiento en frecuencia.
Un resultado interesante proviene de la distribución de Boltzmann aplicada a la población de partículas de los estados excitados y fundamentales que proporciona una relación entre las intensidades de Stokes y anti-Stokes, de la siguiente forma.

$$\frac{I_{anti-Stokes}}{I_{Stokes}} = \frac{(\omega_i + \omega_{vibrational})^4 e^{-\hbar\omega_{vibrational}\beta}}{(\omega_i - \omega_{vibrational})^4} \tag{3-6}$$

[2] La cantidad de particulas en estados excitados es usualmente inferior que aquellas en estados base, por ese motivo, la radiación anti-Stokes es una fracción de la producida en el desplazamiento Stokes, ver **3-7**. Para explicar la diferencia de las intensidades es necesario

[2] Adaptado de [44]. Con $\beta = \frac{1}{k_B T}$

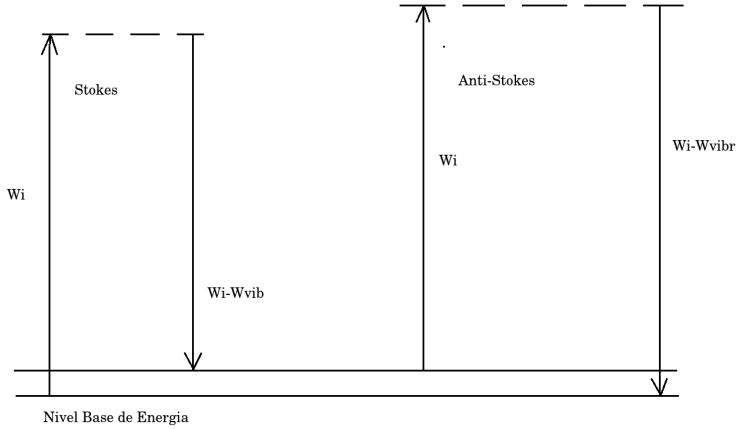


Figura 3-7: Diagrama de Energía de los desplazamientos Stokes y anti-Stokes

un análisis más detallado de las interacciones moleculares, y la única manera de abordar este tipo de problema es con la ayuda de la mecánica cuántica, específicamente con la ecuación KHD (Kramers-Heisemberg-Dirac) [45].

Modelo Cuántico

La figura **3-8**, muestra el cambio en los niveles de energía cuando un campo eléctrico induce una vibración en un grupo de moléculas.

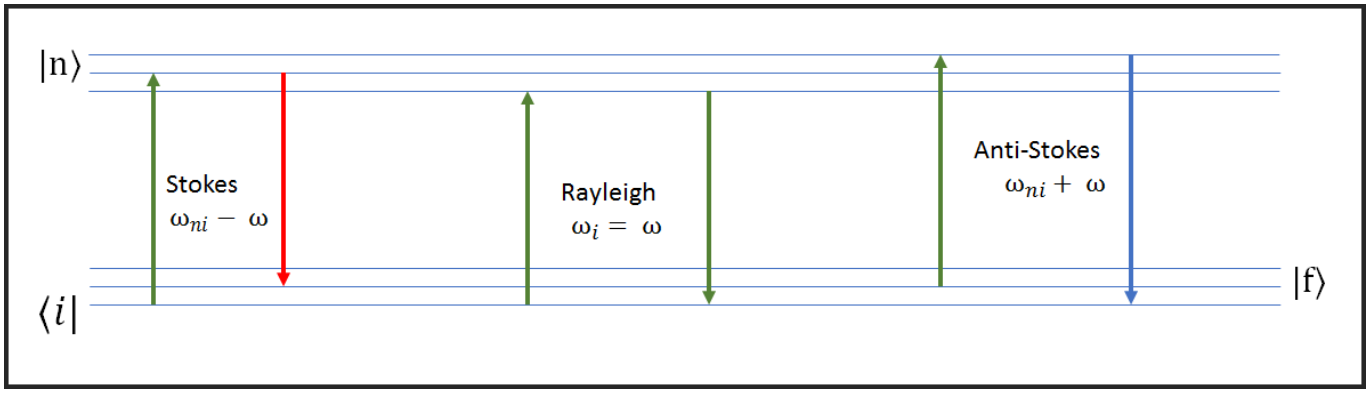


Figura 3-8: Niveles de Energía en el Desplazamiento Raman

Comenzamos con dos estados caracterizados por un fotón incidente y dispersado, como se mencionó, la representación clásica de la dispersión Raman no explica la diferencia en las intensidades entre los modos de vibración de Stokes y anti-Stokes, por lo tanto, si la radiación láser interactúa con una muestra, el resultado de ese evento producirá un dipolo inducido momentáneo, 3-7.

$$\langle \psi_i | \, \mu \, | \psi_f \rangle = e^{i\omega_{if}t}(\mu_{permanent} + \mu_{induced}) \tag{3-7}$$

Este momento dipolar inducido es diferente para cada molécula y depende de la polarizabilidad, de la forma 3-8 [46].

$$\mu_{induced} = \frac{1}{2}(\alpha_{if}e^{i\omega t} + \alpha^*_{fi}e^{-i\omega t}) \tag{3-8}$$

con la condición de que el tensor de polarizabilidad debe ser hermítico,

$$\mu_{if} = \mu^*_{fi} \tag{3-9}$$

El estado final esta definido por el siguiente ket:

$$|\psi_f(t)\rangle = \sum_n C^f_n(t)e^{-iE_nt/\hbar} \, |n\rangle \tag{3-10}$$

con f un estado final sin perturbar en t=0, [47], los coeficientes C^j_n están definidos por,

$$C^j_n = \delta_{nj} - \frac{i}{\hbar}\int_0^t d\tau e^{i\omega_{nj}\tau}V_{nj}(\tau) \tag{3-11}$$

$V_{nj}(t)$ es el operador de perturbación definido por:

$$V_{nj}(t) = -\frac{1}{2}\langle n|\,\mu.E_0\,|j\rangle\,(e^{i\omega t} + e^{-i\omega t}) = -V_{nj}^0 = (e^{i\omega t} + e^{-i\omega t}) \tag{3-12}$$

with

$$V_{nj}^0 = \frac{1}{2}(\mu_{nj}.E_0) \tag{3-13}$$

En 3-13 fue aplicada la definición habitual del delta de Kronecker. Reemplazando 3-12 en 3-11,

$$C_n^f(t) = \delta_{nf} + \frac{iV_{nf}^0}{\hbar}\int_0^t d\tau e^{i\omega_{nf}\tau}(e^{i\omega\tau} + e^{-i\omega\tau}) = \delta_{nf} + \frac{iV_{nf}^0}{\hbar}(\int_0^t e^{i\tau(\omega_{nf}+\omega}d\tau + \int_0^t e^{i\tau(\omega_{nf}-\omega)}d\tau) \tag{3-14}$$

Integrando y teniendo en cuenta que la integral de la parte imaginaria de la identidad de Euler es cero debido a la paridad de la función seno, entonces:

$$C_n^f(t) = \delta_{nf} + \frac{iV_{nf}^0}{\hbar}(\frac{e^{it(\omega_{nf}+\omega)} - 1}{\omega + \omega_{nf}} - \frac{e^{it(\omega-\omega_{nf})} - 1}{\omega - \omega_{nf}}) \tag{3-15}$$

De la ecuación 3-15, si la condición de resonancia es forzada, entonces el segundo término entre paréntesis dará una singularidad, para evitar eso, en el término de energía se puede dar una condición de amortiguamiento, por lo tanto:

$$E_n \to E_n - \frac{i\hbar}{2}\Gamma_n = \hbar(\omega_{nf} - \frac{i\Gamma_n}{2}) \tag{3-16}$$

donde $Gamma_n$ es la tasa de decaimiento del estado de transición, esto prohíbe que el sistema permanezca un tiempo infinito en cualquier estado, eliminando así la singularidad durante la resonancia

$$\Gamma_n = \frac{1}{\hbar^2}\sum_{m<n} |\,\langle n|\,V^0\,|m\rangle\,|^2\rho(\nu_{nm}) \tag{3-17}$$

3-17 es la regla de oro de Fermi, por lo cual 3-15 puede ser reescrita como:

$$C_n^f(t) = \delta_{nf} + \frac{iV_{nf}^0}{\hbar}(\frac{e^{it(\omega_{nf}+\omega)} - 1}{\omega + \omega_{nf} - \frac{i}{2}\Gamma_n} - \frac{e^{it(\omega-\omega_{nf})} - 1}{\omega - \omega_{nf} - \frac{i}{2}\Gamma_n}) \tag{3-18}$$

Reemplazando 3-18 en 3-10:

$$|\psi_f(t)\rangle = \sum_n e^{-iE_f t/\hbar}\,|f\rangle + \frac{iV_{nf}^0}{\hbar}(\frac{e^{i(\omega-\omega_{nf})t} - 1}{\omega + \omega_{nf} - \frac{i\Gamma_n}{2}} - \frac{e^{-i(\omega_{nf}-\omega)t} - 1}{\omega_{nf} - \omega - \frac{i\Gamma_n}{2}})e^{-iE_n t/\hbar}\,|n\rangle \tag{3-19}$$

Factorizando exponenciales y sabiendo que: $e^{-it/\hbar(E_n-E_f)} = e^{-it\omega_{nf}}$:

$$|\psi_f(t)\rangle = e^{-iE_f t/\hbar}(|f\rangle + \sum_{n\neq f} \frac{iV^0_{nf}}{\hbar}(\frac{e^{i(\omega-\omega_{nf})t}-1}{\omega+\omega_{nf}-\frac{i\Gamma_n}{2}} - \frac{e^{-i(\omega_{nf}-\omega)t}-1}{\omega_{nf}-\omega-\frac{i\Gamma_n}{2}})e^{-i\omega_{nf}t}\,|n\rangle) \tag{3-20}$$

Entonces:

$$|\psi_f(t)\rangle = e^{-iE_f t/\hbar}(|f\rangle + \sum_{n\neq f} \frac{iV^0_{nf}}{\hbar}(\frac{e^{i(\omega)t}-e^{-i(\omega_{nf})t}}{\omega+\omega_{nf}-\frac{i\Gamma_n}{2}} - \frac{e^{-i(\omega)t}-e^{-i(\omega_{nf})t}}{\omega_{nf}-\omega-\frac{i\Gamma_n}{2}})\,|n\rangle) \tag{3-21}$$

Cerca de una transición, es factible aplicar la aproximación de onda giratoria [48], la cual ignora los términos con altas frecuencias de oscilación, en este caso todos los términos de la transición $n \to f$ [46]:

$$|\psi_f(t)\rangle = e^{-iE_f t/\hbar}(|f\rangle + \frac{iV^0_{nf}}{\hbar}\sum_{n\neq f} \frac{e^{i\omega t}}{\omega+\omega_{nf}-\frac{i\Gamma_n}{2}} - \frac{e^{-i\omega t}}{\omega_{nf}-\omega-\frac{i\Gamma_n}{2}})\,|n\rangle) \tag{3-22}$$

Tomando el complejo conjugado de 3-22 entonces:

$$\langle\psi_i(t)| = e^{iE_i t/\hbar}(\langle i| + \frac{iV^0_{nf}}{\hbar}\sum_{n\neq i} \frac{e^{-i\omega t}}{\omega+\omega_{ni}-\frac{i\Gamma_n}{2}} - \frac{e^{i\omega t}}{\omega_{ni}-\omega-\frac{i\Gamma_n}{2}})\,\langle n|) \tag{3-23}$$

Reemplazando 3-23, 3-22 en el lado izquierdo de la ecuación 3-7:

$$\begin{aligned} &= e^{iE_i t/\hbar}(\langle i| + \frac{iV^0_{nf}}{\hbar}\sum_{n\neq i} \frac{e^{-i\omega t}}{\omega+\omega_{ni}-\frac{i\Gamma_n}{2}} - \frac{e^{i\omega t}}{\omega_{ni}-\omega-\frac{i\Gamma_n}{2}})\,\langle n|)|\mu| \\ &\quad e^{-iE_f t/\hbar}(|f\rangle + \frac{iV^0_{nf}}{\hbar}\sum_{n\neq f} \frac{e^{i\omega t}}{\omega+\omega_{nf}-\frac{i\Gamma_n}{2}} - \frac{e^{-i\omega t}}{\omega_{nf}-\omega-\frac{i\Gamma_n}{2}})\,|n\rangle) \end{aligned} \tag{3-24}$$

Simplificando un poco esta ultima expresión

$$\begin{aligned} = e^{i\omega_{if}t}(\langle i|\,\mu\,|f\rangle &+ \frac{1}{\hbar}\sum_n V^0_{in}\mu_{nf}(\frac{e^{-i\omega t}}{\omega+\omega_{ni}-\frac{i\Gamma_n}{2}} - \frac{e^{i\omega t}}{\omega_{ni}-\omega-\frac{i\Gamma_n}{2}}) \\ &+ \frac{1}{\hbar}\sum_n \mu_{nf}V^0_{in}(\frac{e^{-i\omega t}}{\omega+\omega_{ni}-\frac{i\Gamma_n}{2}} - \frac{e^{i\omega t}}{\omega_{ni}-\omega-\frac{i\Gamma_n}{2}}) \end{aligned} \tag{3-25}$$

El termino $\langle i|\,\mu\,|f\rangle$ es el dipolo permanente, y reemplazando $V^0_{nj} = \frac{1}{2}(\mu_{nj}.E_0)$, se tiene que:

$$\mu_{if} = \frac{1}{2\hbar}\sum_n \mu_{in}\mu_{nf}(\frac{e^{i\omega t}}{\omega+\omega_{nf}-\frac{i\Gamma_n}{2}} - \frac{e^{-i\omega t}}{\omega_{ni}-\omega-\frac{i\Gamma_n}{2}}).E_0 \tag{3-26}$$

Si 3-26 es comparado con $\mu_{induced} = \frac{1}{2}(\alpha_{if}e^{i\omega t} + \alpha^*_{fi}e^{-i\omega t}$, entonces:

$$\alpha_{if} = \frac{1}{\hbar}\sum_n (\frac{\mu_{in}\mu_{nf}}{\omega + \omega_{nf} - \frac{i\Gamma_n}{2}} - \frac{\mu_{nf}\mu_{in}}{\omega_{ni} - \omega - \frac{i\Gamma_n}{2}}) \tag{3-27}$$

$$(\alpha_{\rho\sigma})_{if} = \frac{1}{\hbar}\sum_n (\frac{\langle i|\,\mu_\rho\,|n\rangle\,\langle n|\,\mu_\sigma\,|f\rangle}{\omega + \omega_{nf} - \frac{i\Gamma_n}{2}} - \frac{\langle i|\,\mu_\sigma\,|n\rangle\,\langle n|\,\mu_\rho\,|f\rangle}{\omega_{ni} - \omega - \frac{i\Gamma_n}{2}}) \tag{3-28}$$

Esta última expresión es la fórmula de Kramers-Heisemberg-Dirac y explica la diferencia entre las intensidades en los Stokes ($\omega - \omega_{nf}$ y las señales anti-Stokes ($\omega + \omega_{nf}$, esta ecuación muestra que el proceso de emisión está mediado por la probabilidad de la transición cuando un campo eléctrico excita una molécula que produce un dipolo artificial (dipolo inducido), esta transición en el caso de la dispersión Raman es entre los niveles de vibración y rotación, ya que es menos probable que se encuentre un estado excitado, entonces la emisión anti-Stokes también es más débil [49].
Según D.A Long [50], la intensidad de un dipolo eléctrico oscilante inducida en una molécula por un campo eléctrico de frecuencia ω es:

$$I = \lambda\omega_s^4\vec{P}^2 sin^2(\theta) \tag{3-29}$$

donde,

$$\lambda = \frac{1}{32\pi^2\epsilon_0 c_0} \tag{3-30}$$

con ϵ_0, c_0 la permitividad del espacio libre y la velocidad de la luz en el vacío respectivamente. El vector de polarización está relacionado con la polarizabilidad así:

$$\vec{P} = \alpha_{\rho\sigma}\vec{E_{\sigma\rho}}(\omega) \tag{3-31}$$

Reemplazando 3-31 en 3-30 entonces:

$$I = \lambda(\omega \pm \omega_{vib})^4\vec{E_{\sigma 0}}(\omega)|\alpha_{\rho\sigma}|^2 sin^2(\theta) \tag{3-32}$$

La ecuación 3-32 resalta un hecho importante en la dispersión Raman, la longitud de onda incidente es fundamental en el proceso de obtención de señales fuertes, porque responde con la cuarta potencia de la longitud de onda incidente. Para este proyecto, se seleccionó una longitud de onda de láser de 532 nm, aun sabiendo que podría inducir una fuerte fluorescencia; sin embargo, la señal Raman será más fuerte (5 veces) en comparación con el láser de 785 nm más común para este tipo de experimentos. Otra razón para usar el láser de 532nm fueron los detectores, ya que estos están diseñados para funcionar en la región visible del espectro.

Kramers-Heisemberg-Dirac solo explora la forma más común de esparcimiento inelástico, sin embargo, existen más variaciones como el Raman estimulado, el esparcimiento Raman amplificado por superficie (SERS), etc., con diferentes fenomenologías involucradas, por ejemplo, en la dispersión Raman estimulada, el proceso es mediado por uno de los fotones creados, produciendo una especie de reacción en cadena que incrementa la señal Raman en algunos órdenes de magnitud [51], [52], [53]. En el caso del SERS, las razones teóricas aún no están claras, sin embargo en la actualidad existen dos teorías, una explica que la amplificación de la intensidad del campo eléctrico es debida a los adsorbatos presentes en superficies específicas a esta se le conoce como teoría electromagnética del SERS. La otra teoría se denomina hipótesis química la cual atribuye la mejora de la señal a la formación de complejos de transferencia de carga. [54], [55].

3.2.4. Bandas Raman, grupos funcionales y vibraciones

Cada vibración/señal corresponde a diferentes frecuencias; La combinación de las intensidades junto con la posición y el ancho de banda, otorga la posibilidad de establecer la identidad molecular de una muestra. Según [56], una buena manera de simplificar esta tarea es mediante el uso de una nomenclatura que incluya el tipo de vibración.
A las vibraciones de estiramiento de dos átomos diferentes como CH se llaman ν(CH); el tipo de vibraciones producidas por ejemplo por CH_3 o CH_2 se les llaman vibraciones simétricas o de tijera ($\delta(CH_2)$, también existen ($\omega(CH_2)$) mecido, ($\tau(CH_2)$) torsión y ($\rho(CH_2)$ balanceo. Las tipo γ son vibraciones de tres átomos y se conocen como vibraciones fuera del plano. En la siguiente tabla, se incluyen algunas de las vibraciones de los grupos más relevantes y sus respectivas frecuencias Raman de acuerdo con las necesidades de este trabajo, ver tabla **3-3** .

Grupo	Vibración	Frecuencias Raman (cm^{-1})
δ C-C	Vibraciones de tijera en cadenas alifáticas	250-400
ν O-O	Vibraciones de estrechamiento, por ejemplo, peróxidos	845-900
νC-O-C	Vibraciones de estrechamiento, 1,4 β Glucano	800-970
δ CH_2, CH_3	vibraciones simetricas, Quitina	1400-1470
ν C=O	Estrechamiento de Quitina	1680-1820

[57]

Tabla 3-3: Vibraciones Raman de Algunas de las Moléculas de interés en este trabajo

Las bandas Raman presentadas se eligieron en función de las dos moléculas seleccionadas como objetivos para detectar la presencia de un fitopatógeno fúngico, el rango de frecuencias obedece al hecho de que la temperatura puede aumentar o reducir el tamaño de la longitud del enlace. Si se reduce la energía total del ensamblaje molecular (enfriamiento) entonces los picos Raman se moverán hacia números de onda más altos debido a que el desplazamiento hacia adelante y hacia atrás del enlace será más lento dada la reducción de temperatura, en el caso contrario (calentamiento), se aprecia un desplazamiento a números de onda más bajos por la misma razón ya señalada.

3.3. Redes Neuronales Convolucionales

Esta sección trata sobre el procesamiento de datos empleado en este trabajo. La técnica seleccionada fue introducida en 1989 por LeCun et al., para reconocer patrones a partir de dígitos escritos a mano, y fue una mejora de una arquitectura de red neuronal más antigua llamada *Neocognitron* que apareció diez años antes. Un momento histórico en que las aplicaciones prácticas eran muy difíciles de implementar debido a la falta de tecnología apropiada, tal como los sistemas de computación en paralelo de alta velocidad [58].
A continuación se presentará una breve descripción de esta técnica de acuerdo con la interpretación de Google^{TM} - Tensorflow, que fue la plataforma elegida para el análisis de imágenes.

3.3.1. Arquitectura de las redes neuronales de Convolución (CNN)

Este tipo de redes se basan en tres ramas diferentes que son, campos receptivos locales, pesos compartidos y agrupaciones, los cuales son los componentes básicos de las CNN. La primera es cómo se envía la información a la red, en el caso de Tensorflow, una pequeña matriz de Los píxeles y las intensidades de 16x16 suelen ser suficiente, para cada entrada se realiza una operación de convolución 2D, de la siguiente manera 3-33

$$z(j, d_1, d_2) = \alpha[\sum_{c=0}^{C-1}\sum_{u=0}^{r-1}\sum_{v=0}^{r-1} x_{c,d_1+u,d_2+u} \times w_{u,v}^{j}] \tag{3-33}$$

con j=0,1,2,...,n y d_1, d_2 = [0,1,2,...,d]
donde w son los pesos en j diferentes conjuntos compartidos por varias neuronas con r campos receptivos trabajando en cada entrada xy, α es el parámetro de rapidez de aprendizaje [59]. Finalmente, el concepto de agrupación se refiere a un proceso de filtrado de cada muestra, la operación de procesamiento más popular es la agrupación máxima en la que cada imagen se reduce a una más pequeña compuesta por rectángulos que no se superponen con los valores más altos de intensidad. Hay otras operaciones como agrupación promedio o agrupación de normas entre otras [60].

Otra de las características importantes de las CNN son las capas ReLU, totalmente conectadas y con pérdida, que proporcionan a la red neuronal la capacidad de aprender.
ReLU significa unidad lineal rectificada y funciona como una función de activación que devuelve el valor máximo de un intervalo, eliminando en el proceso los valores negativos, hay otras funciones de activación como tanh(x) o la función sigmoide [61]. El nivel más alto de razonamiento en la red neuronal se realiza a través de capas totalmente conectadas y es el lugar donde se lleva a cabo el proceso de propagación hacia atrás y propagación hacia adelante [62].

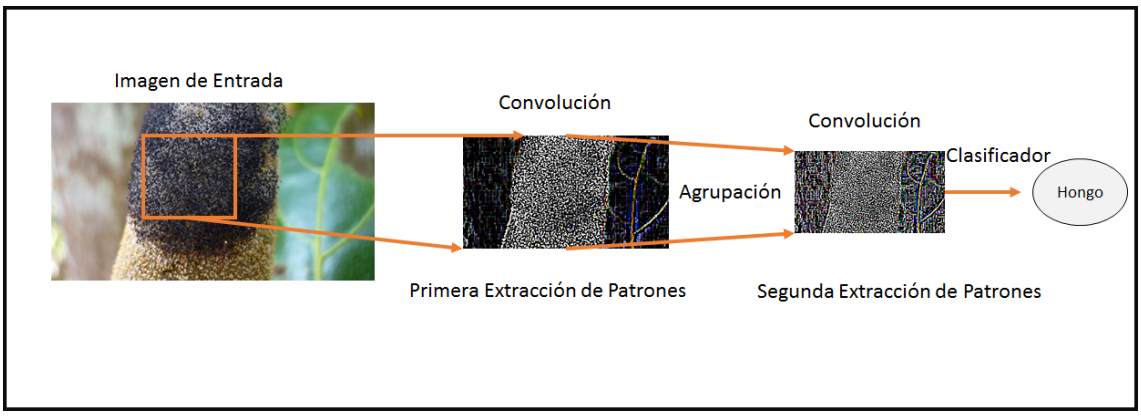


Figura 3-9: Esquema general de una red neuronal convolucional

El esquema mostrado en la figura **3-9** resume todo el proceso de obtención de un mapa de características, las operaciones de convolución sobre la imagen ayudan al sistema a reducir la cantidad de información, extrayendo solo la forma (convolución) de la imagen, de esta manera el sistema puede producir un patrón asociado a una etiqueta de clasificación y finalmente dar una respuesta acerca de qué tan similar es la imagen seleccionada al patrón aprendido. En nuestro caso, para determinar si hay alguna presencia del fitopatógeno fúngico sobre un sustrato.

4 Diseño y Desarrollo

En este capítulo, se explica el proceso creativo paso a paso detrás de este proyecto, a partir de una descripción de los diferentes módulos que lo componen acompañados de la razón científica que da lugar a esa selección específica. El siguiente diagrama de flujo **4-1** resume la conexión entre los diferentes módulos y su interacción para obtener las señales y las imágenes para la identificación de fitopatógenos fúngicos. En la figura, hay cuatro secciones; El software de control, configuración óptica, procesamiento de imágenes y electrónica.

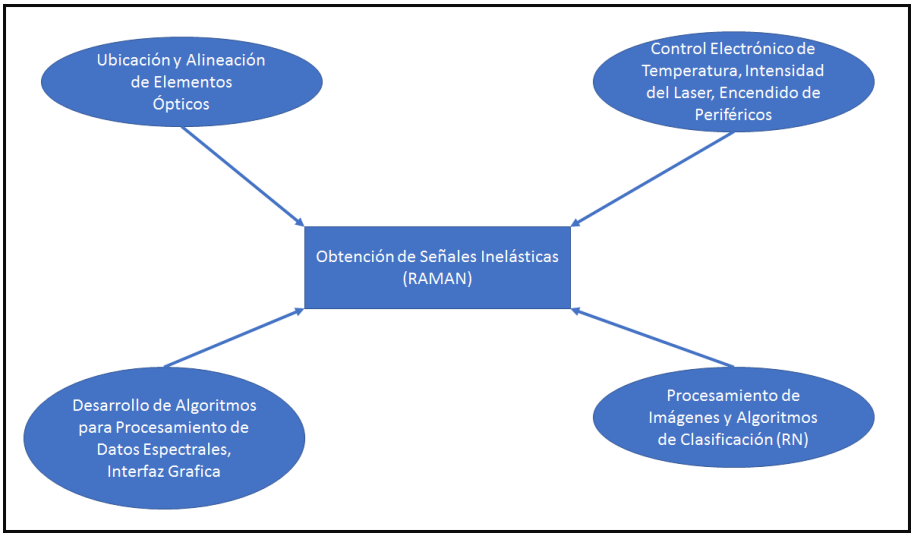


Figura 4-1: Diagrama de Flujo de Actividades

Todas las unidades del sistema fueron diseñadas enteramente en el laboratorio; el software de control fue diseñado para medir o controlar algunas variables en tiempo real y también el encendido y apagado de los módulos, por lo que se creó una interfaz electrónica utilizando un Arduino para sincronizar la PC con el instrumento. Los scripts utilizados para el reconocimiento de imágenes y el proceso de datos del espectrómetro se realizaron en la plataforma Tensorflow de I.A de Google y en python.

Los experimentos con cepas de hongos se realizaron en el laboratorio *"Venenos Naturales"* de la Universidad Nacional de Colombia, con una larga tradición en el manejo y experimentación con todo tipo de microorganismos.

4.1. Selección del Fitopatógeno y técnica espectroscópica

Una selección apropiada del microorganismo fitopatógeno fue un punto clave de este trabajo, por esa razón los criterios consistieron en encontrar una familia de compuestos que podrían estar asociados con una clase particular de organismos, así mismo si el espécimen es una fuente importante de contaminación de cultivos económicamente significativos. Ambas características fueron cumplidas por varias cepas, pero las de origen fúngico fueron las más interesantes en términos de composición química, porque algunas de sus moléculas son raras en otros microorganismos y afectan a productos muy importantes como el banano y el aguacate. Como se mencionó en los capítulos anteriores, *Colletotrichum gloeosporioides* y *Mycosphaerella fijiensis* Morelet fueron los elegidos debido a su relevancia económica y la presencia de quitina y α 1,3 glucano, que son moléculas que revelan la presencia de una cepa fúngica. Es cierto que ambas moléculas son una característica de la pared celular de casi cualquier hongo, pero funcionan como una alarma de la presencia potencial en un lugar en el que no deberían estar, como las cascaras de los frutos en el caso de *Colletotrichum gloeosporioides* o las hojas de los cultivos de banano y plátano para el otro microorganismo seleccionado. La técnica de detección se seleccionó pensando en la selectividad de las moléculas por esa razón, se estudiaron tres opciones diferentes, LIBS (espectroscopia de ruptura inducida por láser), IR y Raman, todas las mencionadas anteriormente pueden obtener una huella espectral de cualquier grupo molecular, las diferencias entre estas surgen de la manipulación de la muestra y el costo económico de la implementación, por estos motivos, se descartaron LIBS e IR, el primero es muy costoso y difícil de sincronizar porque el espectro se toma solo unos microsegundos después de que el plasma comienza a enfriarse y es necesario un láser de alta energía para inducir la ruptura de los enlaces moleculares [63]; en el caso de IR, los costos no son el problema, sino el manejo de la muestra, que debe secarse y dificulta el proceso de detección debido a la naturaleza de las muestras [64]. Por lo tanto, la única técnica sin preparación de muestras y costos razonables para esta aplicación fue la dispersión Raman. En la siguiente sección se detallan los procesos de diseño y construcción.

4.2. Proceso de Diseño y Construcción

Al conocer las moléculas y su respectiva respuesta espectral, y haber seleccionado Raman como medio de detección principal, se seleccionaron las partes ópticas. El dispositivo tiene un sistema de filtros, paso banda para limpiar la línea láser a 532nm con un FWHM = 3nm Thorlabs^{TM}, dos filtros Omega Optical^{TM} centrados en 557.8 FWHM = 2nm OD = 3 y 578nm FWHM = 8nm OD = 4, para capturar las señales de $\beta 1,3-$ glucan y quitina respectivamente; un filtro pasa altos para doblar el rayo láser 90 (Longpass con una longitud de onda de corte a 540nm) que se extrajo de un proyector de video y un filtro EDGE (Omega Optical^{TM}) pasa altos con longitud de onda de corte a 540 nm, OD = 5. Las lentes y las piezas mecánicas se obtuvieron de un microscopio y un estereoscopio que se modificaron para encajar uno en otro, ambos instrumentos se reciclaron de otros laboratorios de la Universidad Nacional de Colombia y se sometieron a un proceso de mantenimiento que consistió en la limpieza y realineación de las partes ópticas de el estereoscopio y la extracción de la parte superior del microscopio, junto con una expansión del soporte para ajustar el estereoscopio allí, la idea era utilizar la etapa de movimiento xy y el tornillo micrométrico del microscopio junto con la parte óptica del otro instrumento que es más adecuado para el estudio de estructuras más grandes como hojas, frutas y similares. Además, se agregó una etapa de refrigeración, con la intención de controlar la temperatura de la muestra; algunos hongos, por ejemplo, necesitan condiciones especiales de temperatura, como *Botrytis cinerea* Pers., cuya temperatura de crecimiento óptima está entre (12-28)°C [65] y que también es un espécimen potencial para detectar ya que ataca Los diferentes tipos de cultivos de flores en Colombia, uno de los más importantes en términos económicos [66]. El sistema de enfriamiento consiste en una celda Peltier conectada a un Arduino que a través de una interfaz controla la energía suministrada a través de un controlador PID el cual fue sintonizado manualmente y un termistor que detecta la temperatura cada 450ms, este parámetro es enviado desde una interfaz de software desarrollada en Python. El termistor es un sensor adafruit ®optimizado y calibrado para la plataforma Arduino. Todos sus parámetros se suministraron para obtener la temperatura en tiempo real, por sustitución en la ecuación de Steinhart-Hart 4-1,

$$T = \frac{\beta}{ln\frac{R}{r_\infty}} \tag{4-1}$$

con $\beta = 4090$, $R = (\frac{1025-10}{a} - 10)/10 + \beta$, a = 1023-Conversion-Analogo-Digital, luego el resultado de esta ecuación se reemplazó en el controlador PID, con los parámetros P = 250, I = 15, D = 20; Para garantizar una respuesta agresiva del sistema lo que significa una rápida corrección de la temperatura. El control se realizó utilizando la biblioteca PID del software de desarrollo de Arduino que resuelve las ecuaciones diferenciales PID.

El siguiente paso fue crear una etapa de potencia para proporcionar el control de energía de la celda Peltier. La principal característica de estos dispositivos es el alto consumo de corriente, por ese motivo, se seleccionó un IRL 540, que es un MOSFET adecuado para manejar

corrientes de hasta 36A y una carga de voltaje de 100V. El funcionamiento del circuito consiste en una señal PWM proveniente del puerto Arduino 9, esta señal es el resultado de resolver una ecuación diferencial relacionada con PID y proporcional la cantidad de potencia necesaria para acercarse al setpoint seleccionado. Si la diferencia es positiva, el Peltier se enciende, de lo contrario se apaga, el sistema no cuenta con un sistema de calefacción. En la figura **4-2** se muestra el esquema completo del circuito. La piedra angular del sistema

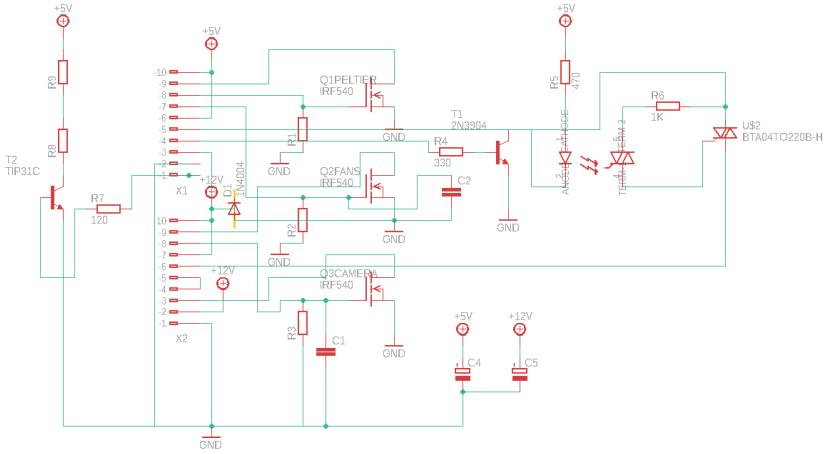


Figura 4-2: Esquema Circuital del Sistema de Estereoscopio Inelástico

de enfriamiento es el intercambiador de calor que consiste en un sistema de bombeo de refrigerante, conectado al disipador de calor de la celda Peltier, para aumentar la eficiencia como consecuencia de mantener fresco el lado caliente, esto puede explicarse por el modelo termoeléctrico.4-2.

$$-\dot{q_{pumped}} = \nabla.(\kappa \nabla T) + J.(\sigma^{-1}J) - TJ.\nabla S \tag{4-2}$$

Todo el calor extraído del sistema es consecuencia del coeficiente de Seebeck (S), un tipo de coeficiente fenomenológico cruzado de Onsager que relaciona el flujo de corriente eléctrica con la energía potencial calórica, este término debe superar el de conducción de calor (primer término del lado derecho de la ecuación) y el término de calentamiento de Joule ($J.(\sigma^{-1}J)$) que es un coeficiente de Onsager directo que toma en cuenta el calor producido por el transporte eléctrico proporcional a la conductividad de el medio (σ) [67]. Teniendo en cuenta todo lo anterior, se tomó la decisión de utilizar enfriamiento líquido en conjunto con el disipador de calor. La imagen **4-3** muestra cómo se ve el acoplamiento con el disipador de calor de la celda Peltier. Todas estas consideraciones se hicieron para otorgar al sistema la posibilidad de mantener constante o reducir la temperatura de la muestra, esto es importante para algunos tipos de hongos como *Botrytis cinerea* Pers, debido a que su temperatura óptima de crecimiento es entre (12- 28) ^oC como se ha mencionado anteriormente y abre una amplia

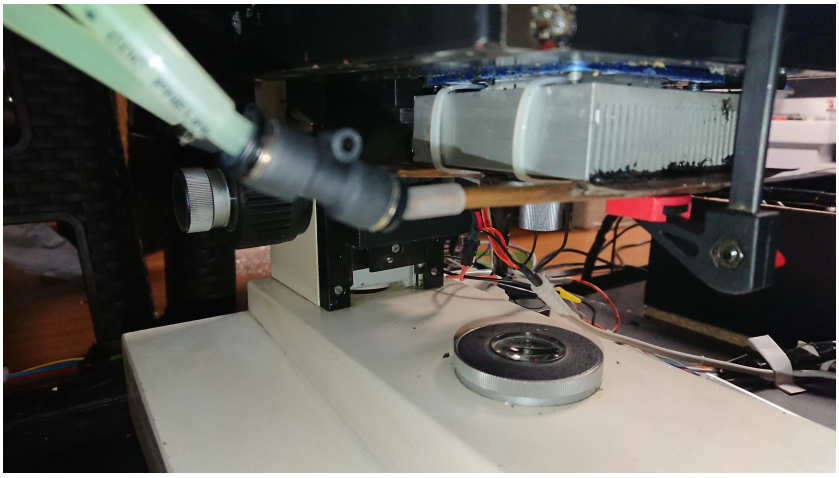


Figura 4-3: Montaje de Refrigeración Liquida

gama de opciones cuando es necesario mantener vivo el espécimen, por ejemplo, cuando produce un metabolito de interés.
Además del control de temperatura, el circuito también proporciona todas las interfaces necesarias para tener control de el resto de periféricos; como bomba, cámara, potencia del láser, ventiladores de soporte y la lámpara del microscopio. Todos los sistemas están controlados por una interfaz en Python que envía señales al Arduino, para activar cada uno de los transistores del MOSFET, en los puertos utilizados para esta tarea los cuales fueron 22, 24, 30, 6 y 9.

Interfaz de Python

Se creó una interfaz para tener el control del sistema desde la PC, esta GUI (interfaz de usuario gráfica) permite ajustar la potencia del láser, temperatura de la muestra y su monitoreo en tiempo real, también es posible ajustar la intensidad de la lámpara del microscopio y encender y apagar el sistema de refrigeración liquida logrando temperaturas de hasta 10°C.
El script se realizó utilizando una arquitectura de subprocesos múltiples (multithreading) para hacer posible la ejecución de diferentes procesos a la vez, en este caso esto fue necesario por la necesidad de conocer la temperatura de la muestra todo el tiempo y controlar la potencia del láser de forma simultanea, y algunas veces también, el encendido de la lámpara del microscopio.

Este paradigma de programación utiliza todos los núcleos del procesador disponibles para realizar un procesamiento paralelo, cada una de estas tareas se les denomina hilo.
La interfaz gráfica se diseñó en QT5, un entorno gráfico para múltiples plataformas que proporciona una manera estable de ejecutar la misma interfaz de usuario en diferentes sistemas operativos, como Windows, Linux, Android, etc. La interfaz diseñada para este trabajo se

muestra en **4-4**, el botón de selección con la etiqueta de la cámara, controla el encendido del sensor CCD, setPoint permite al usuario seleccionar una temperatura inferior a la ambiental. La potencia controla la intensidad del láser como un porcentaje de la potencia total de esté (200 mW, 532nm TianGreen textregistered), el botón deslizante controla la intensidad de la lámpara del microscopio, el botón ON/OFF enciende los ventiladores, la bomba y la etiqueta marcada como temperatura, muestra en tiempo real la temperatura de la muestra.

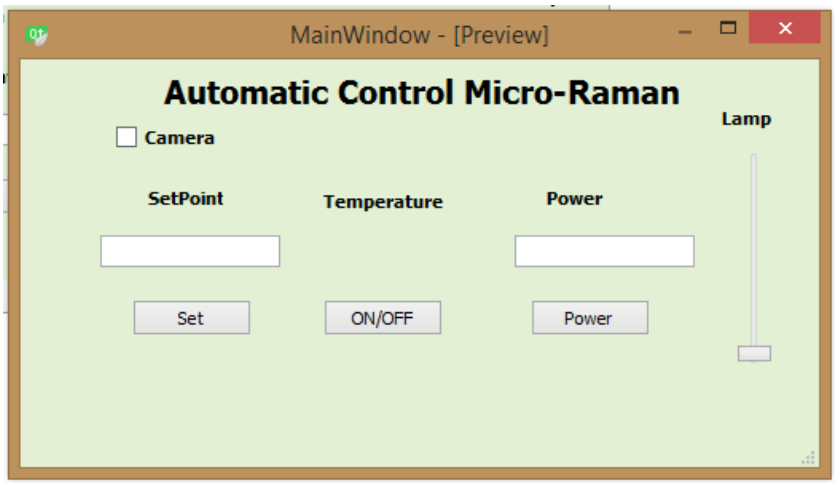


Figura 4-4: Interfaz Gráfica del Estereoscopio Raman

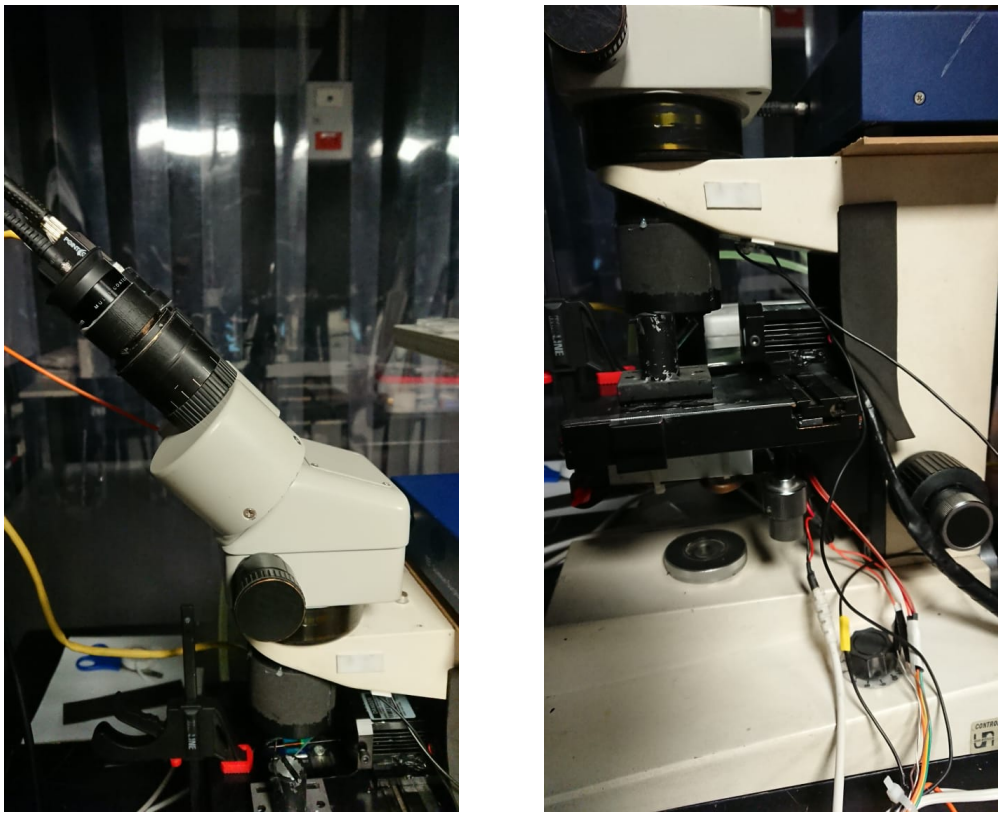
Todas las señales relacionadas con el control de potencia o intensidad se interpretan como PWM (modulación de ancho de fase) por Arduino, que toma el número enviado por la computadora y luego lo transforma en una escala (0-255). Luego, el microprocesador calcula el tiempo de encendido y apagado durante un período de tiempo constante, por ejemplo, si la señal enviada por la computadora es 50, en la escala de 8 bits para el PWM del Arduino, esto significa 127, que para el período de 25 μs preestablecido significa, que durante 12.5 μs el puerto está en estado Alto (encendido) y los otros 12.5μs está apagado, lo que le otorga la mitad de la potencia total para este ejemplo. Lo anterior se puede expresar mejor con la ecuación de potencia promedio sobre un ciclo de trabajo 4-3

$$P_{avg} = \frac{1}{T}\int_0^T p(t)dt \tag{4-3}$$

donde p (t) es la potencia de una señal en el instante t, con el período T [68].
Todo lo mencionado anteriormente justifica el uso de subprocesos múltiples, debido a la cantidad de tareas que ocurren simultáneamente y la necesidad de controlar la temperatura de la muestra todo el tiempo, lo que agrega un gran uso de recursos a la máquina que también necesita ejecutar el software del sensor CCD, la cual es una aplicación muy exigente debido a que sigue un protocolo Gigabit Ethernet para la transferencia de datos. Por todas estas razones, era necesario un esquema de programación en paralelo [69].

4.3. Montaje Óptico

La configuración óptica consta de dos microscopios diferentes, un estereoscopio Boeco^{TM} y un microscopio Rossbach ^{TM}, el primero se usó como etapa óptica, brinda una magnificación de 10x la cual es adecuada para estudiar lesiones en cáscara y hojas permitiendo un campo visual mas amplio lo que permite encontrar de forma más probable (ver figura, **4-5** a)) un proceso de infección por una cepa fúngica, por supuesto, un proceso infeccioso podría ocurrir incluso sin lesiones, pero es un buen punto de partida para intentar detectarla [70]. El microscopio Rossbach^{TM} por su parte, se utilizó para suministrar la etapa de posicionamiento del sistema óptico, específicamente para ajustar el láser y la alineación de la muestra (figura **4-5** b)), el mecanismo (x,y) y los tornillos macrométrico y micrométrico se conservaron.



(a) Sistema Óptico (b) Etapa Mecánica

Figura 4-5: Etapas Mecánica y Óptica del Dispositivo

Los oculares del microscopio Boeco^{TM} se acoplaron con un espectrómetro y una cámara. El primero es un B& W Tek^{TM} (Modelo BTC-110S), modificado y calibrado por los autores de este trabajo, el proceso de calibración consistió en producir diferentes longitudes de onda

con un espectrofotometro UV Shimadzu UV/VIS 160A^{TM}, LEDs con longitudes de onda conocidas, también se emplearon un par de láseres, He-Ne (632.8nm) y un diodo láser de 532nm de TianGreen. Una vez registrada la señal se toman las coordenadas de los píxeles con su respectiva intensidad, después de 15 mediciones diferentes, los datos se interpolaron usando regresión polinómica de orden 2. Luego, se verificó la precisión del espectrómetro contra fuentes de longitud de onda conocidas, para verificar la confiabilidad del instrumento.

La cámara es un PT-GREY ^{TM} textit Flea 3 Gigabit Ethernet, que contiene un sensor CCD ICX 655 Sony ^{TM} con una eficiencia cuántica del 50 % para el canal verde, 54dB de rango dinámico y un ruido oscuro temporal de 7.45 e^{-} cite Visión, esas especificaciones son ideales para obtener señales de intensidad óptica muy bajas, como las que se pueden esperar en la espectroscopia Raman. Este sensor vino sin lentes de enfoque, por eso se usó la lente objetivo 4x del microscopio Rossbach para producir las imágenes.
Finalmente, la etapa de filtrado está conformada por la siguiente lista de elementos, ver tabla **4-1**,

Filters Parameters					
Brand	**Reference**	**FWHM (nm)**	**% T**	**OD**	**CWL (nm)**
Thorlabs	FL532-3 - Ø1"Laser Line Filter	3	>80	6	532
Omega.Inc	557.7BP5	5	>80	3	557.7
Omega.Inc	XCC578BP8	8	>90	5	578
Omega.Inc	RapidEdge 540LP	NA	>90	5	540*
Epson	RGB Projector Filter	NA	>80	NA	550*

Tabla 4-1: Caracteristicas Espectrales de Los Filtros

[1]

[71].

El espejo dicroico fue extraído de un viejo proyector Epson^{TM}; Es un filtro pasa altos cuya función original fue dividir los canales rojo y verde. Su respuesta espectral a 45° permite que la radiación sobre 550nm pase y refleja todo lo que está por debajo (rango visible), la respuesta espectral de este filtro puede apreciarse en la figura **4-6**. Es importante mencionar que el láser se modificó para obtener un spot más ancho, al quitar las lentes de colimación. Originalmente, la mancha tenía un diámetro de 5mm, pero después de la modificación, el tamaño aumentó a 9 mm, debido al ángulo de divergencia (5°).

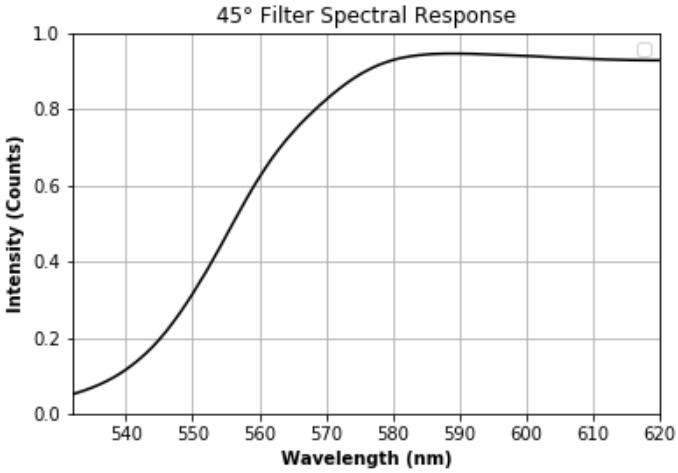


Figura 4-6: Respuesta Espectral del Filtro a 45°

4.4. Reactivos Químicos Probados

La siguiente tabla **4-2** muestra los diferentes reactivos utilizados en este trabajo junto con la marca, la presentación del producto y la utilidad dentro del proyecto.

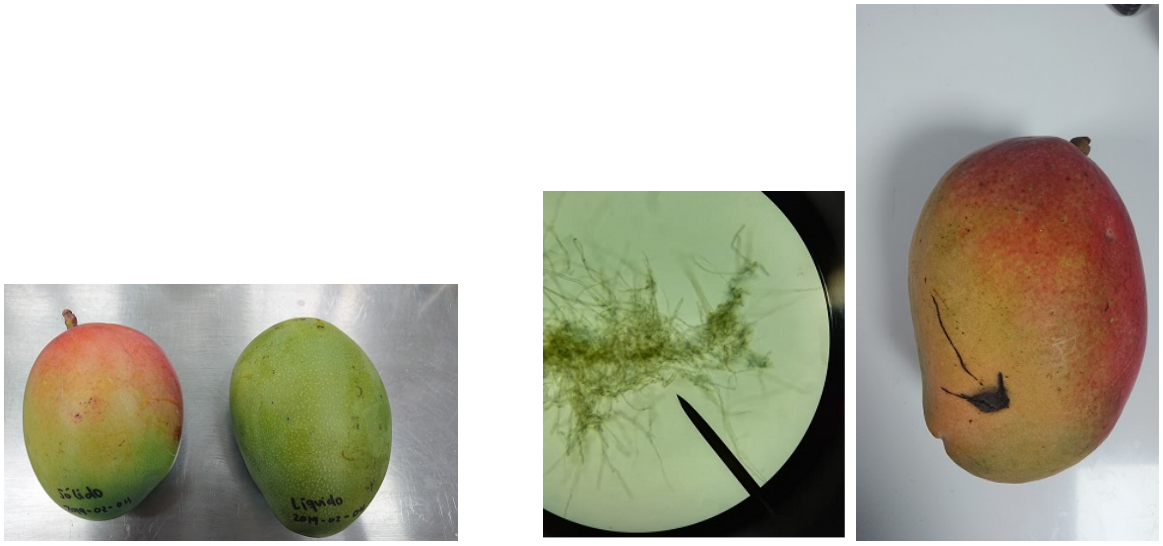
Lista de Reactivos			
Compuesto	**Marca**	**Presentación**	**Utilidad**
Peróxido de Hidrógeno	Protokimica	35 % p/v	Sustancia de Calibración
Carburo de Silicio	Merck	Polvo	Sustancia de Calibración
Inositol	Merck	Cristales	Sustancia de Calibración
Ácido Palmitico	Sigma	Cristales 99 %	Sustancia de Calibración
Carbon Activado	Merck	Polvo	Sustancia de Calibración
PDA Agar y caldo	Sharlau	gel y liquido	Medio de cultivo
Leucina	Carbonwax	Cristal	Sustancia de Calibración
Ácido Oxálico	Merck	Polvo	Sustancia de Calibración

Tabla 4-2: Reactivos empleados en la validación del equipo

4.5. Protocolos Microbiológicos

En este trabajo se utilizaron dos cepas de hongos diferentes, *Colletotrichum gloeosporioides* Penz. y *Mycosphaerella fijensis* Morelet. El primero fue suministrado por el Laboratorio de venenos naturales de la Universidad Nacional de Colombia sede Medellín, el otro fue aislado y preparado en el Laboratorio de Biología Celular y Molecular de la misma institución.
Colletotrichum gloeosporioides se cultivó en agar PDA (Sharlau^{TM}) hasta su esporulación, luego se agregaron 20 ml de solución salina (85 %) junto con una mezcla de 0.1 % tween 80 en la placa de Petri, luego fue cuidadosamente raspado para extraer las esporas. Con esta solución se prepararon diluciones de 10^{-1} a 10^{-3}, luego se observaron las esporas y se contaron en la cámara de Neubauer. El siguiente paso fue realizar una pequeña incisión para

inocular 1 ml de la solución de 10^{-3} en una de las tres frutas de mango seleccionadas previamente para este experimento. Se inoculó otro espécimen, esta vez con agar sólido. El tercer fruto fue designado como control. Después de tres días de incubación en cámara húmeda y a temperatura ambiente, los primeros síntomas aparecieron como se muestra en la figura **5-33**.



(a) Muestras de Mango

(b) Hifa de *C.gloeosporioides*

(c) Mango Infectado con *C.gloeosporioides*

Figura 4-7: Muestras de Mango

Se cortaron dos cuadrados pequeños, uno del control y el otro de la fruta infectada, después de eso, se tomaron varias imágenes y espectros, del tejido sano e infectado.

4.6. Condiciones Experimentales

La siguiente tabla **4-3** muestra las condiciones y los parámetros técnicos de la configuración en cada uno de los experimentos.

Tabla 4-3: Condiciones Experimentales para Cámara, Láser y Espectrómetro

Experimental Conditions				
Experimento	**Potencia del Láser %**	**Tiempo de Exposición (Cámara) (s)**	**Tiempo de Integración (Espectrómetro) (s) y (Promedios)**	**Ganancia Cámara (dB)**
Peróxido de Hidrógeno	45	1.45	21 (3)	26
Carburo de Silicio	100	1.45	20 (3)	26
Myo-Inositol	65	1.45	12 (3)	20
Ácido Palmitico	100	1.45	30 (3)	26
Carbon Activado	100	1.45	50 (3)	26
Acido Oxalico	100	1.45	5 (3)	26
Leucina	100	1.45	8s (5)	24
Colletotrichum gloesporoides	45	1.45	7 (5)	24
Mycosphaerella fijiensis	45	1.45	7s (5)	24
Water	100	1.45	N/A	26

4.7. Algoritmos de Aprendizaje Profundo y Códigos Python

Las técnicas de aprendizaje profundo denominadas redes neuronales de convolución se realizaron con la intención de identificar la presencia del fitopatógeno sobre las hojas en el caso de *M.fijiensis* o en los frutos para *C.gloeosporioides*, el proceso consistió en alimentar el algoritmo con muchas imágenes diferentes, en diferente posición e intensidad variable de iluminación en las regiones con las características de interés (lesiones) para cada una de las

muestras, *M.fijiensis* fue estudiado con dos filtros diferentes sobre placas de agar. Esto se hizo por simplicidad, ya que es muy difícil de aislar y el proceso de infección en las hojas suele durar días.
Para *C.gloeosporioides* se buscó detectar directamente de la cáscara del mango, pero esta vez con un solo filtro, esto con el fin de establecer la configuración mínima necesaria para efectuar esta tarea. El entrenamiento de la red neuronal comienza con 100 imágenes para cada una de las etiquetas, en *M.fijiensis* fueron agar557, agar578, Mycos557, Mycos578. Para *C.gloeosporioides* se eligieron, background y colleto557, que corresponden a tejido sano y enfermo respectivamente, las imágenes se tomaron solo con el filtro de 557nm. Después de esto, la última capa de la red neuronal se inicializó con los siguientes parámetros, vea tabla **4-4**; Todo el proceso de entrenamiento de la RNC (Red neuronal de convolución) tomó 2

Parametros de la Red Neural	
Parámetro	**Valores**
Learning Rate	0.1
Training Batch Size	-1
Training Steps	10000
Validation Batch Size	-1
Random Scales	0
Tensor Flow Module	pnasnet_large

Tabla 4-4: Parámetros para reconocimiento de imágenes

horas en cada caso, en una CPU Corei7 ^{TM} con 12 GB de RAM, en esta ocasión no fue necesario hacer uso de núcleos CUDA de la tarjeta gráfica.

Scripts de Python

Todos los scripts creados o modificados para este trabajo se pueden encontrar en el apéndice. Algunos de ellos serán explicados brevemente en esta sección. El primero y probablemente el algoritmo más importante es el relacionado con el procesamiento de datos espectrales, ya que el espectrómetro utilizado no proporciona una herramienta de visualización adecuada, por esa razón se creó una aplicación para capturar los datos en el formato nativo del espectrómetro, que es un archivo .csv (valores separados por comas) con la información del número de píxeles, la longitud de onda, la intensidad en conteos, el espectro promedio y el ruido. Este script resta la información del ruido eléctrico y térmico, realiza la conversión de longitudes de onda a desplazamientos Raman en cm^{-1}, luego toma el valor más alto de la amplitud y normaliza todos los datos a la unidad, a partir de allí la información se gráfica utilizando un aplicativo que permite personalizar, agregando notas, seleccionando el rango de ejes, agregando etiquetas, suavizando las curvas entre otras características.

Los scripts para el procesamiento de imágenes se diseñaron para mejorar la intensidad, trazar los histogramas 2D y seleccionar solo la zona iluminada de todo el campo de visión de la cámara (todos los algoritmos utilizan la biblioteca openCV de python) [72].
La secuencia de comandos para la mejora de la intensidad intenta simular un tiempo de exposición más largo, al agregar varias imágenes de baja intensidad, este proceso necesita dividir los canales RGB, sumar cada una de las imágenes y luego tomar el promedio, después, se fusionan los canales nuevamente y se crea Una nueva imagen con colores intensificados. La operación de enmascaramiento se realizó para seleccionar solo la zona del campo de visión de la cámara que está iluminada por el láser para evitar el ruido o señales no deseadas ajenas al fenómeno de interés.

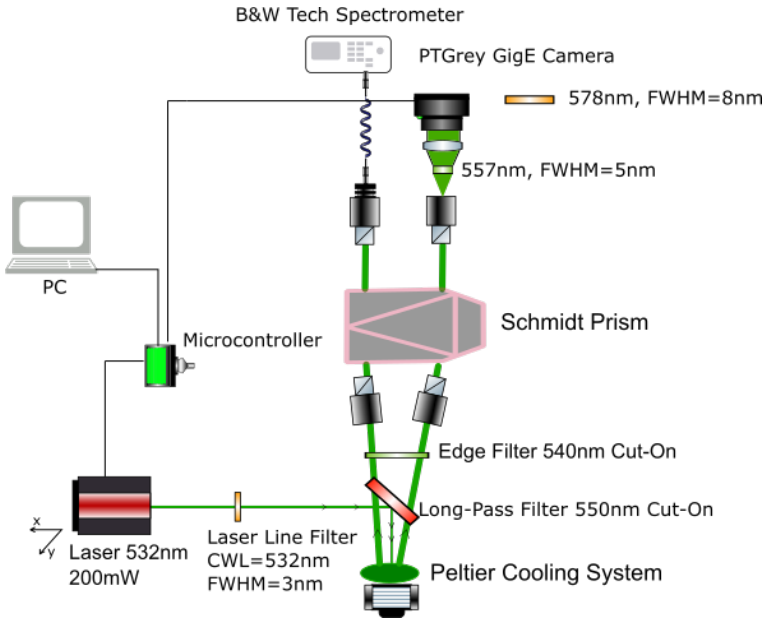
Los histogramas 2D son necesarios para analizar la relación entre los canales RGB, esto es importante porque proporciona información cualitativa sobre la región espectral activa, por ejemplo, si los canales rojo y verde tienen la misma frecuencia en la aparición de los datos, entonces la longitud de onda más probable está en la región amarilla de El espectro visible, que es en la única en la cual ambos canales se traslapan.

5 Validación del Equipo y Detección de Fitopatógenos

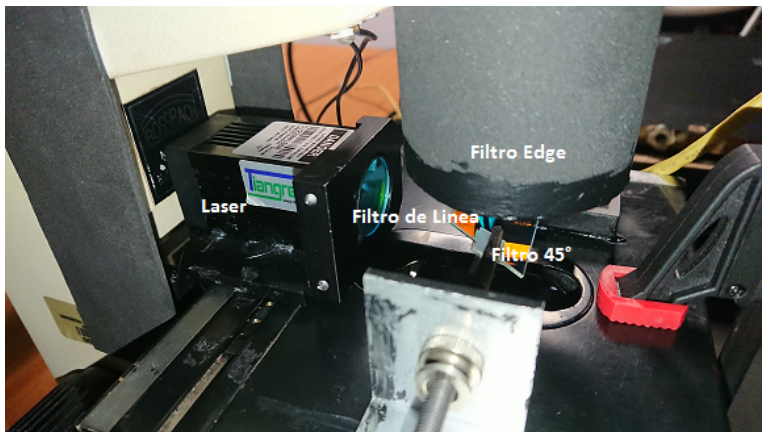
El principal resultado de este trabajo es el instrumento como tal, la técnica y la aplicación es la manera de mostrar la relevancia de esta investigación y probar el funcionamiento correcto del dispositivo, sin embargo, es posible utilizar el instrumento para otros fines, ya sea como un equipo Raman regular o un instrumento para generación de LIF (fluorescencia inducida por láser) por lo cual se puede identificar cualquier otra sustancia de interés más allá de la aplicación propuesta.
Para mostrar su funcionamiento se han tomado varios espectros de diferentes sustancias para reproducir los resultados que se encuentran en diferentes bases de datos espectrales o publicaciones relacionadas, por ejemplo, la base de datos espectral para compuestos orgánicos SDBS `https://sdbs.db.aist.go.jp/sdbs/cgi-bin/cre_index.cgi` de esta forma es posible garantizar la idoneidad de este instrumento para los objetivos aquí trazados.

5.1. Diseño, Desarrollo y Calibración

La figura **5-1** muestra los componentes y los diferentes elementos ópticos del dispositivo, que como se había mencionado anteriormente, está compuesto por un diodo láser de 532nm con un filtro de línea láser Thorlabs, CWL = 532 ± 0.6 nm, FWHM = 3 ± 0.6 nm, el cual busca mejorar la respuesta espectral del láser, a partir de allí, el haz es desviado por un filtro pasa altos, con longitud de onda de corte de 550nm, una vez golpea la muestra, se produce una excitación en las moléculas que la componen, generando de forma temporal un dipolo inducido, este dipolo puede oscilar con la misma frecuencia de el fotón incidente, en cuyo caso, se trata de el fenómeno de esparcimiento Rayleigh [73], O puede emerger con un valor de energía ligeramente distinto, que es dependiente del tipo de moléculas presentes en la muestra, precisamente a este tipo de esparcimiento es al que llamamos, inelástico, el cual no se debe confundir con la emisión de fluorescencia, la cual siempre emite a la misma longitud de onda, independientemente de la longitud de onda de excitación [74], pese a esta marcada diferencia, es uno de los principales obstáculos a la hora de detectar una señal Raman. Una vez que se da la emisión, es necesario eliminar o atenuar el esparcimiento Rayleigh, dado que solo uno de cada mil fotones es inelástico, por lo cual esta señal sera percibida como ruido en comparación con la amplitud de la señal elástica [75].
Para lograr esto, se cuenta con un par de filtros, el primero, es el mismo que desvía el haz



(a) Esquema Óptico



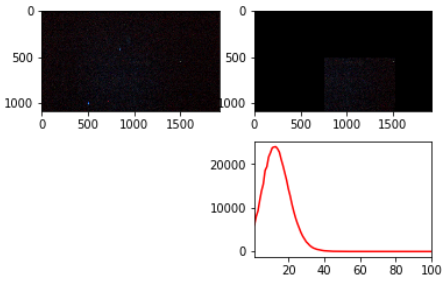
(b) Sección de Filtrado

Figura 5-1: Micro Raman Setup

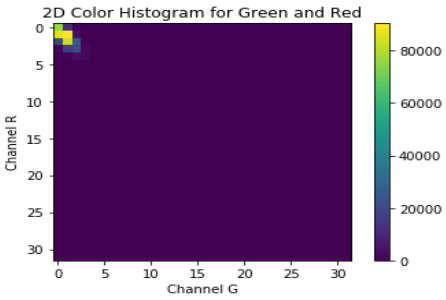
90° y cuya respuesta espectral fue presentada en el capitulo anterior, sin embargo este filtro, no bloquea de forma eficiente toda la radiación Rayleigh, por lo cual un segundo filtro se posiciono en la parte de arriba, para esto fue necesario diseñar un soporte especial el cual fue construido en el laboratorio y posteriormente adaptado a la plataforma de desplazamiento x,y del microscopio. En la figura se muestra también un prisma tipo Schmidt el cual fue necesario desplazar y alinear, para corregir defectos en el campo de visión del instrumento. Para cada uno de los oculares, fue necesario realizar modificaciones para adaptar en uno de ellos la fibra óptica del espectrómetro y en el otro para acomodar la cámara y los filtros. Para el sensor CCD fue necesario emplear una lente negativa con el fin de capturar la imagen de la muestra producida por el equipo y cubrir la mayor parte del sensor, este elemento fue extraído del microscopio Rossbach^{TM}. Como ya se menciono el objetivo es capturar ciertas bandas espectrales, correspondientes a dos moléculas claves, que determinan la presencia de hongos, para lo cual se emplearon dos filtros pasa banda, con longitudes de onda centrales (CWL) a 557nm y 578nm. El primero por su tamaño se acoplaba directamente sobre la lente del sensor CCD, mientras que el segundo se alojaba sobre el ocular del microscopio. Ambos filtros permiten terminar de eliminar la radiación Rayleigh residual y al mismo tiempo producir una imagen espectral en la región de interés. De lado del espectrómetro, fue necesario realizar impresiones 3D, con el fin de adaptar la fibra óptica al ocular del microscopio.

El paso a seguir fue determinar la intensidad mínima a la cual la cámara y el espectrómetro detectaban una señal, lo interesante de esto, fue encontrar que la cámara es varios ordenes de magnitud mas sensible que el espectrómetro, en parte se debe a la calidad del sensor, el cual posee una eficiencia cuántica en el canal verde de 48 % y un rango dinámico de 56.22dB [76], lo que implica una relación de 100.000:1 [77], mientras que en el caso del sensor del espectrómetro la relación es de tan solo 1300:1 [78], lo que explica la marcada diferencia entre el rango mínimo de intensidad que pueden detectar ambos dispositivos. A continuación se presentan algunas fotos con su correspondiente histograma 2D de intensidad (ver figuras **5-2**, **5-3**), que muestra la mínima intensidad del láser que es detectable por la cámara, en este caso se utilizo como muestra espuma negra de etilvinilacetato por su débil señal inelástica

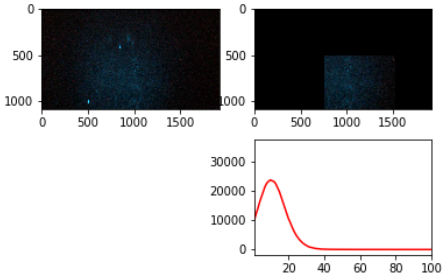
Como se pudo apreciar, con una intensidad del 20 % el instrumento comienza a obtener una señal apreciable, a pesar de que en la foto, no se percibe ningún tipo de imagen, esto es bastante evidente si se compara con las intensidades de láser superiores (80 % y 100 %), este ensayo permitió definir la potencia mínima a la cual se puede ajustar el láser, lo cual es bastante útil cuando se requiera evitar quemar las muestras, algo necesario en muestras biológicas como las que conciernen a este trabajo.



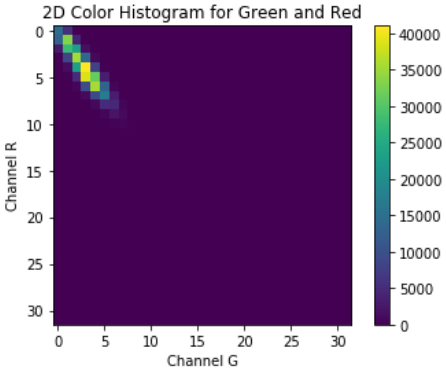
(a) Intensidad del Láser 20 %



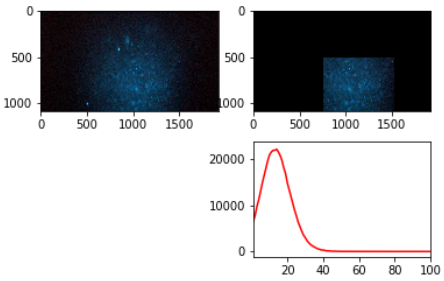
(b) Histograma 2D láser al 20 %



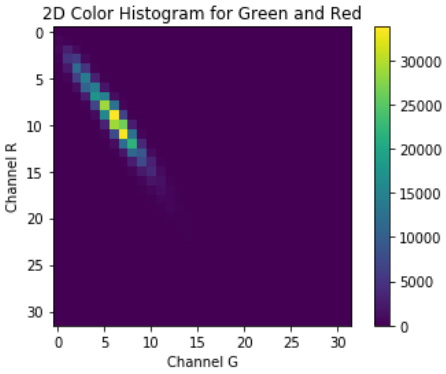
(c) Intensidad del Láser 30 %



(d) Histograma 2D Láser 30 %

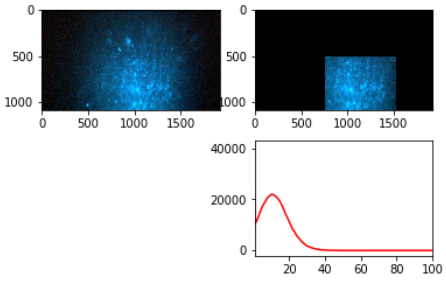


(e) Intensidad del Láser 30 %

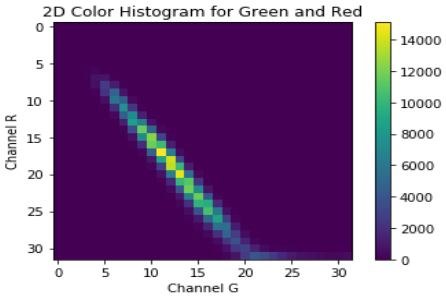


(f) Histograma 2D Láser 40 %

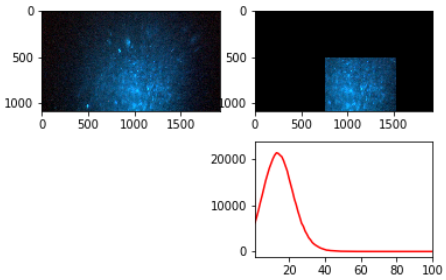
Figura 5-2: Niveles mínimos de detección del instrumento 1



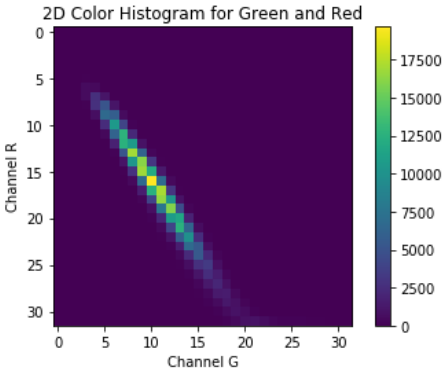
(a) Intensidad del Láser 80 %



(b) Histograma 2D láser al 80 %



(c) Intensidad del Láser 100 %



(d) Histograma 2D Láser 100 %

Figura 5-3: Niveles mínimos de detección del instrumento 2

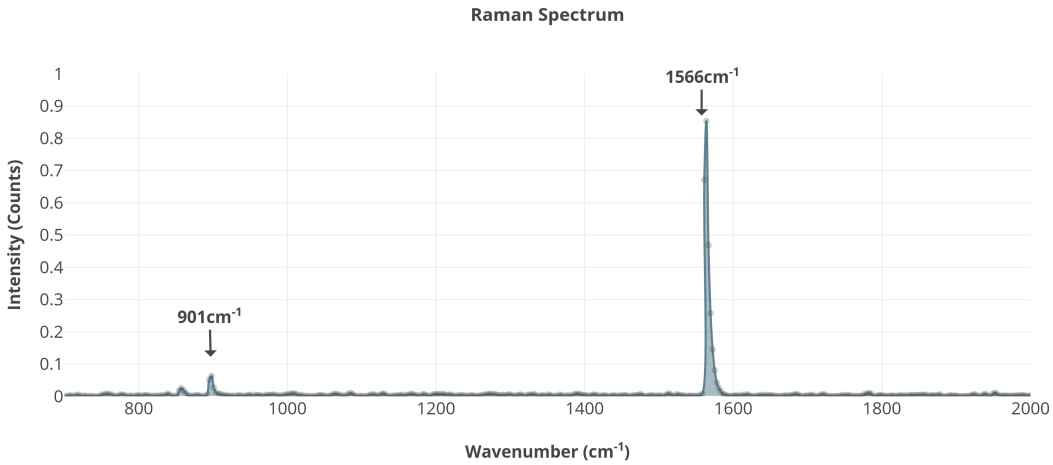
5.2. Validación Experimental del Instrumento

En las siguientes secciones, se mostrara el proceso de validación experimental del instrumento, con una serie de sustancias, con algún tipo de interés científico y/o industrial, con las cuales se buscara tener un contraste entre las imágenes obtenidas y el espectro respectivo de dicha sustancia; todas las especies químicas con las que el instrumento se puso a prueba, son aquellas ya señaladas en el capitulo 4. Los datos espectrales, se pueden ver en el siguiente link `https://plot.ly/~jevelez505#/`

5.2.1. Grafito y Peróxido de Hidrógeno

En la literatura científica, uno de los picos más característicos es el asociado con la banda G de algunos materiales de carbono como el grafito, el carbón activado, etc. [79], también llamado E_{2g}, modo vibracional que está relacionado con la distribución de esfuerzos a lo largo de la muestra [80].
Por lo general esta banda se encuentra alrededor de la región de los $1580cm^{-1}$ [81]. En la figura **5-4** a) utilizando la configuración mostrada en **5-1** fue posible detectar la señal de la banda G a $1566cm^{-1}$ que está bastante cerca de Los resultados encontrados en la literatura, la diferencia podría deberse a la resolución de nuestro espectrómetro que no fue diseñado para proporcionar espectros Raman.
los puntos rojos en la imagen podrían estar asociados con la banda 2D de este tipo de sustancias [82], sin embargo esta se encuentra por encima de la ventana de medición del instrumento, la banda que se observa alrededor de los $901cm^{-1}$ puede deberse a la banda D del grafito, sin embargo es mas común encontrarla alrededor de los $1200cm^{-1}$, por lo cual no hay certeza respecto de la naturaleza de esta señal.

Así mismo, se hicieron pruebas similares para otras sustancias como H_2O_2, SiC, *Myo*-iInositol, ácido oxalico, ácido palmítico, leucina y agua, para todos ellos los espectros y Las fotos se tomaron al mismo tiempo para establecer las contribuciones de cada banda espectral en la imagen. En el caso del H_2O_2 y de acuerdo con el rango de números de onda que abarca el espectrómetro $(278.47\text{-}2667)cm^{-1}$, los únicos picos que el instrumento podría detectar para esta molécula son $878cm^{-1}$ y $1552cm^{-1}$ [83], [84], [85]; el primero está relacionado con el estiramiento O-O y el segundo es el modo de doblamiento del O-H, el espectro y la imagen se muestran en **5-6**
En la figura **5-6** b), el pico más intenso está centrado en $847cm^{-1}$, que está cerca de los $875cm^{-1}$ reportados para el peróxido de hidrógeno en las publicaciones mencionadas anteriormente, de igual manera ocurre para el pico a $1528cm^{-1}$; la diferencia entre estos resultados se debe posiblemente a la resolución del espectrómetro, que no fue diseñado originalmente para los propósitos de este trabajo, sin embargo, y debido a la naturaleza de la sustancia estudiada; no hay ningún otro fenómeno inelástico relevante reportado para el peróxido de



(a) Espectro Raman del Grafito

Pixel Number	wavenumber	averaged_spectrum
1392	882,3580871	0,0024375
1393	885,471367	0,00025
1394	888,5832716	0,001875
1395	891,6938016	0,0053125
1396	894,8029579	0,05175
1397	897,9107414	0,063375
1398	901,0171529	0,0275
1399	904,1221933	0,012125
1400	907,2258634	0,0086875
1401	910,328164	0,0069375
1402	913,4290961	0,003375
1403	916,5286603	0,005

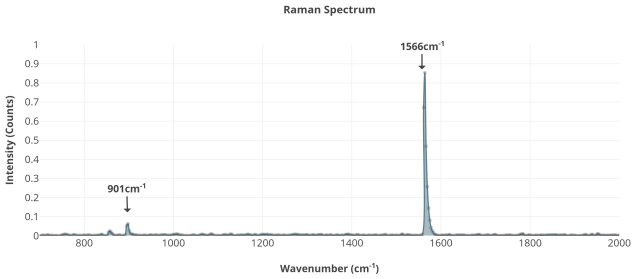
(b) Datos Banda ($901cm^{-1}$) Raman

Pixel Number	wavenumber	averaged_spectrum
1618	1552,529683	0,0014375
1619	1555,352431	0,0036875
1620	1558,173979	0,0079375
1621	1560,994327	0,671625
1622	1563,813477	0,853625
1623	1566,631428	0,4685625
1624	1569,448182	0,2578125
1625	1572,263739	0,1456875
1626	1575,078099	0,081375
1627	1577,891265	0,0428125
1628	1580,703236	0,026125
1629	1583,514012	0,015625

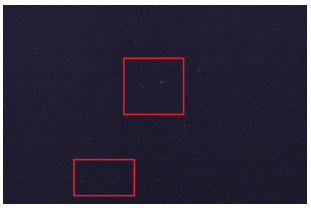
(c) Datos Banda ($1566cm^{-1}$) Raman

Figura 5-4: Espectro del Grafito y Datos del Espectrómetro

(a) Imagen Espectral del Grafito

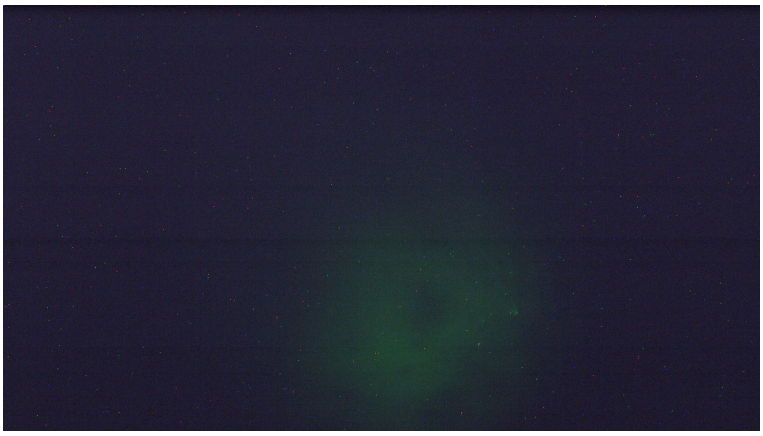


(b) Espectro Raman del Grafito

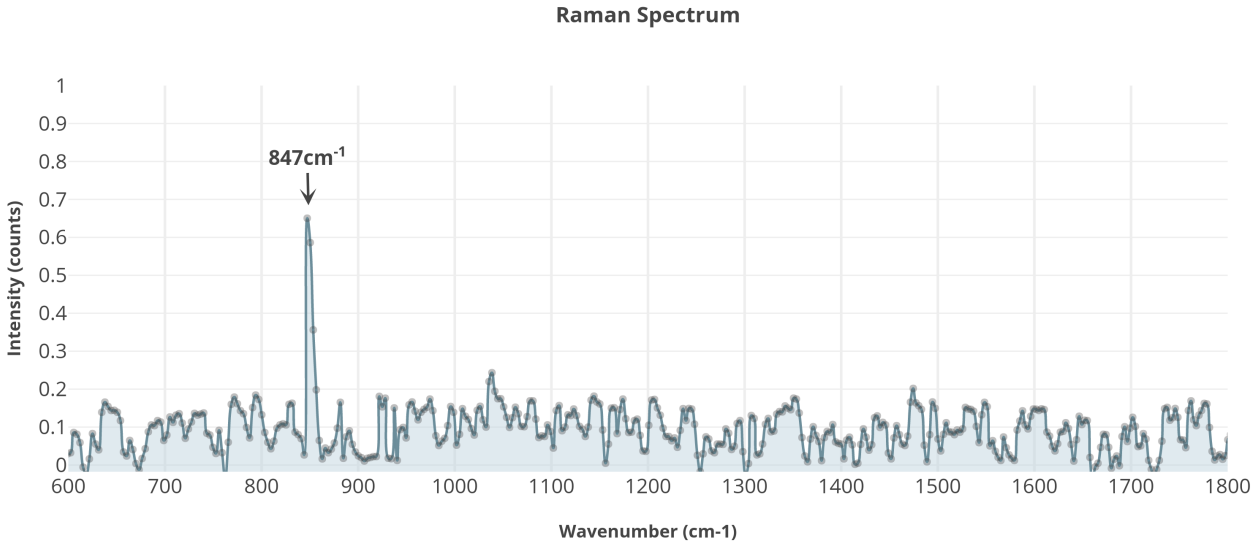


(c) Imagen Espectral con El filtro de 578nm

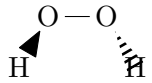
Figura 5-5: Comparación entre una Imagen con y sin Filtro del Grafito



(a) Imagen Espectral de H_2O_2



(b) Espectro Raman de H_2O_2



(c) Peróxido de Hidrógeno

Figura 5-6: Comparación entre la imagen y el espectro Raman del peróxido de Hidrógeno con el filtro pasa banda de 557nm, FWHM=3nm

hidrógeno, por lo cual es posible concluir que los picos son efectivamente el estiramiento νO-O y el doblamiento del grupo O-H, incluso si esto no fuese evidencia suficiente, la imagen **5-6** c) tomada con la cámara acoplada al filtro pasa banda (CW = 557.8nm), muestra claramente la forma de la gota de H_2O_2, y dada la longitud de onda del láser y la longitud de onda central del filtro; El desplazamiento Raman se puede calcular con [44].

$$\Delta\nu = (\frac{1}{\lambda_0} - \frac{1}{\lambda_i}) \times 10^7 cm^{-1} \tag{5-1}$$

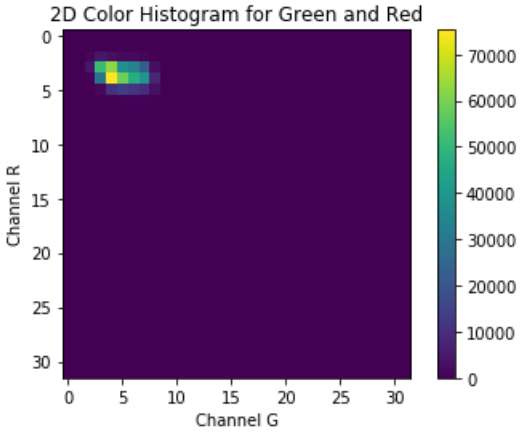
En este caso $\Delta\nu = 869{,}4cm^{-1}$, pero si tenemos en cuenta el FWHM = 5nm, entonces el rango de detección para este filtro es $[788.7\text{-}949.4]cm^{-1}$ que cubre la región teórica donde se encuentra el modo de estiramiento O-O. Por esas razones, el sistema puede detectar picos Raman en el rango mencionado anteriormente, otro tema importante que debe mencionarse es la diferencia de sensibilidad entre el espectrómetro y la cámara; A veces, con el espectrómetro no es posible obtener ninguna señal, sin embargo, la cámara si es sensible a este tipo de emisiones inelásticas, probando ser un excelente medio de medición de este tipo de interacciones luz-materia. Por otro lado, el análisis de la imagen proporciona más información sobre las longitudes de onda involucradas en el proceso, para hacer eso se creó un algoritmo para procesar y analizar los datos de las imágenes, en la figura **5-7** a) el histograma 2D muestra una comparación entre los canales rojo y verde, los que son útiles para evaluar la señal Stokes de la muestra, en esta figura es claro que existe una tendencia hacia el canal verde que es lógica debido a la presencia del filtro, sin embargo, existe Algunas señales en el canal rojo, esto no es una situación extraña, porque los filtros Bayer de la cámara no están diseñados para rechazar completamente un color específico; lo hacen gradualmente para simular la visión humana sin saltos entre canales superponiéndose entre sí por la misma razón, por ejemplo, el color amarillo en una cámara es generado por una superposición de los canales rojo y verde [86]. En **5-7** b) se muestra el proceso de enmascaramiento del algoritmo y la distribución de las intensidades en el lugar puntual.

.

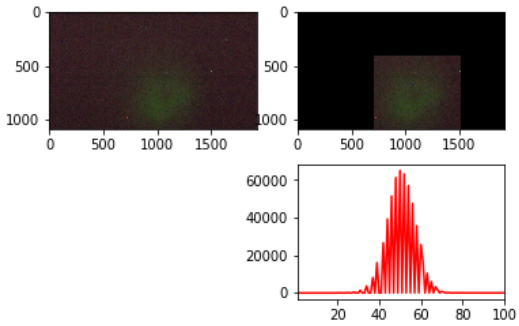
5.2.2. Myo-Inositol

El mismo análisis se realizó para otros compuestos; el siguiente caso involucra un análisis de imagen con dos filtros diferentes (557.8 y 578.5)nm para *Myo*-inositol, la idea era reproducir resultados similares como los obtenidos por (Ganzhorn et al.,(1993)) [87] para confirmar la confiabilidad del sistema con una molécula más compleja.

Este compuesto es muy importante en la transducción de señales celulares y participa en la osmorregulación; se encuentra en muchos tejidos, pero en mayor proporción en el cerebro ya que forma parte del proceso de síntesis de neurotransmisores que se produce dentro de los terminales presinápticos. [88], [89]. La figura **5-9** muestra la respuesta espectral de los



(a) Histograma 2D de intensidades entre los canales rojo y verde para H_2O_2



(b) Región de enmascaramiento de la imagen del H_2O_2

Figura 5-7: Comparación entre un histograma 2D para el H_2O_2 y su respectivo histograma de intensidad para la región enmascarada

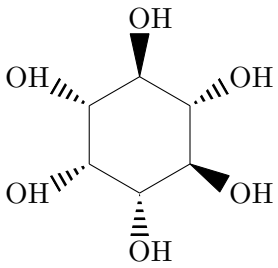


Figura 5-8: Molécula de *Myo*-inositol

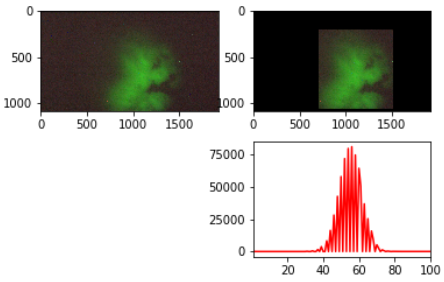
cristales de *Myo* -inositol y las imágenes correspondientes con los dos filtros.

Como se indicó, la figura **5-9** e), reprodujo los resultados de la región Amida I obtenidos por (Ganzhorn et al., 1992), donde se muestra que los dos picos más intensos de esta región están a $1652cm^{-1}$ y $1669cm^{-1}$ respectivamente, mientras que los obtenidos con nuestra configuración fueron $1657cm^{-1}$ y $1668cm^{-1}$, lo cual está en un rango razonable de cercanía para un equipo de bajo costo; para el mismo experimento estos autores, utilizaron un espectrómetro Raman (Instruments S.A, Mole S 3000) con un láser de ion de argón a 514.5nm (Spectra Physics, Stabilite 2016-05S), una configuración más costosa y estándar. Las imágenes, indican la capacidad de la cámara para obtener señales de dispersión inelásticas; ambos histogramas 2D muestran diferentes comportamientos, el histograma del filtro de 578nm enseña una distribución homogénea de los dos canales; esto se espera para un color amarillo (570-589)nm; Al realizar la conversión de unidades del desplazamiento Raman a longitudes de onda (nm), el valor obtenido para $1652cm^{-1}$ es 583.25nm, que se encuentra en la región amarilla del espectro visible. Esto esta en conformidad con los datos espectrales obtenidos y da una conclusión muy interesante sobre la utilidad de los histogramas 2D. Aparentemente, es posible recuperar la información espectral de una muestra con solo hacer una división de canales RGB con imágenes tomadas por una cámara a color.
En el caso del histograma de 557nm, el sesgo hacia el canal verde es bastante grande, esto significa un rango dinámico más alto en comparación con el canal rojo [90], esta diferencia implica que las señales en el rango del filtro Bayer verde son Más fuertes que los del canal rojo. El filtro usado tiene un FWHM = 5 nm, que en términos de desplazamiento Raman para una longitud de onda de excitación de 532 nm el rango de cobertura sería $[778.97\text{-}907]cm^{-1}$. Si se examina la respuesta espectral de la muestra en este rango, **5-10**.

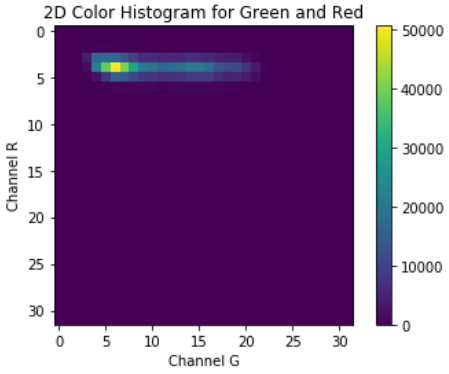
Entonces, una conclusión razonable es que los dos picos en [850-890] cm^{-1} son la mayor contribución al espectro total y ambas longitudes de onda están en el rango del filtro de 557nm como se mencionó anteriormente, por lo que es lógico que el rango dinámico muestre esa tendencia, por lo cual, nuevamente el histograma 2D proporciona alguna evidencia sobre la posibilidad de resolver un espectro simplemente dividiendo los canales RGB. Una vez más la imagen y el espectro son consistentes.

5.2.3. Carburo de Silicio, Agua y algunos Compuestos Orgánicos

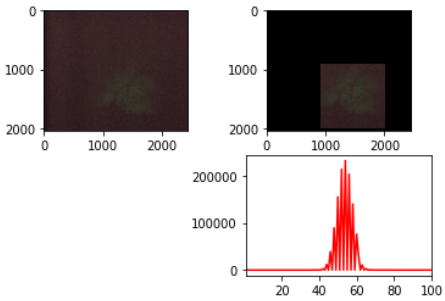
El siguiente análisis se realizó sobre diferentes especies químicas, algunas de ellas fuera de la ventana de detección del espectrómetro, la idea es mostrar las diferencias entre la forma tradicional de la espectroscopia Raman en comparación con una metodología más moderna basada solo en el análisis de imágenes, enseñando las limitaciones de la anterior. El carburo de silicio es un compuesto muy importante para muchas industrias diferentes como, cerámica, semiconductores, defensa, etc. Como dato curioso, una de las primeras aplicaciones de este



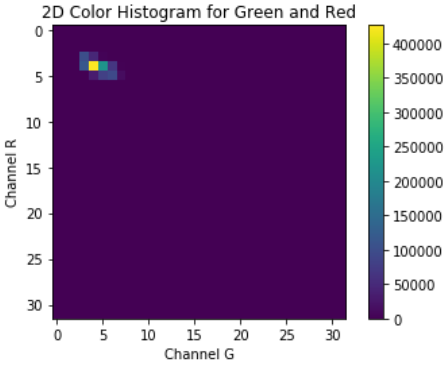
(a) Imagen Espectral de *Myo*-Inositol con el Filtro de 557.8nm



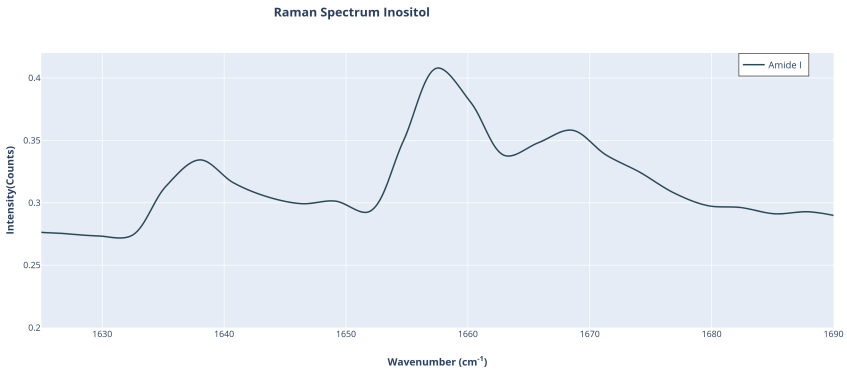
(b) Histograma 2D de *Myo*-inositol para los canales rojo y verde con el filtro de 557nm



(c) Imagen Espectral de*Myo*-inositol con el filtro de 578nm



(d) Histograma 2D de *Myo*-inositol para los canales rojo y verde con el filtro de 578nm



(e) Espectro Raman de *Myo*-inositol

Figura 5-9: Imagenes Espectrales e Histogramas 2D de *Myo*-Inositol

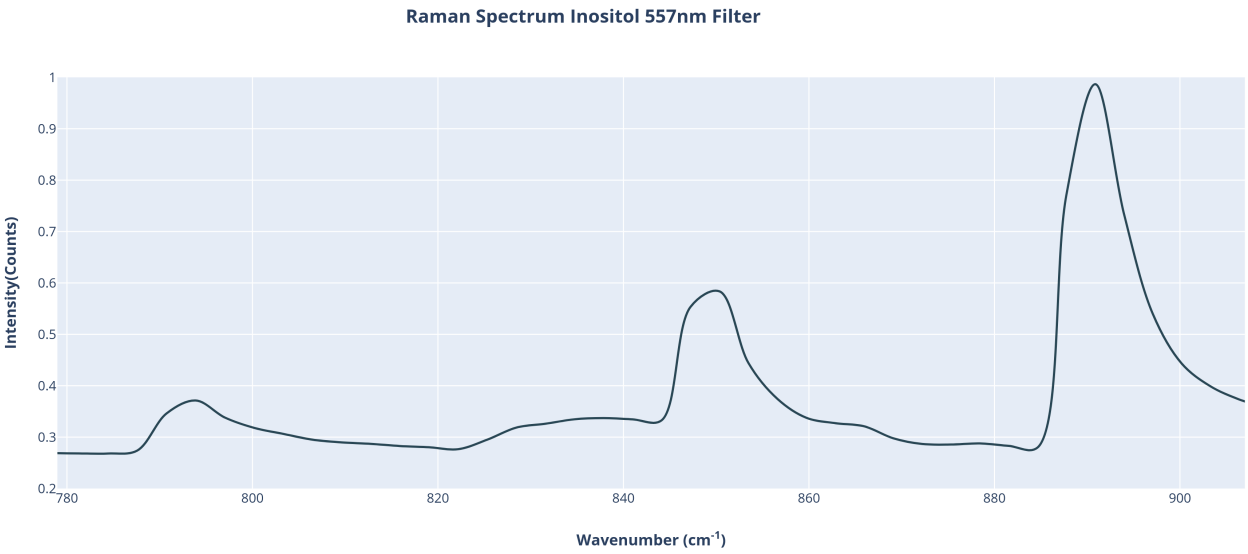


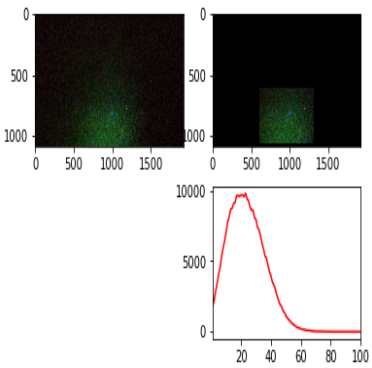
Figura 5-10: Espectro Raman de *Myo*-Inositol en la Región [778-907]cm^{-1}

material fue como el primer LED de la historia, a principios del s.XX Henry Joseph Round encontró una emisión de luz amarilla cuando paso una corriente eléctrica a través de un detector de carburo de silicio [91]; por lo tanto, la idea de esta sección estudiar es demostrar usos potenciales en la industria del instrumento.

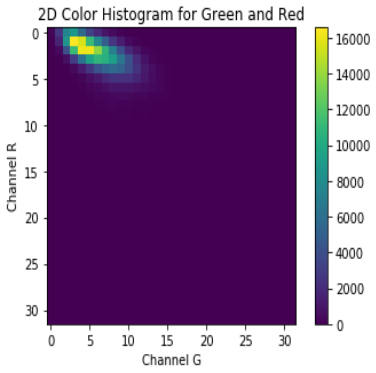
En el artículo de Okumura et al. (1987), reportaron dos picos para diferentes polimorfos de carburo de silicio, 3C y 6H, ambos con un modo muy fuerte TO (óptica transversal) y modo LO (óptica longitudinal) a 796cm^{-1}, 972cm^{-1} y 789cm^{-1}, 967cm^{-1} respectivamente. Si estos resultados se compararan con el espectro mostrado en **5-11**, no está claro qué tipo de polimorfo es el analizado, esto podría complicarse aún más al revisar el trabajo de Liu et al. (2016) [12], porque encontraron que para el 4H-SiC hay una banda a 797.5cm^{-1}. Sin embargo, [92] Chikvaidze y colaboradores encontraron para los cristales grises de SiC (que es el caso) un pico a 892cm^{-1} muy cercano al obtenido en este trabajo 891cm^{-1} demostrando nuevamente la confiabilidad del instrumento incluso para aplicaciones industriales o comerciales.

El análisis de la imagen muestra que hay algo de rojo, esto podría deberse a la presencia de agua en la muestra, lo cual no es una sorpresa, ya que el carburo de silicio es una sustancia bastante higroscópica [93], [94], la siguiente imagen muestra algunos puntos rojos, revelando probablemente la presencia de agua (el rango del espectrómetro no cubre esta región [2600-3500]cm^{-1}). **5-12**.

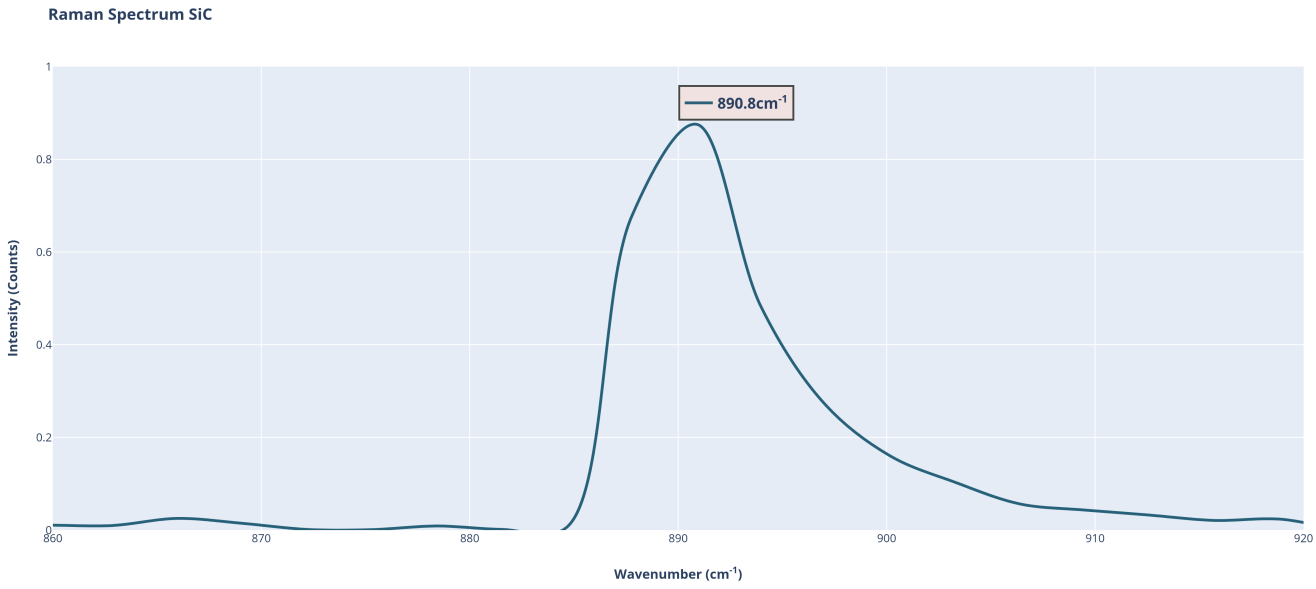
Esta imagen proporcionó cierta evidencia sobre las ventajas de los CCD bidimensionales sobre los lineales, algunos detalles son más fáciles de notar cuando la información proviene de dimensiones adicionales, lo que reduce el esfuerzo en el análisis. Esta idea se explorará en detalle en las siguientes secciones.



(a) SiC Imagen y Distribución de Intensidades



(b) SiC Histograma 2D (Canales Verde y Rojo) sin Filtro



(c) Espectro Raman del SiC

Figura 5-11: Imagen y Análisis Espectral del Carburo de Silicio (SiC)

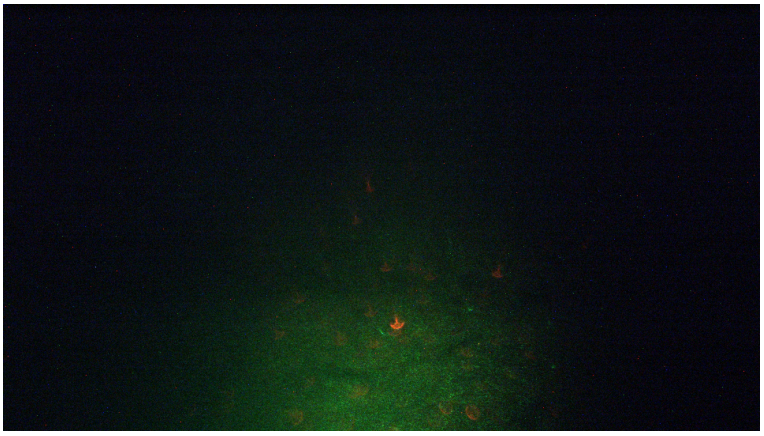


Figura 5-12: Imagen del Carburo de Silicio mostrando unos puntos rojos (Agua)

5.2.4. Ácido Palmítico

Este ácido graso saturado está formado por una cadena carbonada de 16-C y recibe este nombre porque se encuentra en grandes cantidades en el fruto de palma del género *Elaeis* [95] y se usa ampliamente en las industrias de alimentos y cosmética [96].

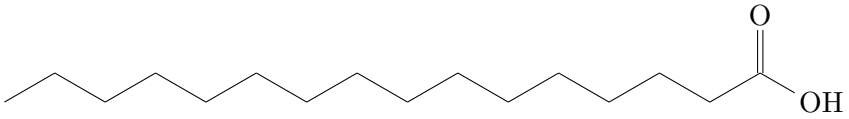
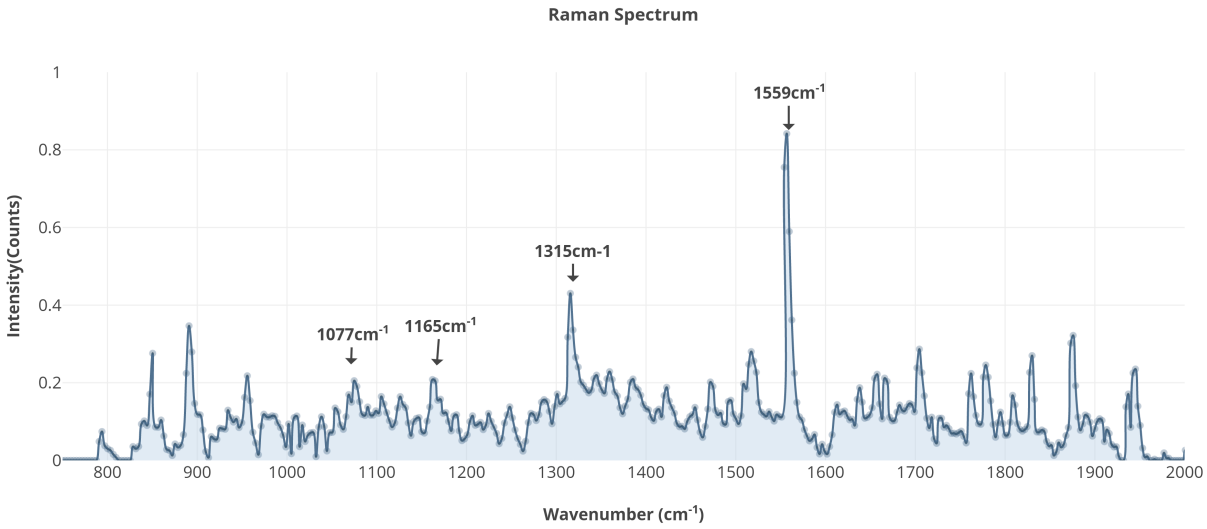
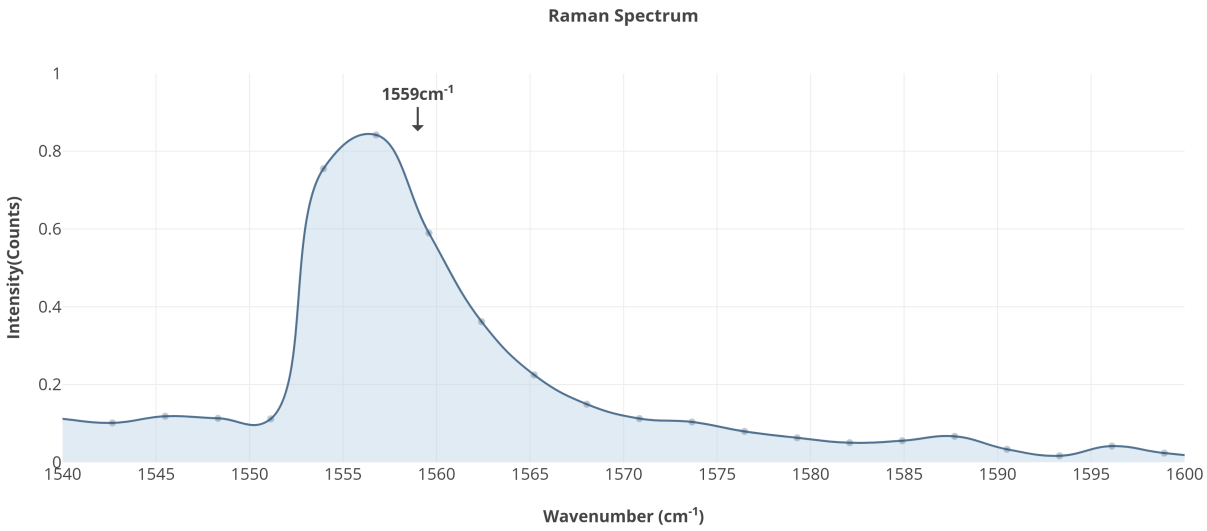


Figura 5-13: Ácido Palmítico (16:0)

La siguiente figura (**5-14**) muestra el espectro Raman en tres diferentes regiones de interés, las más comunes para este tipo de moléculas. Los correspondientes números de onda y tipos de vibración son [1060-1180]cm^{-1} para ν (C-C) de estiramiento, 1300 cm^{-1}, δ (CH_2) vibración de twisting y [1400-1500] cm^{-1}, correspondientes a las deformaciones δ (CH_3) o $\delta(CH_2)$ [97] , [98] .
La imagen de los cristales de ácido palmítico se realizó solo con el filtro EDGE debido a los resultados del espectro, que muestra claramente que los picos más prominentes estaban en δ CH_2, δ CH_3 vibraciones de deformación y δ CH_2 vibración de torsión. En términos de longitudes de onda, corresponde a 579nm y 573nm respectivamente, pero el filtro más cercano a esta región tiene un rango de [573.5-583] nm que rechaza parte de la luz del modo de vibración de torsión. En **5-16** el histograma 2D muestra un patrón casi circular y, nuevamente, un punto caliente en la intersección de los canales rojo y verde, esta característica confirma la presencia no solo de los picos amarillos sino también de la presencia de radiación. De la parte verde del espectro visible. Para esta molécula, las vibraciones asociadas a ν (C-C) (1060-1180)cm^{-1} estiran la vibración y probablemente debido a la oscilación (898-912)cm^{-1} vibración (no visible con nuestro espectrómetro) produce un cambio Raman en la región entre [558-567]nm que explica la pequeña cola hacia el canal verde. Esta situación

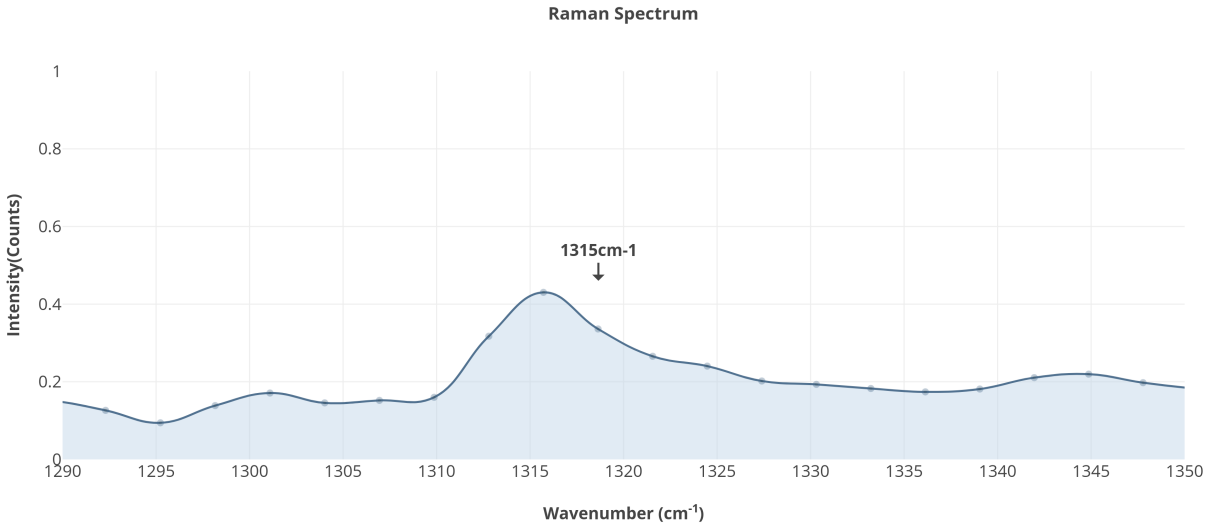


(a) Espectro Completo del Ácido Palmítico

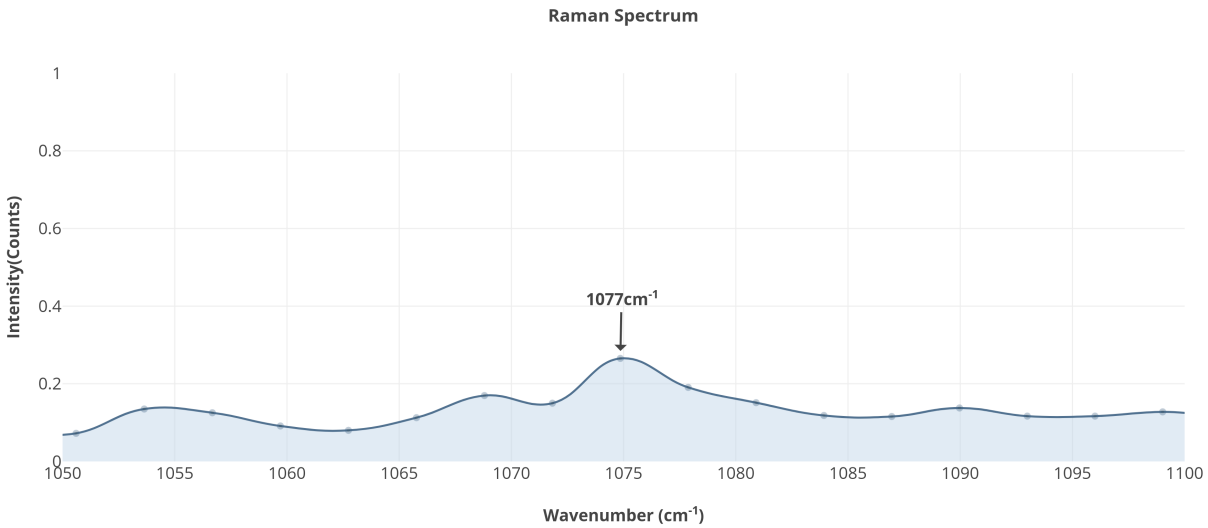


(b) Deformaciones $\delta\ CH_2$ o $\delta\ CH_3$

Figura 5-14: Espectro Raman del Ácido Palmítico para diferentes tipos de Vibraciones 1



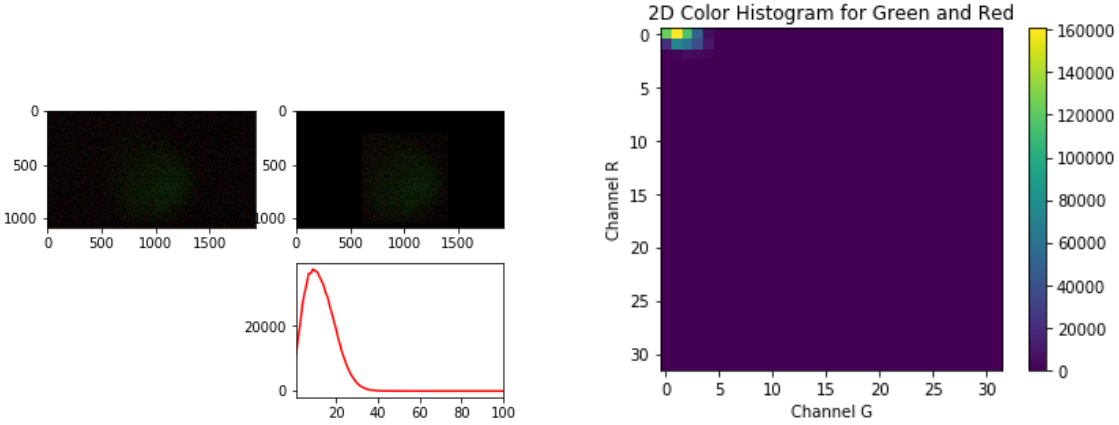
(a) $\delta\ CH_2$ Vibración Twisting



(b) ν (C-C) Vibración de Estiramiento

Figura 5-15: Espectro Raman del Ácido Palmítico para diferentes tipos de Vibraciones 3

proporciona evidencia sobre las ventajas de usar un sensor 2D CCD sobre los lineales de los espectrómetros y es solo una cuestión de números, porque en el caso del espectrómetro B&W Tech^{TM} usado en este trabajo, el CCD lineal tiene un 2048 Arreglo de detectores Mientras que en los sensores de la cámara, el número de detectores podría ser (2048x2048) para una cámara de 4MP (megapíxeles). En este trabajo, se utilizó un sensor de 5MP que supera enormemente la resolución del espectrómetro.



(a) Imagen Espectral de Cristales de Ácido Palmítico

(b) Histogramas 2D Ácido Palmítico

Figura 5-16: Imágenes de Ácido Palmítico Sobre los 540nm

5.2.5. Ácido Oxálico

El ácido oxálico es un sólido cristalino incoloro que se encuentra en muchas plantas y se biosintetiza en el ciclo de Krebs cuando el oxaloacetato se hidroliza a oxalato, este tipo de moléculas son interesantes para mostrar la versatilidad del equipo en diversos campos de investigación. [99] **5-17**.

Nuevamente, el procedimiento comienza con la evaluación del espectro Raman a partir de

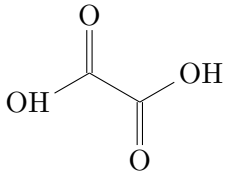


Figura 5-17: Ácido Oxálico

una base de datos de espectros confiable, en este caso el AIST (Instituto Nacional de Ciencia y Tecnología Industrial Avanzada) de japón [100] en el que se encontraron los picos más fuertes en [859, 1496, 1659 y 1751]cm^{-1} o [557.48-586.55]nm, esto significa que es posible realizar un análisis con los dos filtros pasa banda disponibles.
En la imagen **5-19**, los únicos picos distinguibles se encontraron en 887.7cm^{-1} y 1556.7cm^{-1}, el primero podría deberse al estiramiento (ν C-C) [101], [102], este último es probablemente la vibración de estiramiento de (ν -C=O) que generalmente responde en el rango de [1670-1600]cm^{-1}, muy cerca del pico resaltado en la imagen y si se compara con el espectro AIST está muy cerca de la banda de 1659cm^{-1}, por esta razón es plausible concluir que la señal mostrada corresponde al modo Raman mencionado anteriormente. En las imágenes de ambos filtros **5-19** b), d); el polvo se puede distinguir con una intensidad mayor para el filtro de 557 nm que está de acuerdo con el espectro tomado, en este caso, el pico de intensidad del histograma supera los 40000 bins, mientras que la imagen con el filtro de 578 nm está por debajo de este número. La interpretación de los histogramas 2D refuerza el hecho de obtener información espectral del proceso de separación de los canales RGB, en este caso el histograma 2D de la imagen del filtro de 578nm muestra el mismo patrón que en los otros casos cuando hay principalmente bandas alrededor de la región amarilla del espectro visible (distribución homogénea alrededor de un punto caliente), la otra muestra una tendencia hacia el verde sobre un área más amplia, lo que significa una distribución más homogénea en la imagen, en otras palabras, el sensor es capaz de detectar radiación verde de regiones diferentes del espectro.

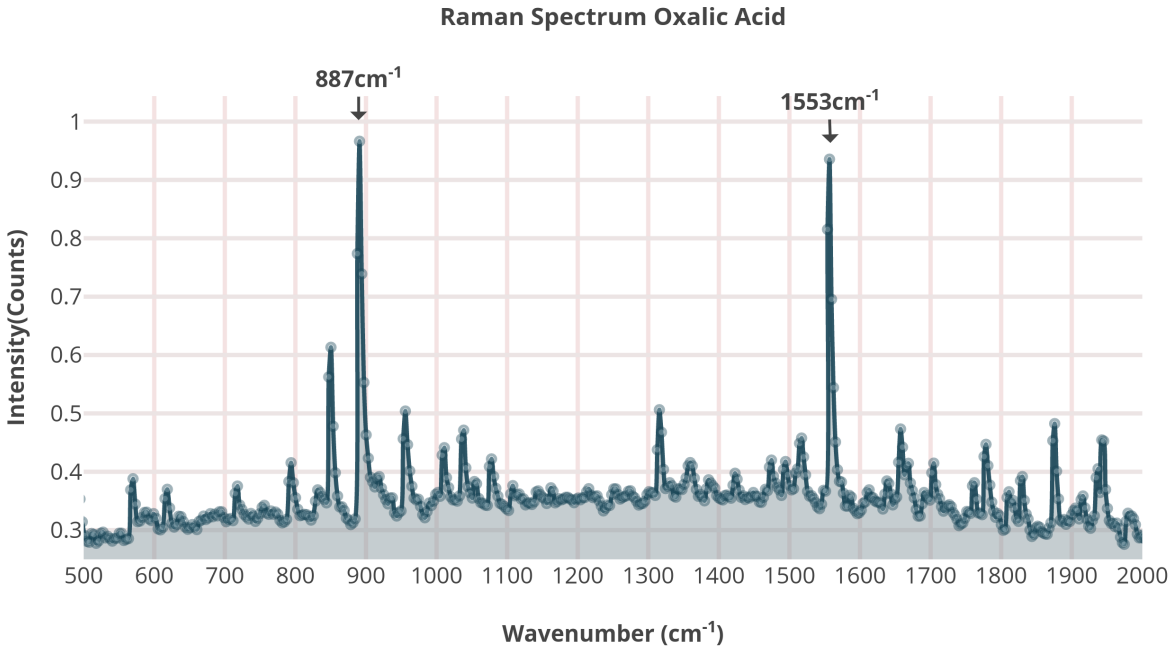
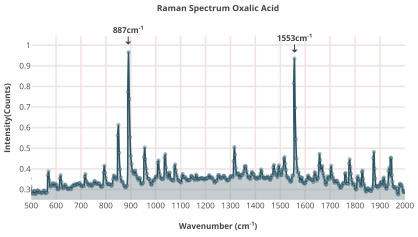
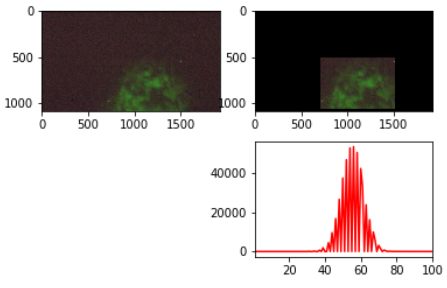


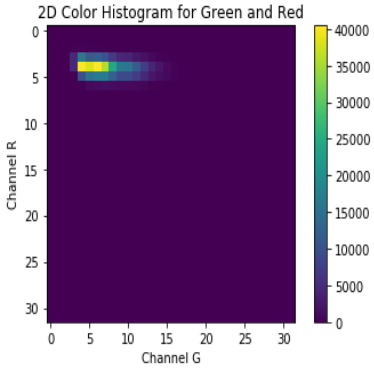
Figura 5-18: Espectro del Ácido Oxálico



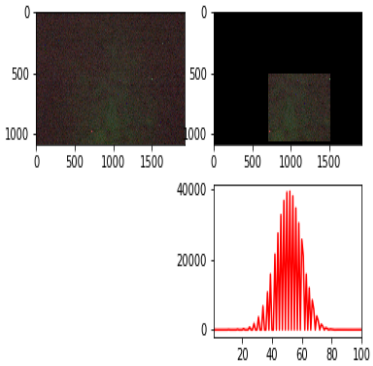
(a) Espectro del Ácido Oxálico



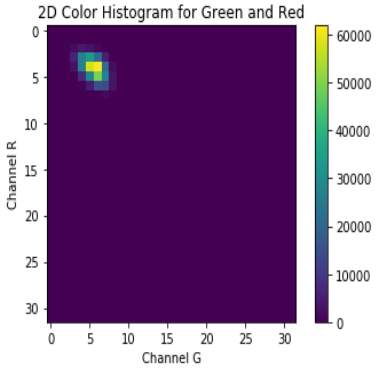
(b) Imagen del Ácido Oxálico en Polvo para el filtro de 557.8nm .png



(c) Histograma 2D para con el filtro 557nm



(d) Imagen con el filtro de 578.5nm .png



(e) Histograma 2D para el Ácido Oxálico con el filtro de 578.5nm

Figura 5-19: Imágenes y Espectro del Ácido Oxálico para Diferentes Filtros Pasa Banda

5.2.6. L-Leucina

Este aminoácido esencial (no se puede sintetizar) para humanos está involucrado en la producción de acetil-CoA, una molécula muy importante en el metabolismo de carbohidratos, proteínas y lípidos y acetoacetato, un ácido β ceto producido en la mitocondria del hígado en condiciones de ayuno (uno de los tres cuerpos cetónicos) [103], [104]. Figura **5-20**. El estudio de este tipo de moléculas (más cerca del objetivo de este trabajo) prueba la relevancia de este dispositivo como herramienta para la investigación bioquímica.

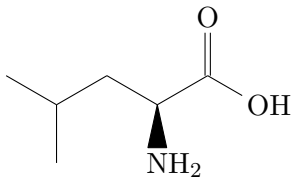


Figura 5-20: L-Leucine

El espectro y las imágenes asociadas se muestran en la figura **5-22**.

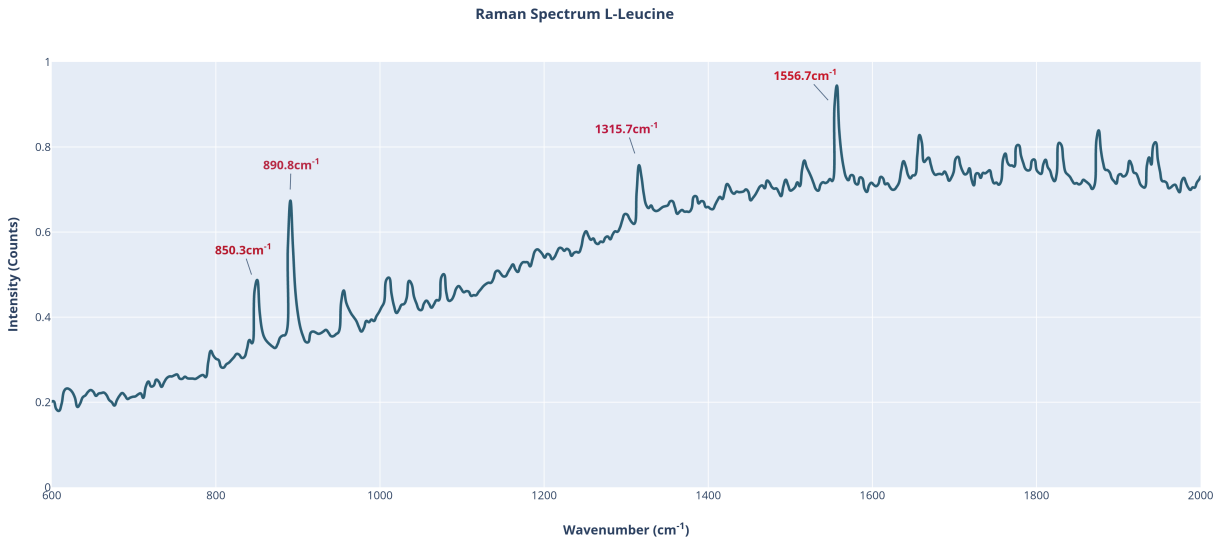
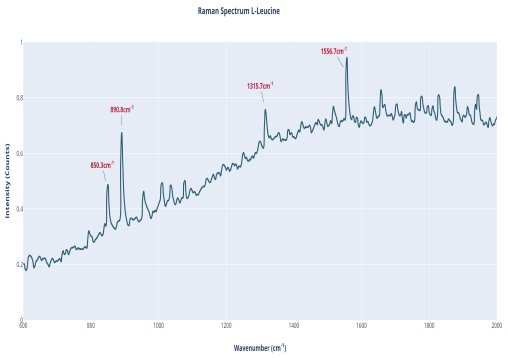
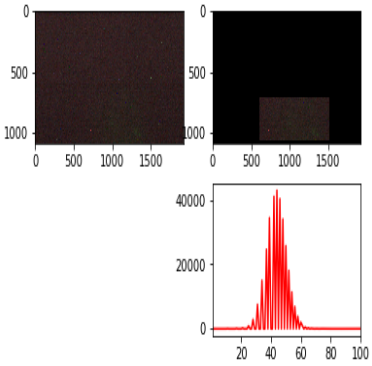


Figura 5-21: Espectro de la L-Leucina

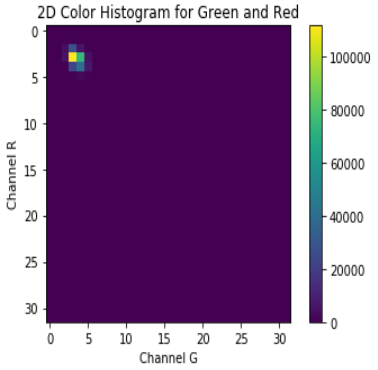
Según [105] y [106], las bandas resaltadas en la figura como $850cm^{-1}$ y $890.8cm^{-1}$ **5-22** a), corresponde a la vibración de oscilación $\Gamma(CH_3)$, junto con el modo de estiramiento (C-C). Existe una banda característica debido a la deformación -CH del grupo isopropilo que se espera que ocurra a $1355cm^{-1}$, en este caso la señal más probable correspondiente a esta



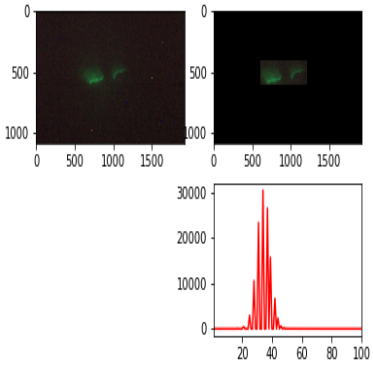
(a) Espectro de la Leucina



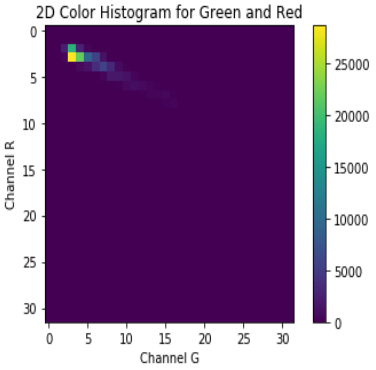
(b) Imagen de Cristales de Leucina con Filtro de 557.8nm



(c) Histograma 2D de la L-Leucina para filtro de 557.8nm



(d) Cristales de Leucina con Filtro de 578nm

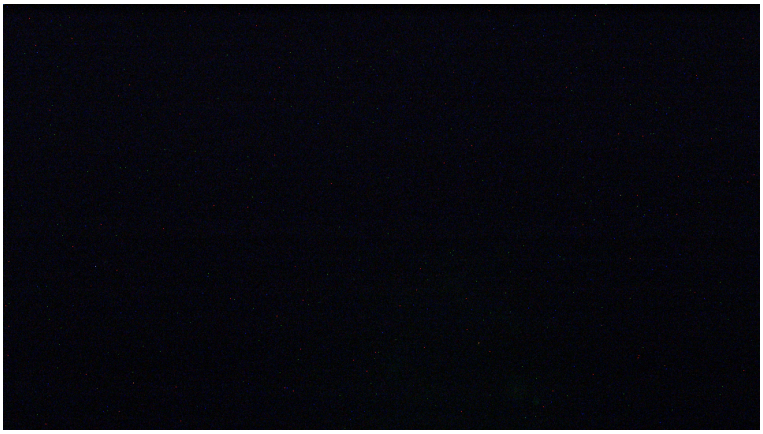


(e) Histograma 2D con filtro de 578nm

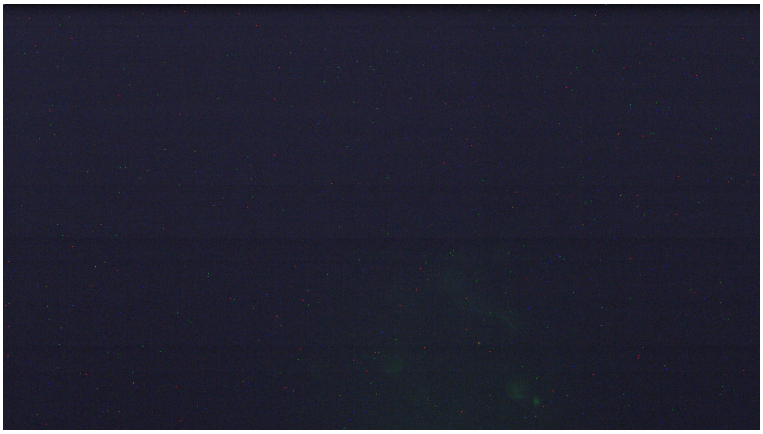
Figura 5-22: Imágenes de L-Leucina con Diferentes Filtros Pasa Banda

banda es la que se obtiene alrededor de $1315cm^{-1}$. La banda a 1556 cm^{-1} podría asociarse con la flexión simétrica (-NH_3^+) que generalmente se encuentra en $1581cm^{-1}$, a pesar del hecho que hay otra banda teórica en la misma zona (1500 cm^{-1}) correspondiente al estiramiento asimétrico (C-N), se descartó en este análisis porque es una banda muy débil que probablemente no aparezca en este tipo de espectrómetro [107].
Las imágenes tienen características similares a las del caso del ácido oxálico, tal vez una mejor distribución entre los canales verde y rojo en el caso de la imagen con el filtro de 578.5nm, pero esto es algo esperado ya que, según el espectro AIST de la L-leucina [100], hay alrededor de 10 bandas diferentes en la región de cobertura del filtro, lo que explica esta distribución homogénea en ambos canales. La única diferencia con los otros experimentos es el procesamiento de la imagen con el filtro de 557.8nm, porque en condiciones normales no era posible ver ninguna característica, ya que a simple vista era solo una imagen sin información, por lo que se implementó un procesamiento de imagen para extraer la mayor cantidad de información posible simulando un mayor tiempo de exposición mediante la adición de varias imágenes y produciendo una nueva como el promedio de todas ellas, esto se realizó con la biblioteca cv2 para el procesamiento de imágenes de Python, los resultados son los siguientes, ver figura **5-23**.

Las diferencias entre las dos imágenes son notablemente evidentes, mostrando la eficacia del algoritmo para producir el resultado deseado. La secuencia de comandos de Python se creó originalmente para el único propósito de mostrar que incluso en las condiciones más extremas, una cámara con un buen sensor puede capturar señales de dispersión inelásticas. La fluorescencia inducida por láser afecta fuertemente el rango de 578nm, lo que explica las intensidades más fuertes y la nitidez de la imagen en ese caso.



(a) Imagen de Leucina con Filtro de 557nm sin Procesamiento

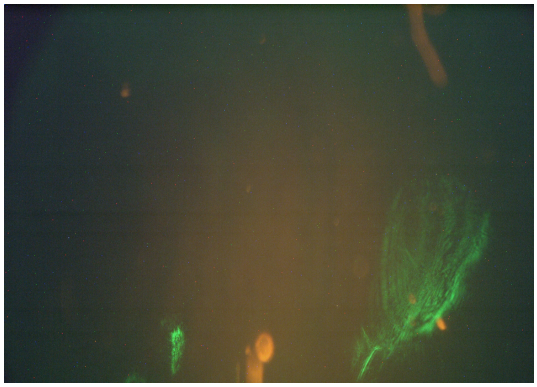


(b) Imagen Después del Procesamiento

Figura 5-23: Imágenes de Leucina con y Sin Procesamiento

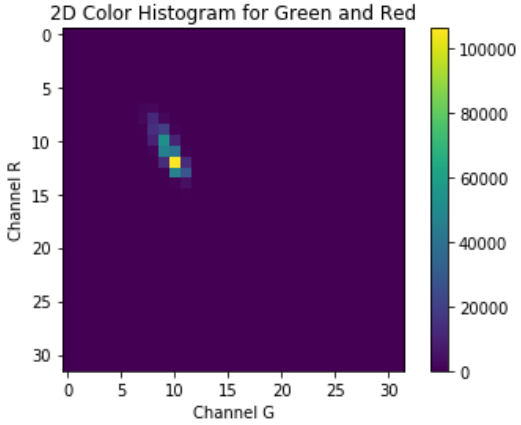
5.2.7. Agua

En este caso, solo se tomaron imágenes, ya que el espectrómetro no puede obtener señales más allá de $2500cm^{-1}$. Según Carey et al.(1998), [108] es posible medir algunas propiedades de los líquidos utilizando espectroscopia Raman, en este caso, calcularon la fuerza de la entalpía del enlace de hidrógeno. Según la literatura, se espera una banda Raman debido al estiramiento (OH) en la región de [2950-3700] cm^{-1} [109], en longitudes de onda este rango corresponde a [623.17-664.58]nm una banda bastante ancha en la región roja del espectro visible. La figura **5-24** b) muestra un evidente tono rojo proveniente de la gota de agua destilada, este color solo se explica por la dispersión inelástica de las moléculas de agua, consistente con lo que se esperaba para la banda Raman del estiramiento de OH mencionado. La figura **5-24** c), muestra una distribución sobre el canal rojo, las pequeñas desviaciones podrían deberse a que los fotones de Rayleigh no son rechazados por el filtro como se puede ver en la **5-24** a) , este fenómeno es producido por el láser que opera a plena potencia (200 mW). Sin embargo, el histograma 2D elimina cualquier duda sobre el origen y las bandas espectrales involucradas, dando lugar a la siguiente conclusión. Con el análisis de imagen apropiado, podría ser posible obtener información espectral de una muestra sin el uso de un espectrómetro al menos en el caso de la dispersión inelástica. En este caso el espectro estaba fuera del rango de detección del espectrometro, sin embargo, se aprecia que en torno a los $2645cmQ^{-1}$ comienza a crecer una banda espectral, la cual puede deberse precisamente al modo de vibración señalado (ver figura **5-25**.



(a) Gota de Agua Foto Original

(b) Imagen de la Gota de Agua Despues del Enmascaramiento



(c) Histograma 2D del Agua

Figura 5-24: Imagen Inelástica de Una Gota de Agua

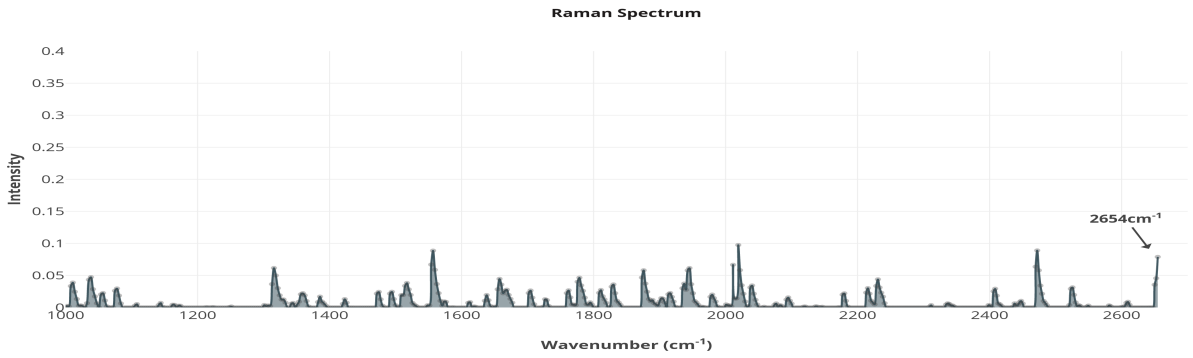


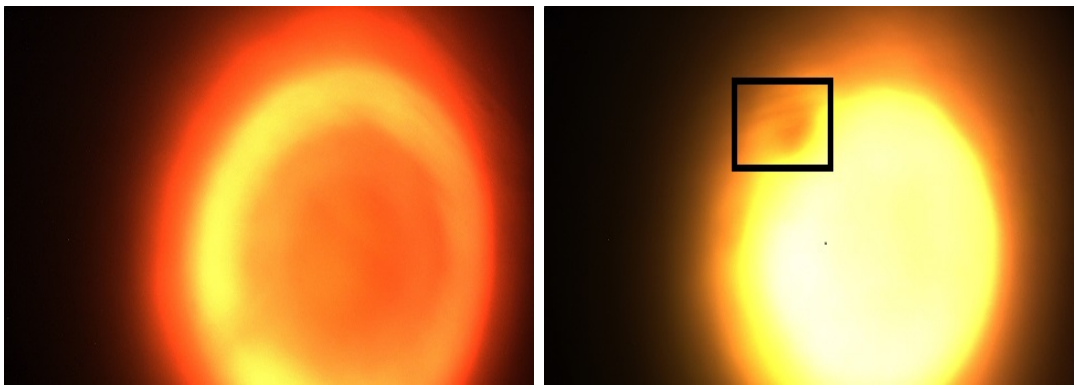
Figura 5-25: Espectro Raman del Agua

5.3. Detección de Mycosphaerella fijiensis

El procedimiento para proporcionar evidencia sobre el uso potencial del instrumento en la detección de la cepa fúngica *Mycosphaerella fijiensis* sigue la misma estructura y metodología utilizada para el proceso de calibración con diferentes compuestos químicos, con solo una diferencia crítica, el uso de Redes neuronales para establecer la presencia del patógeno. El experimento consistió en la evaluación de tres muestras diferentes y el control, todos ellos proporcionados por el laboratorio de biología molecular de la Universidad Nacional de Colombia Sede Medellín. Las dos hojas infectadas fueron marcadas como 080930 y C139, como se puede observar en la imagen. **5-26**. Las imágenes tomadas sin filtros de una muestra infectada y el control se muestran en **5-27** La mancha negra en **5-27** b) corresponde a una de las lesiones observadas en la figura **5-26** b). El láser resalta dos de esos puntos, proporcionando evidencia sobre las posibilidades de la luz láser en el diagnóstico fitosanitario. Sin embargo, la intención de este trabajo es demostrar si la dispersión inelástica podría dar información sobre la presencia de estos organismos. Nuevamente se tomó un espectro de las lesiones mostrando las siguientes características **5-28** Como se mencionó en el capítulo 3, las bandas Raman principales para quitina y $\beta 1,3-$glucano son las asociadas con el estiramiento ν -C=O de los enlaces peptídicos (Amida I) y las vibraciones de flexión β del grupo C-H respectivamente [9], [110], esas señales generalmente se encuentran entre [1655, 893] cm^{-1}. En la figura **5-28**, hay dos picos con esas características ubicadas en (1563.81-897.91)cm^{-1}, por lo que es plausible afirmar que corresponde a las vibraciones mencionadas anteriormente. En este sentido, los filtros fueron seleccionados correctamente para las imágenes espectrales. El análisis de imágenes proporcionará la evidencia definitiva sobre las ventajas de usar este dispositivo para la detección de fitopatógenos. Después del procesamiento de la imagen para aumentar el nivel de intensidad a través de una corrección de gamma, los resultados fueron los siguientes **5-29**:

(a) Muestras (b) Mancha Negra de Sigatoka

Figura 5-26: Hojas Infectadas Con *M. fijiensis* (080930 & C139) y Control



(a) Control (b) *M. fijiensis*

Figura 5-27: Imágenes Tomadas sin Filtro

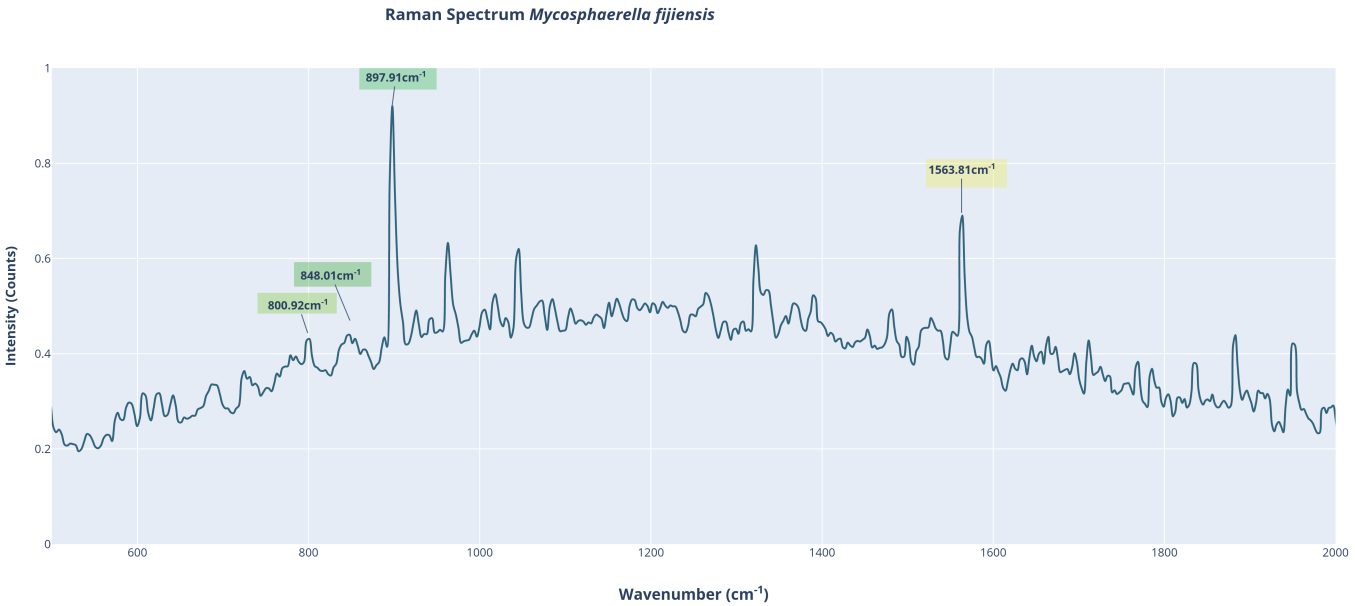
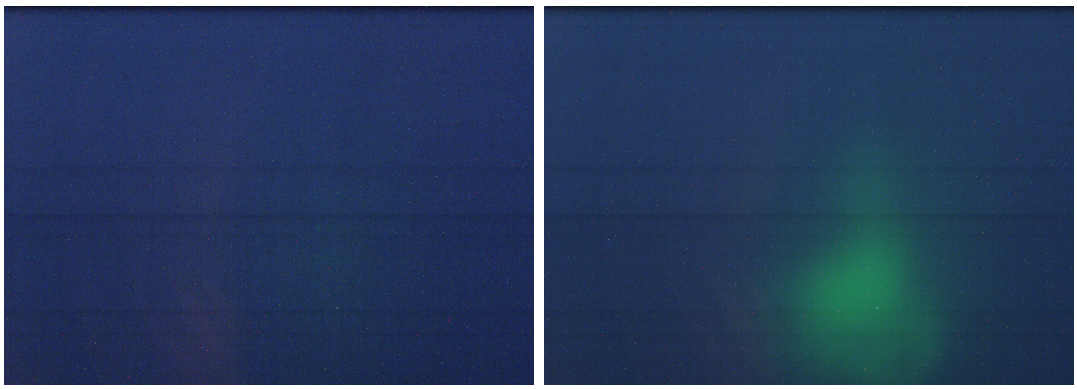
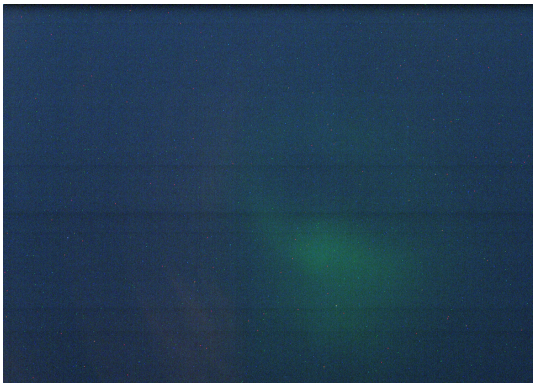


Figura 5-28: Espectro Inelástico de *M. fijiensis*

Como se dijo antes, la diferencia con respecto al control en las mismas condiciones de iluminación (532nm, FWHM = 3nm, 112.5mW) trae resultados concluyentes sobre la presencia del microorganismo en las diferentes muestras, sin embargo, un análisis de la relación entre rojo y se hicieron canales verdes, esto por supuesto con la intención de crear un protocolo para manejar todo el proceso de diagnóstico de fitopatógenos. Una vez más se hizo un histograma 2D (ver figura **5-30**)

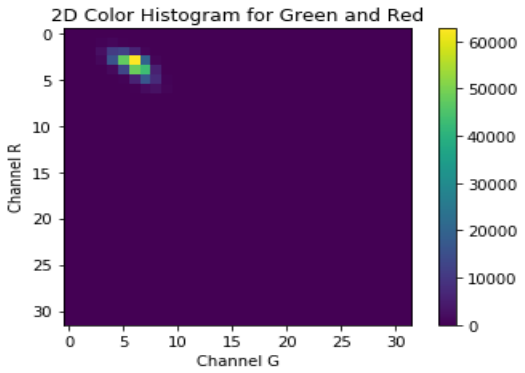


(a) Control (b) Muestra *M. fijiensis* 557nm filter 080930

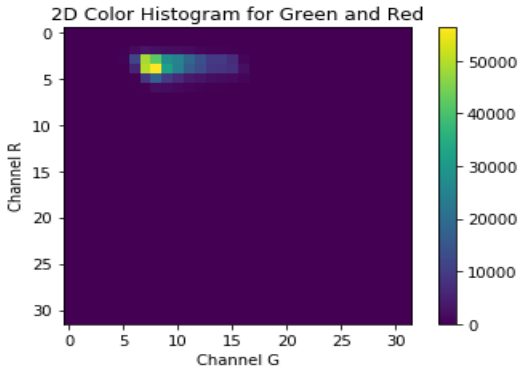


(c) Muestra *M. fijiensis* 557nm filter c139

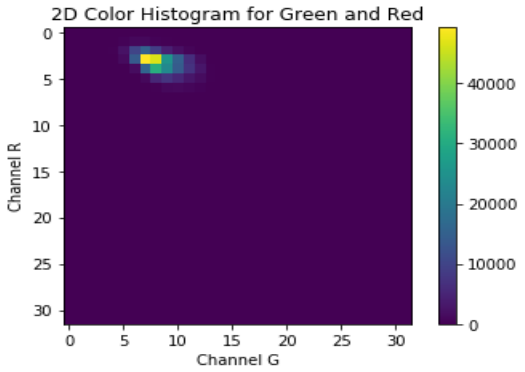
Figura 5-29: Control y dos muestras tomadas con filtro de 557nm de*M.fijiensis* y corrección gamma



(a) Control 2D



(b) Muestra con Filtro 557nm*M. fijiensis* 080930 2D



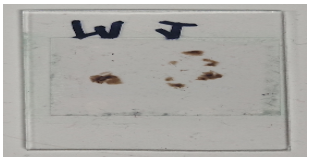
(c) Muestra con Filtro 557nm*M. fijiensis* c139

Figura 5-30: Histogramas 2D *M.fijiensis* Control y Muestras Filtro 557nm

La metodología de procesamiento de imágenes muestra diferencias significativas entre las

muestras y el control, aunque hay una señal proveniente del control, los histogramas 2D muestran un comportamiento similar en las dos muestras, una tendencia hacia el canal verde, más evidente en el caso de la muestra. 080930 pero también perceptible para C139. Con esta información, el siguiente paso lógico es probar una forma más automática de distinguir entre tejido enfermo y sano, para hacerlo se diseñó un algoritmo para diferenciarlos mediante el análisis de un conjunto de imágenes en cada condición. Para este experimento, se prepararon dos muestras, una con solo agar PDA, la otra con pequeñas colonias de *M.fijiensis* creciendo en el mismo tipo de agar, la idea es entrenar una red neuronal con un conjunto de imágenes de ambas muestras con los dos filtros y luego verificar si el sistema es capaz de detectar cuál es el agar con el fitopatógeno con una imagen no incluida en el conjunto de entrenamiento, para hacer eso se crearon cuatro etiquetas diferentes agar557, agar578, Mycos557 y Mycos578 Luego, después del proceso de entrenamiento, se ejecuta un algoritmo de clasificación que proporciona información sobre cuál de las etiquetas corresponde a la imagen de la muestra, de esa manera es posible inferir si la muestra está infectada o no. La figura **5-31** muestra las fotos tomadas con diferentes filtros.

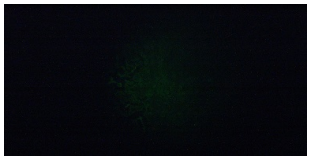
(a) Agar PDA



(b) PDA con Colonias de *M.fijiensis*

(c) Muestra de *M. fijiensis* con filtro de 557nm

(d) Muestra de *M. fijiensis* con filtro de 578nm

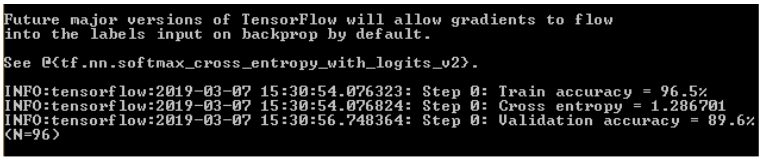


(e) PDA con Filtro de 557nm

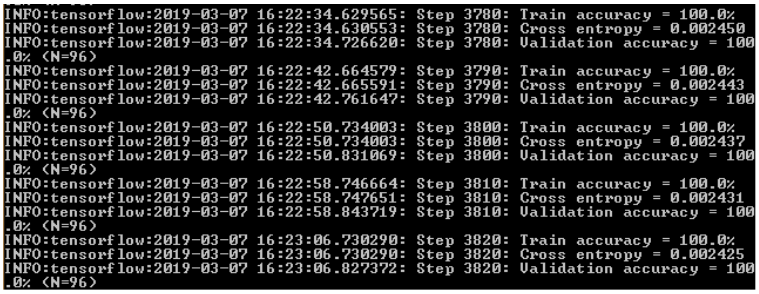
(f) PDA con Filtro de 578nm

Figura 5-31: *M.fijiensis* Imágenes de Entrenamiento con diferentes Filtros

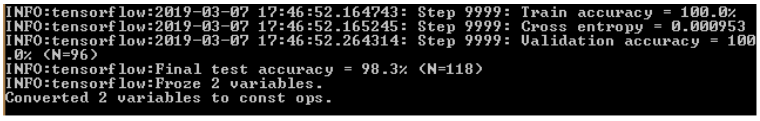
produjo el siguiente resultados después de un entrenamiento de 2h



(a) Primera Iteración del Entrenamiento



(b) Reducción de la entropía Cruzada en el intermedio del entrenamiento



(c) Iteración Final

Figura 5-32: Proceso de Entrenamiento para *M.fijiensis*

La iteración final muestra una reducción exitosa de la entropía cruzada, un parámetro que describe la precisión del modelo en la estimación de las características de las imágenes. Si la máquina está clasificando correctamente, entonces el valor de la entropía cruzada se reducirá como se muestra en la figura **5-32** a), b) y c). Otra forma de interpretar esto es que el sistema está convergiendo a la etiqueta seleccionada.

Para probar el entrenamiento, se seleccionaron tres muestras diferentes de cada una de las etiquetas (Imágenes espectrales de la muestra), ninguna de ellas se usó previamente para el entrenamiento, en las tablas **5-1**, **5-2**, **5-3**, **5-4** se mostrarán los resultados para cada muestra. El significado de las etiquetas es el siguiente: Agar557, imagen de agar tomada con filtro de 557nm sin microorganismo, Agar578 imagen de agar tomada con filtro 578 sin microorganismo, Mycos 557, imagen de agar con el microorganismo tomada con filtro de 557nm y Mycos578 imagen de agar con el microorganismo tomada con filtro de 578nm.
En todos los casos, la red neuronal pudo identificar exitosamente la presencia de *Mycosphaerella fijiensis* con una certeza superior al 98 %, que era el propósito general de este trabajo.

M.fijiensis, 557nm			
% de Detección			
Etiqueta	**Ensayo 1**	**Ensayo 2**	**Ensayo 3**
Agar 557	0.041	0.012	0.069
Agar 578	0.092	0.008	0.064
Mycos 557	99.81	99.97	99.83
Mycos 578	0.026	0.007	0.037

Tabla 5-1: Clasificación Para *M.fijiensis* Imagen Tomada con Filtro de 557nm

M.fijiensis, 578nm			
% de Detección			
Etiqueta	**Ensayo 1**	**Ensayo 2**	**Ensayo 3**
Agar 557	0.004	0.002	0.007
Agar 578	0.017	0.003	0.057
Mycos 557	0.002	0.001	0.004
Mycos 578	99.97	99.99	99.93

Tabla 5-2: Clasificación Para *M.fijiensis* Imagen Tomada con Filtro de 578nm

Agar 557nm			
% de Detección			
Etiqueta	**Ensayo 1**	**Ensayo 2**	**Ensayo 3**
Agar 557	99.89	99.98	98.89
Agar 578	0.065	0.016	0.283
Mycos 557	0.004	0.001	0.006
Mycos 578	0.033	0.005	0.823

Tabla 5-3: Clasificación Para Agar PDA Imagen Tomada con Filtro de 557nm

Agar 557nm			
% de Detección			
Etiqueta	**Ensayo 1**	**Ensayo 2**	**Ensayo 3**
Agar 557	0.009	0.006	0.023
Agar 578	99.98	99.83	99.82
Mycos 557	0.002	0.001	0.009
Mycos 578	0.008	0.17	0.148

Tabla 5-4: Clasificación Para Agar PDA Imagen Tomada con Filtro de 578nm

Cada una de las tablas tiene en la parte superior una etiqueta que indica, el experimento efectuado, por ejemplo, para la tabla **5-1** se ve la clasificación efectuada por el algoritmo para tres imágenes diferentes, en el primer ensayo, el sistema fue capaz de reconocer el patrón espectral dejado por el microorganismo con un 99.81 %, para esa imagen en concreto, dicha imagen no fue suministrada a la maquina para el entrenamiento, fue tomada posteriormente y en efecto corresponde a la etiqueta señalada, por lo cual se concluye que el sistema si tuvo la capacidad de clasificar correctamente esta etiqueta. El mismo procedimiento se realizó para *Colletotrichum gloeosporioides* con resultados similares. Cabe aclarar que este porcentaje, se dio en las condiciones controladas de experimentación aplicadas.

En resumen, después del entrenamiento con las imágenes espectrales, todas las veces que al sistema se le suministraron imágenes con las diferentes estructuras infectadas o no, fue capaz de responder de forma acertada, no importando el tipo de filtro o muestra, la razón de esto como se vio en los espectros y los histogramas 2D radica en que cada imagen crea una distribución muy particular de intensidades en los canales de color correspondientes, haciendo posible .ªprender.ª diferenciar estas características, por lo cual con un entrenamiento más profundo y en diferentes condiciones, sería posible generar un sistema robusto capaz de ejecutar esta tarea en el campo o la industria.

5.4. Detección de Colletotrichum gloeosporioides

En la detección de este fitopatógeno se buscaron las mismas bandas de Raman para quitina y $\beta 1,3-$ glucano, en ese sentido, se realizaron experimentos similares para detectar la presencia de este fitopatógeno, que tiene una alta variabilidad en su morfología, haciendo muy difícil la tarea de identificar aislados de esta especie con métodos tradicionales tales como análisis por microscopia, el cual requiere de un experto, lo cual mediante el método de aprendizaje de maquinas se puede omitir. [111].
Se estudiaron dos muestras diferentes de cáscara de mango, una con tejido enfermo evidente y una sana, ver figura **5-33**.

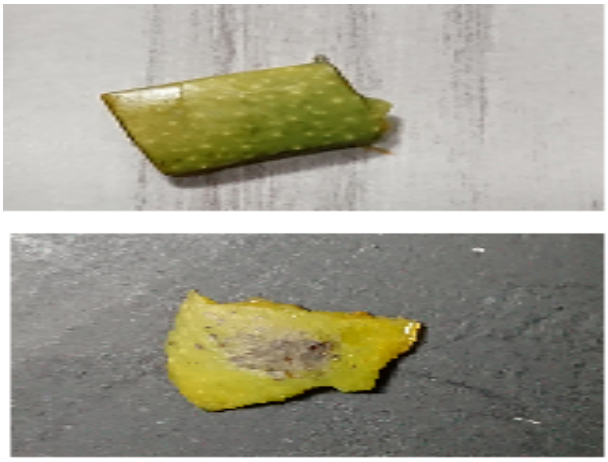


Figura 5-33: Muestras de Cascara de Mango con Tejido Enfermo y Sano

Este trabajo proporciona una solución poco convencional en la detección e identificación de *Colletotrichum gloeosporioides*, que generalmente se aborda mediante el uso de la PCR, una técnica de biología molecular que es costosa en términos de tiempo y dinero, por lo que el desarrollo ulterior del el método presentado aquí está justificado [112].
En la figura **5-35**, se presentan el espectro y una imagen sin filtrar, se resaltaron las bandas más relevantes, sin embargo, la fluorescencia en el experimento fue enorme, como se muestra en la figura **5-35** a) , b), esto probablemente sepultó algunas vibraciones interesantes.
Las imágenes se tomaron esta vez solo con un filtro de 557nm debido a la amplitud prominente de la fluorescencia en la zona de cobertura del filtro de 578nm, la figura **5-36** b) muestra el fuerte *"background"* en la región amarilla del espectro visible [570-588]nm.

Las figuras **5-36** a) y b) muestran nuevamente el fuerte impacto del fondo (*"Background"*) en la señal Raman, ambas imágenes se procesaron de la misma manera al tomar 10 imágenes al azar y promediarlas. Las diferencias son bastante obvias ya que la imagen del fondo tiene niveles de intensidad apreciablemente más altos, es como si la presencia del hongo

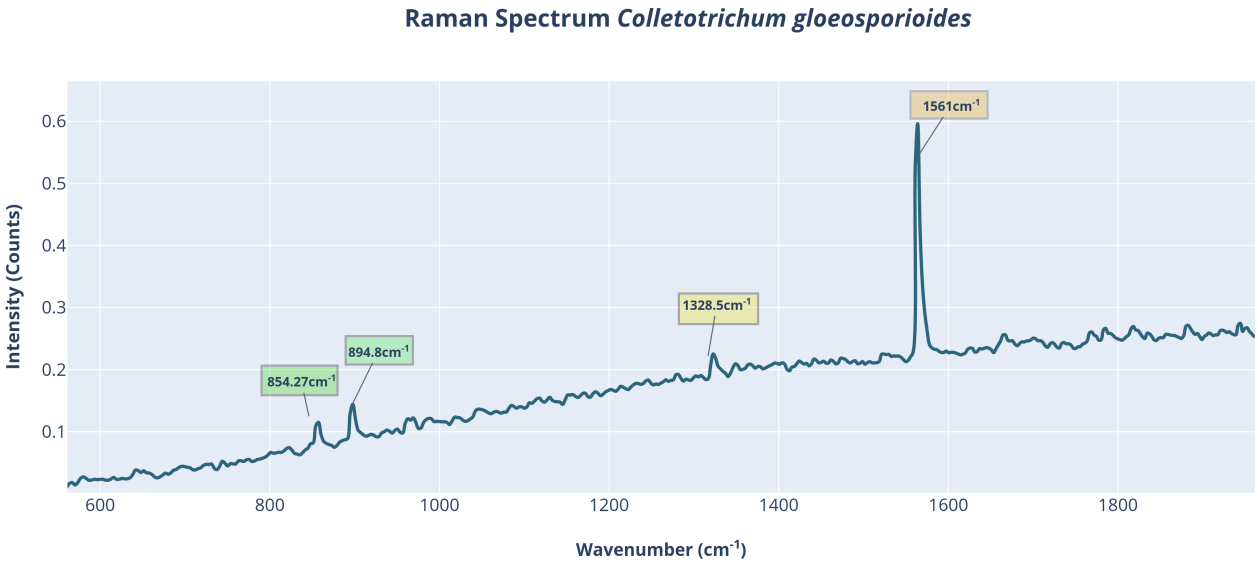
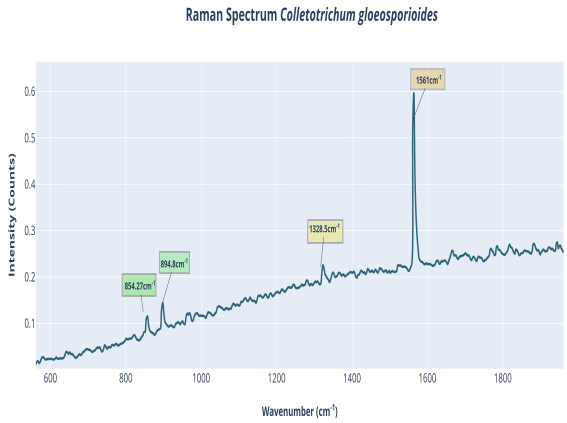


Figura 5-34: Espectro de *C. gloeosporioides*

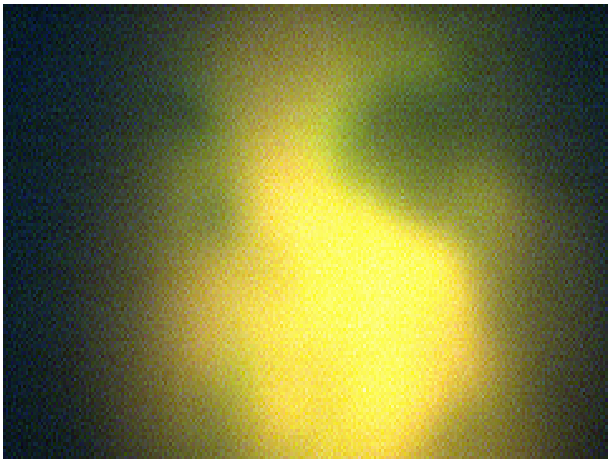
produjera un efecto de atenuación en la señal sobre la cáscara del mango. Además, hay un halo rojo alrededor del punto verde, lo que podría significar un posible filtrado en los bordes del filtro, pero no es importante ya que es el mismo tipo de ruido para ambas imágenes, el procesamiento de la imagen siguió las mismas pautas que en el caso de *M.fijiensis*.
La comparación entre los dos histogramas 2D muestra una saturación en el canal verde, más fuerte para *C.gloeosporioides* que en el caso del *"Background"*, esto significa que hay más información concentrada en la región espectral cubierta por el filtro Bayer verde, esto podría Ser debido a las bandas más fuertes se encuentran en esa región. Mientras tanto, en el histograma del *"Background"* hay una distribución más homogénea de las intensidades con valores más bajos en un área más amplia, algo que se espera para la fluorescencia y es consistente con la figura **5-35** b) en el lugar sin manchas, estos resultados nos permiten concluir que esta técnica, debido a su simplicidad de interpretación, podría implementarse en zonas sin personal capacitado.

Los resultados después del proceso de entrenamiento se muestran en las tablas siguientes. Nuevamente, se tomaron cien imágenes para alimentar la red neuronal y se proporcionaron dos etiquetas para el tejido sano de mango denominado background y el tejido infectado, denominado infected.

Las tablas **5-5**, **5-6**, revelan el poder de la metodología aquí presentada, no importa cuán sutiles sean las diferencias en las imágenes, el sistema siempre es capaz de diferenciar entre los dos casos, dando Absoluta certeza sobre la presencia de fitopatógenos. La interpretación de estas tablas al igual que en el caso anterior, expresan el nivel de seguridad con el

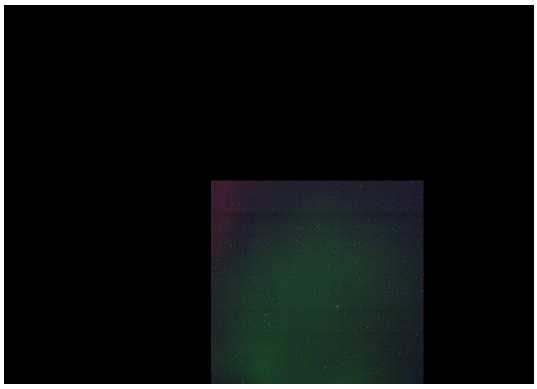


(a) Espectro de*Colletotrichum gloeosporioides*

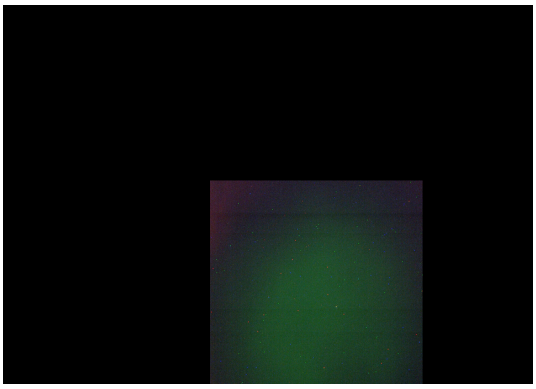


(b) *C. gloeosporioides* Sin Filtros

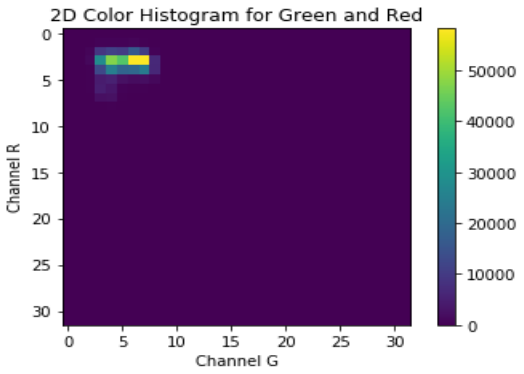
Figura 5-35: Espectro e Imagen sin Filtrar de *C. gloeosporioides*



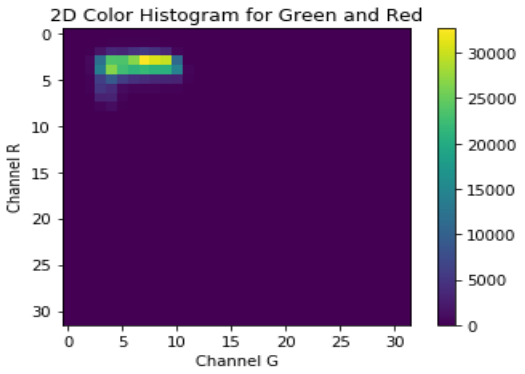
(a) Imagen Enmascarada con Filtro de 557nm*Colletotrichum gloeosporioides*



(b) Imagen Enmascarada Cascara de Mango Tejido Filtro de 557nm



(c) *C. gloeosporioides* Histograma 2D



(d) Histograma 2D Tejido Sano

Figura 5-36: *C. gloeosporioides* Imagenes Espectrales e Histograma 2D

***C.gloeosporioides* Pruebas de Detección de La Infección**			
% de Detección			
Etiqueta	**Ensayo 1**	**Ensayo 2**	**Ensayo 3**
Background	4.8	1.95	2.3
Infected	95.2	98.05	97.7

Tabla 5-5: Clasificación para tres Imágenes de Infección de *C.gloeosporioides* (557nm)

Ensayo con Imagenes de Tejido Sano			
% de Detección			
Etiqueta	**Ensayo 1**	**Ensayo 2**	**Ensayo 3**
Background	99.997	99.989	99.921
Infected	0.003	0.011	0.079

Tabla 5-6: Clasificación de Tres Imágenes Diferentes Para Tejido Sano de Cascara de Mango

cual la red neuronal pudo clasificar las imágenes suministradas, como se menciono, ninguna de las fotos con tejido sano e infectado fueron previamente utilizadas en el proceso de entrenamiento de la red neuronal, en todos lo casos aquí presentados el sistema clasifico correctamente las imágenes, es decir, en la columna etiquetada como ensayo 1 de la tabla **5-5**, el sistema indica que esta un 95.2 % seguro de que esa imagen corresponde a tejido de cascara de mango infectado con el fitopatógeno, lo cual, es acertado. De nuevo es necesario reiterar que dicha imagen no fue suministrada en el entrenamiento lo que es evidencia de que el sistema encontró un patrón de distribución de energía generada por el esparcimiento inelástico característico de la presencia de *C.gloeosporioides* sobre la cascara de mango.

6 Conclusiones

- Se desarrolló un conjunto de herramientas para crear un sistema capaz de detectar fitopatógenos de origen fúngico, combinando algoritmos de procesamiento de imágenes, electrónica, óptica, técnicas de aprendizaje profundo y diseño mecánico. Todo el sistema es una sinergia de aquellos elementos que junto con el conocimiento de la fenomenología de esparcimiento inelástico, introduce en el entorno agrícola colombiano un diseño que podría competir con soluciones similares de origen extranjero.

- La estrategia en la selección de filtros, permitió acotar el conjunto de números de onda a regiones donde las posible señales inelásticas asociadas a fitopatógenos fúngicos se manifiestan de forma exclusiva, permitiendo establecer su presencia tanto de forma individual como sobre tejido vegetal, dicha estrategia se baso en la revisión de decenas de trabajos similares en el área, principalmente aquellos donde se estudiaba la bioquímica de la pared celular de estos organismos, la cual *.A priori"* era la estructura celular mas accesible para la radiación láser. Lo anterior se pudo constatar en el desarrollo de este trabajo.

- El controlador PID le dio al instrumento una gran versatilidad ya que es posible controlar la temperatura para proporcionar las condiciones adecuadas a algunos organismos, por ejemplo, algunas cepas de hongos como *B.cinerea* tienen una temperatura de crecimiento óptima de alrededor de 15°C. La ventaja de este tipo de controlador es la precisión en el mantenimiento de una temperatura constante (+/- 0.1°C). Otro aspecto importante del equipo es la posibilidad de controlar la potencia del láser, ya que algunas de las situaciones más problemáticas cuando se usa este tipo de radiación para estudiar muestras biológicas es el daño potencial de la muestra, esto se evitó en este equipo mediante el uso de una modulación PWM. que puede controlar la intensidad del láser, otorgando condiciones de seguridad adecuadas a este tipo de muestras.

- El uso de piezas recicladas fue crucial para la configuración óptica desarrollada, de esta manera un proyecto inviable a primera vista en términos de costos, fue posible a través de algunas adaptaciones de estos elementos mediante impresión 3D y maquinado, ser utilizados para el desarrollo del equipo.

- Los algoritmos de procesamiento de imágenes demostraron ser un eficaz complemento en el análisis de la señal de esparcimiento Raman, en el sentido de conjugar la información obtenida de los espectros con las imágenes de forma simultanea. En los

experimentos con *C. gloeosporioides* y *M.fijiensis*, las operaciones sobre la imagen y el subsiguiente entrenamiento de aprendizaje profundo dieron a la computadora la capacidad de evaluar las características de la imagen y concluir sobre la presencia de el organismo sobre la estructura vegetal de interés de forma exitosa.

- La virtud del equipo desarrollado, al disponer de un doble canal de registro (imágenes y espectros) hace factible que el usuario tenga la posibilidad de usar una imagen significativa asociada con la presencia de los fitopatógenos de una forma más expedita, que en comparación con una curva espectral puede resultar mas amigable en el mensaje. Posibilitando en un futuro, la necesidad de la presencia de personal experto.

- Según los resultados obtenidos para *C.gloeosporioides* donde solo se usó un filtro y de igual manera fue posible detectar la presencia del fitopatógeno, deja entrever que la configuración podría simplificarse aún mas al menos en este tipo de aplicaciones, el uso del otro filtro se puede justificar si La intención de este trabajo fuese estudiar estructuras muy específicas en la pared celular de los hongos, de tal forma que la superposición de imágenes permita observar estructuras con algún interés particular.

- El éxito del equipo respecto a la detección de ambos fitopatógenos se debe fundamentalmente a la forma como se realizo el arreglo óptico, el cual fue ligado exclusivamente a la recolección de fotones característicos del esparcimiento Raman, lo cual simplifica las tareas de calculo, logrando de esta forma obtener un sistema robusto desde el punto de vista de la detección, pero liviano desde la perspectiva del costo computacional. Lo anterior abre la puerta a la posibilidad de integrar esta solución en sistemas mucho más portátiles.

- Uno de los principales logros de este trabajo, fue la posibilidad de llevar información espectral a un sistema de adquisición de imágenes, lo que repercutirá a futuro en facilitar la capacitación de personal no experto en el diagnostico de cultivos, por lo cual es posible afirmar que la dirección que tomó este proyecto fue la de cerrar un poco la brecha tecnológica que existe entre los agricultores locales y aquellos en naciones desarrolladas, lo que posiblemente tenga repercusión en un mejoramiento de la competitividad del agro nacional, cuando desde la universidad existan políticas reales y efectivas de transmitir el conocimiento hacia la sociedad.

6.1. Proyección

Como expectativa de este trabajo se prevé la reproducibilidad del equipo, con el propósito de que al menos por cada asociación de agricultores se disponga de uno de ellos, empezando por Antioquia y en algún momento cubriendo todo el país.

6.2. Aportes

El presente trabajo deja como resultado los siguientes productos:

6.2.1. Óptica

Fue posible desarrollar un sistema óptico compuesto por filtros de diferentes tipos qué permitió depurar las señales inelásticas provenientes de la interacción del láser con las diferentes muestras, esta configuración presentada para la detección de fitopatógenos, no presenta en la literatura especializada actual una alternativa similar, hasta donde se ha podido verificar. Por lo anterior, es posible que este instrumento pudiese ser sometido a un proceso de patente. Otro punto importante a resaltar, fue la potenciación y adaptación del uso de dos microscopios, considerados obsoletos por la Universidad, los cuales mediante la reconfiguración de sus partes pudieron extender sus vidas útiles.

6.2.2. Sistema de Enfriamiento

Este trabajo contribuyo también en el diseño de un sistema de enfriamiento especializado, para el trabajo con muestras biológicas, adaptando una celda Peltier al porta-muestras de uno de los microscopios y mediante un sistema de disipación de calor liquido conjugado con un controlador PID, permitió reducir la temperatura de la muestra hasta los 10°C.

6.2.3. Algoritmos de Procesamiento de Datos Espectrales

La detección de fitopatógenos no hubiese sido posible sin la capacidad de analizar los datos de manera personalizada, la razón de esto, es que los software de muchos de estos equipos comerciales poseen impedimentos legales y técnicos que evitan un apropiado procesamiento de datos, por lo anterior se crearon una serie de algoritmos que permiten extraer la información del espectrómetro, logrando de esta manera un manejo personalizado de los datos, permitiendo aplicar las técnicas numéricas mas recientes sin ninguna restricción. Estos algoritmos permiten eliminar la señal de background a oscuras (ruido), normalizar los datos y realizar la conversión de longitudes onda a números de onda. así mismo permite importarlos a la plataforma plotly, la cual da acceso libre a una serie de aplicativos enfocados a la personalización de gráficos y la aplicación de técnicas numéricas sobre los datos.

6.2.4. Algoritmos de Procesamiento de Imágenes

Estos algoritmos permiten realizar varias tareas sobre las imágenes, una de ellas es la intensificación de la señal, lo cual consiste en tomar una serie de imágenes de bajo contraste, en las cuales no se aprecian detalles y mediante la adición de cada una de ellas y su posterior promedio, se crea una imagen nueva, permitiendo distinguir características espectrales antes

no apreciables, la motivación de este script es evitar descartar experimentos exitosos por la aparente falta de información. También se desarrollaron herramientas de software para la separación de los canales RGB, lo cual se desarrolló con el fin de mostrar la distribución de longitudes de onda en los canales rojo y verde mediante los histogramas 2D, los cuales son los de interés para la aplicación aquí propuesta y que permiten identificar los parámetros espectrales que posteriormente serán introducidos en la red neuronal.

6.2.5. Diseño Electrónico

Para este trabajo fue necesario crear un sistema que permitiese la integración de multiples dispositivos, tales como, láser, sistema de refrigeración, control de temperatura PID y el encendido y apagado de cada uno de los periféricos a través del computador, por lo cual se creo un circuito electrónico para esta tarea, el cual se adapta a un microcontrolador para efectuar cada una de las tareas señaladas, cabe aclarar, que el microcontrolador lleva embebido el controlador PID, mientras que el computador es el encargado de controlar el encendido de la cámara, la intensidad del láser y la bomba del sistema de enfriamiento liquido mediante una interfaz gráfica de usuario (GUI), diseñada específicamente para tener un control total de las variables del experimento.

6.2.6. Detector de Fitopatógenos

El producto final, no es mas que la integración de todos los elementos señalados anteriormente, básicamente se trata de un sistema óptico que adapta la señal inelástica producida por la interacción del láser con la muestra y que posteriormente se emplea para tomar una serie de fotografías con información espectral, la cual es utilizada para entrenar una red neuronal artificial que es la encargada de clasificar estas imágenes de las muestras y discernir si se encuentra infectada o no. Lo anterior más allá de un simple dispositivo es toda una metodología que brinda la posibilidad de solucionar una problemática de la agro-industria Colombiana, como lo es la detección temprana de fitopatógenos.

6.2.7. Productos Derivados del Trabajo

- Patent Pending: "High Spectral Resolution Monochromator".
- Patent Pending: "QC-Eye: Quality Control Portable Device"(US Patent)
- Article: Ïnelastic Scattering System for *Pseudocercospora fijiensis* Detection.
- Poster XV ENO – VI CANCOA. 2017

7 Apéndice

7.1. Algoritmos en Python y C

Los siguientes son los scripts implementados para este trabajo, incluye los algoritmos de aprendizaje profundo, el procesamiento de datos espectrales y el procesamiento de imágenes.

Procesamiento de Datos Espectrales

```
# -*- coding: utf-8 -*-
import plotly.plotly as py
import plotly.graph_objs as go
import pandas as pd
import matplotlib.pyplot as plt

headers = ['Pixel Number','Wavelength (nm)','Sum Spectrum','Averaged Spectrum',
#headers = ['Pixel Number','Wavelength (nm)','Sum Spectrum','Averaged Spectrum
df = pd.read_csv('colleto7s3avg.CSV',names=headers, index_col='Pixel Number', h
#bg = pd.read_csv('backgrounsic8s3avg.CSV',names=headers, index_col='Pixel Num



#df['Wavelength (nm)'] = df['Wavelength (nm)'].map(lambda x: datetime.strptime(
x = df['Wavelength (nm)']
#w = bg['Wavelength (nm)']
#t = bg['Averaged Spectrum']

r = df['Averaged Spectrum']

resto=0

O = r.astype(float)

contador=0
parameter = 100
```

```
while contador < len(O):

    if (O[contador])<parameter: #Homogenizador de datos

        resto = parameter - O[contador]

        O[contador] = O[contador]-resto

        if(O[contador]<0):

            O[contador]=resto




    contador += 1

z=((1/532.8)-(1/x))*1e7

trace0 = go.Scatter(
    x = z,
    y = (O )/8000,
 #  y = df['Averaged Spectrum'],
    mode ='lines+markers',
    marker = dict(
          size = 10,
          color = 'rgba(155, 182, 193, .9)',
          line = dict(
              width = 2,
          )
    )


)

data=[trace0]
# plot

layout = dict(title = 'Raman Spectrum',
```

```
67                 yaxis = dict(zeroline = False, range = (0,1), title = 'Intensity'
68                 xaxis = dict(zeroline = False, range = (500,2000), title = 'Waven
69
70               )
71
72 fig = dict(data=data, layout=layout)
73
74
75
76 py.plot(fig, filename = 'coleto', auto_open=True)
77 #plt.plot(x,y)
78 # beautify the x-labels
79 #plt.gcf().autofmt_xdate()
80
81 #plt.show()
```

7.1.1. Procesamiento de Imágenes

Corrección Gamma y Mejora de Intensidad

```
1 # -*- coding: utf-8 -*-
2 from __future__ import print_function
3 import numpy as np
4 import argparse
5 from matplotlib import pyplot as plt
6 import cv2
7 #from PIL import Image
8 import glob
9 image_list = []
10 (rAvg, gAvg, bAvg) = (None, None, None)
11 total = 0
12 i = 1
13 for filename in glob.glob('imagenes/backgroundColleto/*.jpg'): #assuming gif
14     im = cv2.imread(filename)
15     (B,G,R) = cv2.split(im.astype(float))
16     if rAvg is None:
17         rAvg = R
18         bAvg = B
19         gAvg = G
20     else:
21         rAvg = ((total * rAvg) + (1 * R)) / (total + 1.0)
```

```
22          gAvg = ((total * gAvg) + (1 * G)) / (total + 1.0)
23          bAvg = ((total * bAvg) + (1 * B)) / (total + 1.0)
24      total += 1
25
26      im = im.astype(float)
27      image_list.append(im)
28
29 avg = cv2.merge([bAvg, gAvg, rAvg]).astype("uint8")
30 gamma = 1.8
31 invGamma = 1.0 / gamma
32 teo = 1.0 / 2.5
33 table = np.array([((i / 255.0) ** invGamma) * 255
34 for i in np.arange(0, 256)]).astype("uint8")
35
36 table1 = np.array([((j / 255.0) ** teo) * 255
37 for j in np.arange(0, 256)]).astype("uint8")
38
39 uno = cv2.LUT(avg, table)
40 dos = cv2.LUT(avg, table1)
41 plt.subplot(211),plt.imshow(uno)
42 plt.xticks([]), plt.yticks([])
43 plt.subplot(212),plt.imshow(dos)
44 plt.xticks([]), plt.yticks([])
45 #plt.show()
46 #plt.imshow(uno)
47 cv2.imwrite(r'backcolleto55718.png',uno)
48 cv2.imwrite(r'backcolleto55725.png',dos)
49
50
51
52
53 #OutImage = image_list[0]
54
55 #while(i<=(len(image_list)-1)):
56
57  #      OutImage = cv2.addWeighted(OutImage,1/len(image_list),image_list[i],1
58   #     i += 1
```

Histogramas 2D y Enmascaramiento

```
1 # -*- coding: utf-8 -*-
```

```
2
3 import cv2
4 import numpy as np
5 from matplotlib import pyplot as plt
6
7 img = cv2.imread('backcolleto55718.png')
8
9 # create a mask
10 mask = np.zeros(img.shape[:2], np.uint8)
11 mask[500:1100, 750:1520] = 255
12 masked_img = cv2.bitwise_and(img,img,mask = mask)
13
14 # Calculate histogram with mask and without mask
15 # Check third argument for mask
16 hist_full = cv2.calcHist([img],[0],None,[256],[0,256])
17 hist_mask = cv2.calcHist([img],[0],mask,[256],[0,256])
18
19 plt.subplot(221), plt.imshow(img, 'gray')
20 plt.subplot(222), plt.imshow(masked_img,'gray')
21 #plt.subplot(223), plt.imshow(masked_img, 'gray')
22 plt.subplot(224), plt.plot(hist_mask,'r')
23 plt.xlim([1,100])
24
25 plt.show()
26 cv2.imwrite('backcolleto55718masked.png',masked_img)
27
28 chans = cv2.split(masked_img)
29 colors = ("b", "g", "r")
30 plt.figure()
31 plt.title("'Flattened' Color Histogram")
32 plt.xlabel("Bins")
33 plt.ylabel("# of Pixels")
34 features = []
35
36
37 for (chan, color) in zip(chans, colors):
38         # create a histogram for the current channel and
39         # concatenate the resulting histograms for each
40         # channel
41         hist = cv2.calcHist([chan], [0], mask, [256], [0, 256])
```

```
42          features.extend(hist)
43
44          # plot the histogram
45          plt.plot(hist, color = color)
46          plt.xlim([0, 256])
47
48 histo = cv2.calcHist([chans[2], chans[1]], [0, 1], mask,
49          [32,32], [0, 256, 0, 256])
50 fig = plt.figure()
51 p = plt.imshow(histo, interpolation = "nearest")
52 plt.title("2D Color Histogram for Green and Red")
53 plt.xlabel("Channel G")
54 plt.ylabel("Channel R")
55 plt.colorbar(p)
56 cv2.imwrite('backcolleto557182d.png',histo)
```

7.1.2. Interfase de Control

```
1 # -*- coding: utf-8 -*-
2 import sys, time, serial, traceback
3 from PyQt5.QtWidgets import QMainWindow, QApplication, QCheckBox, QSlider
4 from PyQt5.QtCore import QObject, QThread, pyqtSignal, pyqtSlot
5 from PyQt5 import uic
6 from PyQt5.QtGui import *
7 from PyQt5.QtWidgets import *
8 from PyQt5.QtCore import *
9
10 # Cargar nuestro archivo .ui
11 Ui_MainWindow, QtBaseClass = uic.loadUiType('darthvader.ui')
12 i=0
13 #arduinoSerialData = serial.Serial('COM4',9600)
14 #SERIALPORT = 'COM3' #Arduino Mega
15 SERIALPORT = 'COM6'#Arduino Due
16 ser = serial.Serial(SERIALPORT, 57600, bytesize=8,parity='N', stopbits=1)
17 rawdata = []
18 count = 0
19 flag = 0
20
21 class WorkerSignals(QObject):
22     '''
23     Defines the signals available from a running worker thread.
```

```
24
25      Supported signals are:
26
27      finished
28          No data
29
30      error
31          `tuple` (exctype, value, traceback.format_exc() )
32
33      result
34          `object` data returned from processing, anything
35
36      progress
37          `int` indicating % progress
38
39      '''
40      finished = pyqtSignal()
41      error = pyqtSignal(tuple)
42      result = pyqtSignal(object)
43      progress = pyqtSignal(int)
44
45  class Worker(QRunnable):
46      '''
47      Worker thread
48
49      Inherits from QRunnable to handler worker thread setup, signals and wrap-up
50
51      :param callback: The function callback to run on this worker thread. Suppli
52                       kwargs will be passed through to the runner.
53      :type callback: function
54      :param args: Arguments to pass to the callback function
55      :param kwargs: Keywords to pass to the callback function
56
57      '''
58
59      def __init__(self, fn, *args, **kwargs):
60          super(Worker, self).__init__()
61
62          # Store constructor arguments (re-used for processing)
63          self.fn = fn
```

```
64          self.args = args
65          self.kwargs = kwargs
66          self.signals = WorkerSignals()
67
68          # Add the callback to our kwargs
69          self.kwargs['progress_callback'] = self.signals.progress
70
71      @pyqtSlot()
72      def run(self):
73          '''
74          Initialise the runner function with passed args, kwargs.
75          '''
76
77          # Retrieve args/kwargs here; and fire processing using them
78          try:
79              result = self.fn(*self.args, **self.kwargs)
80          except:
81              traceback.print_exc()
82              exctype, value = sys.exc_info()[:2]
83              self.signals.error.emit((exctype, value, traceback.format_exc()
84          else:
85              self.signals.result.emit(result)  # Return the result of the pr
86          finally:
87              self.signals.finished.emit()  # Done
88
89
90
91
92
93
94
95
96
97  class MyApp(QMainWindow):
98
99
100     def __init__(self, *args, **kwargs):
101         super(MyApp, self).__init__(*args, **kwargs)
102         self.ui = Ui_MainWindow()
103         self.ui.setupUi(self)
```

```
104             self.ui.SetP.clicked.connect(self.PowerB_clicked)
105             self.ui.PowerB.clicked.connect(self.SetP_clicked)
106             self.ui.ON_button.clicked.connect(self.ON_clicked)
107             self.ui.checkBox.stateChanged.connect(self.Camera_clicked)
108             self.ui.verticalSlider.sliderReleased.connect(self.Lamp)
109             self.threadpool = QThreadPool()
110             print("Multithreading with maximum %d threads" % self.threadpool.maxThr
111
112
113             self.timer = QTimer()
114             self.timer.setInterval(450)
115             self.timer.timeout.connect(self.recurring_timer)
116             self.timer.start()
117
118         def Lamp(self):
119             # time.sleep(0.2)
120               ser.write(b'8')
121               lamp = str(self.ui.verticalSlider.value())
122               ser.write(lamp.encode())
123               self.ui.SlideValue.setText(str(lamp))
124               print(ser.readline().decode())
125
126
127
128
129         def PowerB_clicked(self):
130
131             ser.write(b'1')
132 #             while(count<3):
133  #                 rawdata.append(str(ser.readline().rstrip()))
134   #                count += 1
135
136             # ser.write(b'20')
137               setpoint = self.ui.lineEdit.text()
138               TempsetPoint = setpoint.encode()
139               ser.write(TempsetPoint)
140  #            self.ui.Power.setText(str(setpoint))
141               print(ser.readline().decode())
142               time.sleep(1)
143 # str(setpoint.decode('utf-8'))
```

```
144 # Evento del boton btn_FtoC
145     def SetP_clicked(self):
146         ser.write(b'2')
147 
148       #
149         potencia = self.ui.Power.text()
150         data = potencia.encode()
151         print(data)
152         ser.write(data)
153       #  if(ser.inWaiting()>0):
154 
155         Temp = ser.readline()
156      #    Temp = (Temp[0:len(Temp)-2]).decode("utf-8")
157       #  print(Temp)
158 #         self.ui.lineEdit.setText(str(Temp))
159 
160     def ON_clicked(self):
161         global flag
162         ser.write(b'3')
163         if (flag == 0):
164             ser.write(b'0')
165             flag = 1
166 
167         else:
168             ser.write(b'5')
169             print(ser.readline())
170             flag = 0
171 
172     def Camera_clicked(self, camera):
173         # global camera
174          ser.write(b'4')
175          if (camera == Qt.Checked):
176               ser.write(b'1')
177               #camera = 1
178 
179          else:
180               ser.write(b'6')
181               #camera = 0
182 
183         #flag = ser.read()
```

```
184         #flag = (flag[0:len(flag)-2].decode("utf-8"))
185         #if flag==1:
186          #    ser.write(b'3')
187
188
189
190
191
192       #   self.ui.Temperatura.setText(str(cel))
193
194   # def readTemp(self):
195
196
197     def recurring_timer(self):
198                         # ser.write(b'0')
199                          if (ser.inWaiting()>0):
200                              myData = ser.readline().rstrip()
201
202                              self.ui.Temperatura.setText(str((myData.decode('asc
203                 # global i
204                 # i += 1
205                  #print(i)
206                  #self.ui.Temperatura.setText(str(i))
207
208
209
210
211
212
213 if __name__ == '__main__':
214
215
216     app = QApplication(sys.argv)
217     window = MyApp()
218     window.show()
219     sys.exit(app.exec_())
```

7.2. Controlador PID: Arduino

```
1 //#include "DHT.h"
```

```
2 #include <PID_v1.h>
3 //#define DHTPIN 2     // what digital pin we're connected to
4 #define analogPin A0
5 #define beta 4090 //the beta of the thermistor
6 #define resistance 10 //the value of the pull-down resistor
7 #define power   4     //Peltier
8 // Uncomment whatever type you're using!
9 //#define DHTTYPE DHT11   // DHT 11
10 //#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321
11 //#define DHTTYPE DHT21   // DHT 21 (AM2301)
12 # define led 9 //Laser
13 #define  fan 24
14 #define  camera 30
15 #define  lamp 2
16
17
18 // Connect pin 1 (on the left) of the sensor to +5V
19 // NOTE: If using a board with 3.3V logic like an Arduino Due connect pin 1
20 // to 3.3V instead of 5V!
21 // Connect pin 2 of the sensor to whatever your DHTPIN is
22 // Connect pin 4 (on the right) of the sensor to GROUND
23 // Connect a 10K resistor from pin 2 (data) to pin 1 (power) of the sensor
24
25 // Initialize DHT sensor.
26 // Note that older versions of this library took an optional third paramete
27 // tweak the timings for faster processors.  This parameter is no longer ne
28 // as the current DHT reading algorithm adjusts itself to work on faster pr
29 //DHT dht(DHTPIN, DHTTYPE);
30
31 const int On = 22;
32 double Setpoint, Input, Output;
33
34
35 //Specify the links and initial tuning parameters
36 PID myPID(&Input, &Output, &Setpoint,250,15,20, REVERSE);
37 int data;
38 int flag;
39 int i = 0;
40 void setup() {
41   Serial.begin(576);
```

```
42  // Serial.println("DHTxx test!");
43    Setpoint = 22;
44    pinMode(lamp,OUTPUT);
45    pinMode(led,OUTPUT);
46    pinMode(power,OUTPUT);
47    pinMode(On,OUTPUT);
48    pinMode(camera,OUTPUT);//Control Camera On/Off port 30
49    pinMode(fan,OUTPUT); // Control Fan On/Off port 24
50 //   pinMode(On1,OUTPUT);
51    myPID.SetMode(AUTOMATIC);
52 while (!Serial) {
53      ; // wait for serial port to connect. Needed for native USB
54    }
55  // dht.begin();
56 }
57
58 void loop() {
59    // Wait a few seconds between measurements.
60
61
62    double a =1023 - analogRead(analogPin);
63    double tempC = beta /(log((1025.0 * 10 / a - 10) / 10) + beta / 298.0) - 273.
64       Input=tempC;
65       data = Serial.read();
66
67       switch(data){
68         case '1' :
69          {
70         while(!Serial.available()){}
71         Setpoint = Serial.parseFloat();
72
73          Serial.println(String(Output));
74         // Serial.flush();
75          break;
76          }
77         case '2':
78         {
79
80
81        while(!Serial.available()){}
```

```
82          int asus= Serial.parseInt();
83      //    float f = asus; //convert the array into a float
84          float conversion = 2.55*asus;
85
86          analogWrite(led,conversion);
87          Serial.println(String(conversion));
88       //   Serial.flush();
89
90          delay(50);
91
92
93          break;
94          }
95
96          case '3':{
97            while(!Serial.available()){}
98            int status = Serial.parseInt();
99
100           if(status == 0){
101           digitalWrite(On,HIGH);
102           digitalWrite(fan,HIGH);
103           Serial.println(status);
104           }
105           else if(status == 5){
106             digitalWrite(On,LOW);
107             digitalWrite(fan,LOW);
108             Serial.println("OK");
109
110           }
111
112
113           break;
114         }
115
116         case '4':{
117           while(!Serial.available()){}
118           int on_camera = Serial.parseInt();
119           if(on_camera == 1){
120           digitalWrite(camera,HIGH);
121
```

```
122             }
123             else if(on_camera == 6){
124               digitalWrite(camera,LOW);
125
126
127             }
128             break;
129           }
130
131           case '8':{
132             while(!Serial.available()){}
133             int slider = (Serial.parseInt());
134             int riven = 2.55*abs(slider);
135             delay(10);
136             analogWrite(lamp,riven);
137             Serial.println("Value: "+riven);
138
139             break;
140           }
141
142
143
144
145
146
147         }
148     Serial.println(Input);
149     data = 0;
150     myPID.Compute();
151     analogWrite(power,Output);
152     delay(200);
153
154 }
155 }
```

7.3. Redes Neuronales y Algoritmos de Clasificación

Los siguientes algoritmos son los diseñados por Google, no hay modificaciones, los que se utilizan para el proceso de entrenamiento para el dispositivo portátil o la detección de hongos se modificaron de los que se presentan aquí.

```
# Copyright 2015 The TensorFlow Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ==============================================================================
r"""Simple transfer learning with Inception v3 or Mobilenet models.

With support for TensorBoard.

This example shows how to take a Inception v3 or Mobilenet model trained on
ImageNet images, and train a new top layer that can recognize other classes
images.

The top layer receives as input a 2048-dimensional vector (1001-dimensional
Mobilenet) for each image. We train a softmax layer on top of this
representation. Assuming the softmax layer contains N labels, this corresp
to learning N + 2048*N (or 1001*N) model parameters corresponding to the
learned biases and weights.

Here's an example, which assumes you have a folder containing class-named
subfolders, each full of images for each label. The example folder flower_p
should have a structure like this:

~/flower_photos/daisy/photo1.jpg
~/flower_photos/daisy/photo2.jpg
...
~/flower_photos/rose/anotherphoto77.jpg
...
~/flower_photos/sunflower/somepicture.jpg

The subfolder names are important, since they define what label is applied
```

```
41 each␣image,␣but␣the␣filenames␣themselves␣don't␣matter.␣Once␣your␣images␣are
42 prepared,␣you␣can␣run␣the␣training␣with␣a␣command␣like␣this:
43
44
45 ```bash
46 bazel␣build␣tensorflow/examples/image_retraining:retrain␣&&␣\
47 bazel-bin/tensorflow/examples/image_retraining/retrain␣\
48 ␣␣␣␣--image_dir␣~/flower_photos
49 ```
50
51 Or,␣if␣you␣have␣a␣pip␣installation␣of␣tensorflow,␣`retrain.py`␣can␣be␣run
52 without␣bazel:
53
54 ```bash
55 python␣tensorflow/examples/image_retraining/retrain.py␣\
56 ␣␣␣␣--image_dir␣~/flower_photos
57 ```
58
59 You␣can␣replace␣the␣image_dir␣argument␣with␣any␣folder␣containing␣subfolders␣of
60 images.␣The␣label␣for␣each␣image␣is␣taken␣from␣the␣name␣of␣the␣subfolder␣it's
61 in.
62
63 This␣produces␣a␣new␣model␣file␣that␣can␣be␣loaded␣and␣run␣by␣any␣TensorFlow
64 program,␣for␣example␣the␣label_image␣sample␣code.
65
66 By␣default␣this␣script␣will␣use␣the␣high␣accuracy,␣but␣comparatively␣large␣and
67 slow␣Inception␣v3␣model␣architecture.␣It's␣recommended␣that␣you␣start␣with␣this
68 to␣validate␣that␣you␣have␣gathered␣good␣training␣data,␣but␣if␣you␣want␣to␣deplo
69 on␣resource-limited␣platforms,␣you␣can␣try␣the␣`--architecture`␣flag␣with␣a
70 Mobilenet␣model.␣For␣example:
71
72 ```bash
73 python␣tensorflow/examples/image_retraining/retrain.py␣\
74 ␣␣␣␣--image_dir␣~/flower_photos␣--architecture␣mobilenet_1.0_224
75 ```
76
77 There␣are␣32␣different␣Mobilenet␣models␣to␣choose␣from,␣with␣a␣variety␣of␣file
78 size␣and␣latency␣options.␣The␣first␣number␣can␣be␣'1.0',␣'0.75',␣'0.50',␣or
79 '0.25'␣to␣control␣the␣size,␣and␣the␣second␣controls␣the␣input␣image␣size,␣eithe
80 '224',␣'192',␣'160',␣or␣'128',␣with␣smaller␣sizes␣running␣faster.␣See
```

```
https://research.googleblog.com/2017/06/mobilenets-open-source-models-for.h
for more information on Mobilenet.

To use with TensorBoard:

By default, this script will log summaries to /tmp/retrain_logs directory

Visualize the summaries with this command:

tensorboard --logdir /tmp/retrain_logs

"""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse
import collections
from datetime import datetime
import hashlib
import os.path
import random
import re
import sys
import tarfile

import numpy as np
from six.moves import urllib
import tensorflow as tf

from tensorflow.python.framework import graph_util
from tensorflow.python.framework import tensor_shape
from tensorflow.python.platform import gfile
from tensorflow.python.util import compat

FLAGS = None

# These are all parameters that are tied to the particular model architectu
# we're using for Inception v3. These include things like tensor names and
# sizes. If you want to adapt this script to work with another model, you w
```

```
# need to update these to reflect the values in the network you're using.
MAX_NUM_IMAGES_PER_CLASS = 2 ** 27 - 1  # ~134M


def create_image_lists(image_dir, testing_percentage, validation_percentage):
  """Builds a list of training images from the file system.

  Analyzes the sub folders in the image directory, splits them into stable
  training, testing, and validation sets, and returns a data structure
  describing the lists of images for each label and their paths.

  Args:
    image_dir: String path to a folder containing subfolders of images.
    testing_percentage: Integer percentage of the images to reserve for tests.
    validation_percentage: Integer percentage of images reserved for validation

  Returns:
    A dictionary containing an entry for each label subfolder, with images spli
    into training, testing, and validation sets within each label.
  """
  if not gfile.Exists(image_dir):
    tf.logging.error("Image directory '" + image_dir + "' not found.")
    return None
  result = collections.OrderedDict()
  sub_dirs = [
    os.path.join(image_dir,item)
    for item in gfile.ListDirectory(image_dir)]
  sub_dirs = sorted(item for item in sub_dirs
                    if gfile.IsDirectory(item))
  for sub_dir in sub_dirs:
    extensions = ['jpg', 'jpeg', 'JPG', 'JPEG']
    file_list = []
    dir_name = os.path.basename(sub_dir)
    if dir_name == image_dir:
      continue
    tf.logging.info("Looking for images in '" + dir_name + "'")
    for extension in extensions:
      file_glob = os.path.join(image_dir, dir_name, '*.' + extension)
      file_list.extend(gfile.Glob(file_glob))
    if not file_list:
```

```
161       tf.logging.warning('No files found')
162       continue
163     if len(file_list) < 20:
164       tf.logging.warning(
165           'WARNING: Folder has less than 20 images, which may cause issues.
166     elif len(file_list) > MAX_NUM_IMAGES_PER_CLASS:
167       tf.logging.warning(
168           'WARNING: Folder {} has more than {} images. Some images will '
169           'never be selected.'.format(dir_name, MAX_NUM_IMAGES_PER_CLASS))
170     label_name = re.sub(r'[^a-z0-9]+', ' ', dir_name.lower())
171     training_images = []
172     testing_images = []
173     validation_images = []
174     for file_name in file_list:
175       base_name = os.path.basename(file_name)
176       # We want to ignore anything after '_nohash_' in the file name when
177       # deciding which set to put an image in, the data set creator has a u
178       # grouping photos that are close variations of each other. For exampl
179       # this is used in the plant disease data set to group multiple pictur
180       # the same leaf.
181       hash_name = re.sub(r'_nohash_.*$', '', file_name)
182       # This looks a bit magical, but we need to decide whether this file s
183       # go into the training, testing, or validation sets, and we want to k
184       # existing files in the same set even if more files are subsequently
185       # added.
186       # To do that, we need a stable way of deciding based on just the file
187       # itself, so we do a hash of that and then use that to generate a
188       # probability value that we use to assign it.
189       hash_name_hashed = hashlib.sha1(compat.as_bytes(hash_name)).hexdigest
190       percentage_hash = ((int(hash_name_hashed, 16) %
191                           (MAX_NUM_IMAGES_PER_CLASS + 1)) *
192                          (100.0 / MAX_NUM_IMAGES_PER_CLASS))
193       if percentage_hash < validation_percentage:
194         validation_images.append(base_name)
195       elif percentage_hash < (testing_percentage + validation_percentage):
196         testing_images.append(base_name)
197       else:
198         training_images.append(base_name)
199     result[label_name] = {
200         'dir': dir_name,
```

```
201         'training': training_images,
202         'testing': testing_images,
203         'validation': validation_images,
204       }
205     return result
206
207
208 def get_image_path(image_lists, label_name, index, image_dir, category):
209     """Returns a path to an image for a label at the given index.
210
211     Args:
212       image_lists: Dictionary of training images for each label.
213       label_name: Label string we want to get an image for.
214       index: Int offset of the image we want. This will be moduloed by the
215       available number of images for the label, so it can be arbitrarily large.
216       image_dir: Root folder string of the subfolders containing the training
217       images.
218       category: Name string of set to pull images from - training, testing, or
219       validation.
220
221     Returns:
222       File system path string to an image that meets the requested parameters.
223
224     """
225     if label_name not in image_lists:
226       tf.logging.fatal('Label does not exist %s.', label_name)
227     label_lists = image_lists[label_name]
228     if category not in label_lists:
229       tf.logging.fatal('Category does not exist %s.', category)
230     category_list = label_lists[category]
231     if not category_list:
232       tf.logging.fatal('Label %s has no images in the category %s.',
233                        label_name, category)
234     mod_index = index %len(category_list)
235     base_name = category_list[mod_index]
236     sub_dir = label_lists['dir']
237     full_path = os.path.join(image_dir, sub_dir, base_name)
238     return full_path
239
240
```

```
241 def get_bottleneck_path(image_lists, label_name, index, bottleneck_dir,
242                         category, architecture):
243   """Returns a path to a bottleneck file for a label at the given index.
244
245   Args:
246     image_lists: Dictionary of training images for each label.
247     label_name: Label string we want to get an image for.
248     index: Integer offset of the image we want. This will be moduloed by th
249     available number of images for the label, so it can be arbitrarily larg
250     bottleneck_dir: Folder string holding cached files of bottleneck values
251     category: Name string of set to pull images from - training, testing, o
252     validation.
253     architecture: The name of the model architecture.
254
255   Returns:
256     File system path string to an image that meets the requested parameters
257   """
258   return get_image_path(image_lists, label_name, index, bottleneck_dir,
259                         category) + '_' + architecture + '.txt'
260
261
262 def create_model_graph(model_info):
263   """Creates a graph from saved GraphDef file and returns a Graph object.
264
265   Args:
266     model_info: Dictionary containing information about the model architect
267
268   Returns:
269     Graph holding the trained Inception network, and various tensors we'll
270     manipulating.
271   """
272   with tf.Graph().as_default() as graph:
273     model_path = os.path.join(FLAGS.model_dir, model_info['model_file_name'
274     with gfile.FastGFile(model_path, 'rb') as f:
275       graph_def = tf.GraphDef()
276       graph_def.ParseFromString(f.read())
277       bottleneck_tensor, resized_input_tensor = (tf.import_graph_def(
278           graph_def,
279           name='',
280           return_elements=[
```

```
281                 model_info['bottleneck_tensor_name'],
282                 model_info['resized_input_tensor_name'],
283             ]))
284   return graph, bottleneck_tensor, resized_input_tensor
285
286
287 def run_bottleneck_on_image(sess, image_data, image_data_tensor,
288                             decoded_image_tensor, resized_input_tensor,
289                             bottleneck_tensor):
290   """Runs inference on an image to extract the 'bottleneck' summary layer.
291
292   Args:
293     sess: Current active TensorFlow Session.
294     image_data: String of raw JPEG data.
295     image_data_tensor: Input data layer in the graph.
296     decoded_image_tensor: Output of initial image resizing and  preprocessing.
297     resized_input_tensor: The input node of the recognition graph.
298     bottleneck_tensor: Layer before the final softmax.
299
300   Returns:
301     Numpy array of bottleneck values.
302   """
303   # First decode the JPEG image, resize it, and rescale the pixel values.
304   resized_input_values = sess.run(decoded_image_tensor,
305                                   {image_data_tensor: image_data})
306   # Then run it through the recognition network.
307   bottleneck_values = sess.run(bottleneck_tensor,
308                                {resized_input_tensor: resized_input_values})
309   bottleneck_values = np.squeeze(bottleneck_values)
310   return bottleneck_values
311
312
313 def maybe_download_and_extract(data_url):
314   """Download and extract model tar file.
315
316   If the pretrained model we're using doesn't already exist, this function
317   downloads it from the TensorFlow.org website and unpacks it into a directory.
318
319   Args:
320     data_url: Web location of the tar file containing the pretrained model.
```

```
321   """
322   dest_directory = FLAGS.model_dir
323   if not os.path.exists(dest_directory):
324     os.makedirs(dest_directory)
325   filename = data_url.split('/')[-1]
326   filepath = os.path.join(dest_directory, filename)
327   if not os.path.exists(filepath):
328
329     def _progress(count, block_size, total_size):
330       sys.stdout.write('\r>> Downloading %s %.1f%% %
331                        (filename,
332                         float(count * block_size) / float(total_size) * 100
333       sys.stdout.flush()
334
335     filepath, _ = urllib.request.urlretrieve(data_url, filepath, _progress)
336     print()
337     statinfo = os.stat(filepath)
338     tf.logging.info('Successfully downloaded', filename, statinfo.st_size,
339                     'bytes.')
340   tarfile.open(filepath, 'r:gz').extractall(dest_directory)
341
342
343 def ensure_dir_exists(dir_name):
344   """Makes sure the folder exists on disk.
345
346   Args:
347     dir_name: Path string to the folder we want to create.
348   """
349   if not os.path.exists(dir_name):
350     os.makedirs(dir_name)
351
352
353 bottleneck_path_2_bottleneck_values = {}
354
355
356 def create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
357                            image_dir, category, sess, jpeg_data_tensor,
358                            decoded_image_tensor, resized_input_tensor,
359                            bottleneck_tensor):
360   """Create a single bottleneck file."""
```

```
361     tf.logging.info('Creating bottleneck at ' + bottleneck_path)
362     image_path = get_image_path(image_lists, label_name, index,
363                                 image_dir, category)
364     if not gfile.Exists(image_path):
365       tf.logging.fatal('File does not exist %s', image_path)
366     image_data = gfile.FastGFile(image_path, 'rb').read()
367     try:
368       bottleneck_values = run_bottleneck_on_image(
369           sess, image_data, jpeg_data_tensor, decoded_image_tensor,
370           resized_input_tensor, bottleneck_tensor)
371     except Exception as e:
372       raise RuntimeError('Error during processing file %s (%s)' % (image_path,
373                                                                    str(e)))
374     bottleneck_string = ','.join(str(x) for x in bottleneck_values)
375     with open(bottleneck_path, 'w') as bottleneck_file:
376       bottleneck_file.write(bottleneck_string)
377
378
379 def get_or_create_bottleneck(sess, image_lists, label_name, index, image_dir,
380                              category, bottleneck_dir, jpeg_data_tensor,
381                              decoded_image_tensor, resized_input_tensor,
382                              bottleneck_tensor, architecture):
383     """Retrieves or calculates bottleneck values for an image.
384
385     If a cached version of the bottleneck data exists on-disk, return that,
386     otherwise calculate the data and save it to disk for future use.
387
388     Args:
389       sess: The current active TensorFlow Session.
390       image_lists: Dictionary of training images for each label.
391       label_name: Label string we want to get an image for.
392       index: Integer offset of the image we want. This will be modulo-ed by the
393       available number of images for the label, so it can be arbitrarily large.
394       image_dir: Root folder string  of the subfolders containing the training
395       images.
396       category: Name string of which  set to pull images from - training, testing
397       or validation.
398       bottleneck_dir: Folder string holding cached files of bottleneck values.
399       jpeg_data_tensor: The tensor to feed loaded jpeg data into.
400       decoded_image_tensor: The output of decoding and resizing the image.
```

```
401     resized_input_tensor: The input node of the recognition graph.
402     bottleneck_tensor: The output tensor for the bottleneck values.
403     architecture: The name of the model architecture.
404
405   Returns:
406     Numpy array of values produced by the bottleneck layer for the image.
407   """
408   label_lists = image_lists[label_name]
409   sub_dir = label_lists['dir']
410   sub_dir_path = os.path.join(bottleneck_dir, sub_dir)
411   ensure_dir_exists(sub_dir_path)
412   bottleneck_path = get_bottleneck_path(image_lists, label_name, index,
413                                         bottleneck_dir, category, architect
414   if not os.path.exists(bottleneck_path):
415     create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
416                            image_dir, category, sess, jpeg_data_tensor,
417                            decoded_image_tensor, resized_input_tensor,
418                            bottleneck_tensor)
419   with open(bottleneck_path, 'r') as bottleneck_file:
420     bottleneck_string = bottleneck_file.read()
421   did_hit_error = False
422   try:
423     bottleneck_values = [float(x) for x in bottleneck_string.split(',')]
424   except ValueError:
425     tf.logging.warning('Invalid float found, recreating bottleneck')
426     did_hit_error = True
427   if did_hit_error:
428     create_bottleneck_file(bottleneck_path, image_lists, label_name, index,
429                            image_dir, category, sess, jpeg_data_tensor,
430                            decoded_image_tensor, resized_input_tensor,
431                            bottleneck_tensor)
432     with open(bottleneck_path, 'r') as bottleneck_file:
433       bottleneck_string = bottleneck_file.read()
434     # Allow exceptions to propagate here, since they shouldn't happen after
435     # fresh creation
436     bottleneck_values = [float(x) for x in bottleneck_string.split(',')]
437   return bottleneck_values
438
439
440 def cache_bottlenecks(sess, image_lists, image_dir, bottleneck_dir,
```

```
441                       jpeg_data_tensor, decoded_image_tensor,
442                       resized_input_tensor, bottleneck_tensor, architecture):
443   """Ensures all the training, testing, and validation bottlenecks are cached.
444
445   Because we're likely to read the same image multiple times (if there are no
446   distortions applied during training) it can speed things up a lot if we
447   calculate the bottleneck layer values once for each image during
448   preprocessing, and then just read those cached values repeatedly during
449   training. Here we go through all the images we've found, calculate those
450   values, and save them off.
451
452   Args:
453     sess: The current active TensorFlow Session.
454     image_lists: Dictionary of training images for each label.
455     image_dir: Root folder string of the subfolders containing the training
456     images.
457     bottleneck_dir: Folder string holding cached files of bottleneck values.
458     jpeg_data_tensor: Input tensor for jpeg data from file.
459     decoded_image_tensor: The output of decoding and resizing the image.
460     resized_input_tensor: The input node of the recognition graph.
461     bottleneck_tensor: The penultimate output layer of the graph.
462     architecture: The name of the model architecture.
463
464   Returns:
465     Nothing.
466   """
467   how_many_bottlenecks = 0
468   ensure_dir_exists(bottleneck_dir)
469   for label_name, label_lists in image_lists.items():
470     for category in ['training', 'testing', 'validation']:
471       category_list = label_lists[category]
472       for index, unused_base_name in enumerate(category_list):
473         get_or_create_bottleneck(
474             sess, image_lists, label_name, index, image_dir, category,
475             bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
476             resized_input_tensor, bottleneck_tensor, architecture)
477
478         how_many_bottlenecks += 1
479         if how_many_bottlenecks % 100 == 0:
480           tf.logging.info(
```

```
481                 str(how_many_bottlenecks) + ' bottleneck files created.')
482
483
484 def get_random_cached_bottlenecks(sess, image_lists, how_many, category,
485                                   bottleneck_dir, image_dir, jpeg_data_tens
486                                   decoded_image_tensor, resized_input_tenso
487                                   bottleneck_tensor, architecture):
488   """Retrieves bottleneck values for cached images.
489
490   If no distortions are being applied, this function can retrieve the cache
491   bottleneck values directly from disk for images. It picks a random set of
492   images from the specified category.
493
494   Args:
495     sess: Current TensorFlow Session.
496     image_lists: Dictionary of training images for each label.
497     how_many: If positive, a random sample of this size will be chosen.
498     If negative, all bottlenecks will be retrieved.
499     category: Name string of which set to pull from - training, testing, or
500     validation.
501     bottleneck_dir: Folder string holding cached files of bottleneck values
502     image_dir: Root folder string of the subfolders containing the training
503     images.
504     jpeg_data_tensor: The layer to feed jpeg image data into.
505     decoded_image_tensor: The output of decoding and resizing the image.
506     resized_input_tensor: The input node of the recognition graph.
507     bottleneck_tensor: The bottleneck output layer of the CNN graph.
508     architecture: The name of the model architecture.
509
510   Returns:
511     List of bottleneck arrays, their corresponding ground truths, and the
512     relevant filenames.
513   """
514   class_count = len(image_lists.keys())
515   bottlenecks = []
516   ground_truths = []
517   filenames = []
518   if how_many >= 0:
519     # Retrieve a random sample of bottlenecks.
520     for unused_i in range(how_many):
```

```
521         label_index = random.randrange(class_count)
522         label_name = list(image_lists.keys())[label_index]
523         image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
524         image_name = get_image_path(image_lists, label_name, image_index,
525                                     image_dir, category)
526         bottleneck = get_or_create_bottleneck(
527             sess, image_lists, label_name, image_index, image_dir, category,
528             bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
529             resized_input_tensor, bottleneck_tensor, architecture)
530         ground_truth = np.zeros(class_count, dtype=np.float32)
531         ground_truth[label_index] = 1.0
532         bottlenecks.append(bottleneck)
533         ground_truths.append(ground_truth)
534         filenames.append(image_name)
535     else:
536       # Retrieve all bottlenecks.
537       for label_index, label_name in enumerate(image_lists.keys()):
538         for image_index, image_name in enumerate(
539             image_lists[label_name][category]):
540           image_name = get_image_path(image_lists, label_name, image_index,
541                                       image_dir, category)
542           bottleneck = get_or_create_bottleneck(
543               sess, image_lists, label_name, image_index, image_dir, category,
544               bottleneck_dir, jpeg_data_tensor, decoded_image_tensor,
545               resized_input_tensor, bottleneck_tensor, architecture)
546           ground_truth = np.zeros(class_count, dtype=np.float32)
547           ground_truth[label_index] = 1.0
548           bottlenecks.append(bottleneck)
549           ground_truths.append(ground_truth)
550           filenames.append(image_name)
551     return bottlenecks, ground_truths, filenames
552
553
554 def get_random_distorted_bottlenecks(
555     sess, image_lists, how_many, category, image_dir, input_jpeg_tensor,
556     distorted_image, resized_input_tensor, bottleneck_tensor):
557   """Retrieves bottleneck values for training images, after distortions.
558
559   If we're training with distortions like crops, scales, or flips, we have to
560   recalculate the full model for every image, and so we can't use cached
```

```
561   bottleneck values. Instead we find random images for the requested catego
562   run them through the distortion graph, and then the full graph to get the
563   bottleneck results for each.
564
565   Args:
566     sess: Current TensorFlow Session.
567     image_lists: Dictionary of training images for each label.
568     how_many: The integer number of bottleneck values to return.
569     category: Name string of which set of images to fetch - training, testi
570     or validation.
571     image_dir: Root folder string of the subfolders containing the training
572     images.
573     input_jpeg_tensor: The input layer we feed the image data to.
574     distorted_image: The output node of the distortion graph.
575     resized_input_tensor: The input node of the recognition graph.
576     bottleneck_tensor: The bottleneck output layer of the CNN graph.
577
578   Returns:
579     List of bottleneck arrays and their corresponding ground truths.
580   """
581   class_count = len(image_lists.keys())
582   bottlenecks = []
583   ground_truths = []
584   for unused_i in range(how_many):
585     label_index = random.randrange(class_count)
586     label_name = list(image_lists.keys())[label_index]
587     image_index = random.randrange(MAX_NUM_IMAGES_PER_CLASS + 1)
588     image_path = get_image_path(image_lists, label_name, image_index, image
589                                 category)
590     if not gfile.Exists(image_path):
591       tf.logging.fatal('File does not exist %s', image_path)
592     jpeg_data = gfile.FastGFile(image_path, 'rb').read()
593     # Note that we materialize the distorted_image_data as a numpy array be
594     # sending running inference on the image. This involves 2 memory copies
595     # might be optimized in other implementations.
596     distorted_image_data = sess.run(distorted_image,
597                                     {input_jpeg_tensor: jpeg_data})
598     bottleneck_values = sess.run(bottleneck_tensor,
599                                  {resized_input_tensor: distorted_image_dat
600     bottleneck_values = np.squeeze(bottleneck_values)
```

```
601     ground_truth = np.zeros(class_count, dtype=np.float32)
602     ground_truth[label_index] = 1.0
603     bottlenecks.append(bottleneck_values)
604     ground_truths.append(ground_truth)
605   return bottlenecks, ground_truths
606
607
608 def should_distort_images(flip_left_right, random_crop, random_scale,
609                           random_brightness):
610   """Whether any distortions are enabled, from the input flags.
611
612   Args:
613     flip_left_right: Boolean whether to randomly mirror images horizontally.
614     random_crop: Integer percentage setting the total margin used around the
615     crop box.
616     random_scale: Integer percentage of how much to vary the scale by.
617     random_brightness: Integer range to randomly multiply the pixel values by.
618
619   Returns:
620     Boolean value indicating whether any distortions should be applied.
621   """
622   return (flip_left_right or (random_crop != 0) or (random_scale != 0) or
623           (random_brightness != 0))
624
625
626 def add_input_distortions(flip_left_right, random_crop, random_scale,
627                           random_brightness, input_width, input_height,
628                           input_depth, input_mean, input_std):
629   """Creates the operations to apply the specified distortions.
630
631   During training it can help to improve the results if we run the images
632   through simple distortions like crops, scales, and flips. These reflect the
633   kind of variations we expect in the real world, and so can help train the
634   model to cope with natural data more effectively. Here we take the supplied
635   parameters and construct a network of operations to apply them to an image.
636
637   Cropping
638   ~~~~~~~~
639
640   Cropping is done by placing a bounding box at a random position in the full
```

```
641 image. The cropping parameter controls the size of that box relative to t
642 input image. If it's zero, then the box is the same size as the input and
643 cropping is performed. If the value is 50%, then the crop box will be hal
644 width and height of the input. In a diagram it looks like this:
645
646 <          width          >
647 +-------------------------+
648 |                         |
649 |    width - crop%        |
650 |     <       >           |
651 |     +-------+           |
652 |     |       |           |
653 |     |       |           |
654 |     |       |           |
655 |     +-------+           |
656 |                         |
657 |                         |
658 +-------------------------+
659
660 Scaling
661 ~~~~~~~
662
663 Scaling is a lot like cropping, except that the bounding box is always
664 centered and its size varies randomly within the given range. For example
665 the scale percentage is zero, then the bounding box is the same size as t
666 input and no scaling is applied. If it's 50%, then the bounding box will
667 a random range between half the width and height and full size.
668
669 Args:
670   flip_left_right: Boolean whether to randomly mirror images horizontally
671   random_crop: Integer percentage setting the total margin used around th
672   crop box.
673   random_scale: Integer percentage of how much to vary the scale by.
674   random_brightness: Integer range to randomly multiply the pixel values
675   graph.
676   input_width: Horizontal size of expected input image to model.
677   input_height: Vertical size of expected input image to model.
678   input_depth: How many channels the expected input image should have.
679   input_mean: Pixel value that should be zero in the image for the graph.
680   input_std: How much to divide the pixel values by before recognition.
```

```
681
682   Returns:
683     The jpeg input layer and the distorted result tensor.
684   """
685
686   jpeg_data = tf.placeholder(tf.string, name='DistortJPGInput')
687   decoded_image = tf.image.decode_jpeg(jpeg_data, channels=input_depth)
688   decoded_image_as_float = tf.cast(decoded_image, dtype=tf.float32)
689   decoded_image_4d = tf.expand_dims(decoded_image_as_float, 0)
690   margin_scale = 1.0 + (random_crop / 100.0)
691   resize_scale = 1.0 + (random_scale / 100.0)
692   margin_scale_value = tf.constant(margin_scale)
693   resize_scale_value = tf.random_uniform(tensor_shape.scalar(),
694                                          minval=1.0,
695                                          maxval=resize_scale)
696   scale_value = tf.multiply(margin_scale_value, resize_scale_value)
697   precrop_width = tf.multiply(scale_value, input_width)
698   precrop_height = tf.multiply(scale_value, input_height)
699   precrop_shape = tf.stack([precrop_height, precrop_width])
700   precrop_shape_as_int = tf.cast(precrop_shape, dtype=tf.int32)
701   precropped_image = tf.image.resize_bilinear(decoded_image_4d,
702                                               precrop_shape_as_int)
703   precropped_image_3d = tf.squeeze(precropped_image, squeeze_dims=[0])
704   cropped_image = tf.random_crop(precropped_image_3d,
705                                  [input_height, input_width, input_depth])
706   if flip_left_right:
707     flipped_image = tf.image.random_flip_left_right(cropped_image)
708   else:
709     flipped_image = cropped_image
710   brightness_min = 1.0 - (random_brightness / 100.0)
711   brightness_max = 1.0 + (random_brightness / 100.0)
712   brightness_value = tf.random_uniform(tensor_shape.scalar(),
713                                        minval=brightness_min,
714                                        maxval=brightness_max)
715   brightened_image = tf.multiply(flipped_image, brightness_value)
716   offset_image = tf.subtract(brightened_image, input_mean)
717   mul_image = tf.multiply(offset_image, 1.0 / input_std)
718   distort_result = tf.expand_dims(mul_image, 0, name='DistortResult')
719   return jpeg_data, distort_result
720
```

```
721
722 def variable_summaries(var):
723   """Attach a lot of summaries to a Tensor (for TensorBoard visualization).
724   with tf.name_scope('summaries'):
725     mean = tf.reduce_mean(var)
726     tf.summary.scalar('mean', mean)
727     with tf.name_scope('stddev'):
728       stddev = tf.sqrt(tf.reduce_mean(tf.square(var - mean)))
729     tf.summary.scalar('stddev', stddev)
730     tf.summary.scalar('max', tf.reduce_max(var))
731     tf.summary.scalar('min', tf.reduce_min(var))
732     tf.summary.histogram('histogram', var)
733
734
735 def add_final_training_ops(class_count, final_tensor_name, bottleneck_tenso
736                            bottleneck_tensor_size):
737   """Adds a new softmax and fully-connected layer for training.
738
739   We need to retrain the top layer to identify our new classes, so this fun
740   adds the right operations to the graph, along with some variables to hold
741   weights, and then sets up all the gradients for the backward pass.
742
743   The set up for the softmax and fully-connected layers is based on:
744   https://www.tensorflow.org/versions/master/tutorials/mnist/beginners/inde
745
746   Args:
747     class_count: Integer of how many categories of things we're trying to
748     recognize.
749     final_tensor_name: Name string for the new final node that produces res
750     bottleneck_tensor: The output of the main CNN graph.
751     bottleneck_tensor_size: How many entries in the bottleneck vector.
752
753   Returns:
754     The tensors for the training and cross entropy results, and tensors for
755     bottleneck input and ground truth input.
756   """
757   with tf.name_scope('input'):
758     bottleneck_input = tf.placeholder_with_default(
759         bottleneck_tensor,
760         shape=[None, bottleneck_tensor_size],
```

```
761         name='BottleneckInputPlaceholder')
762
763     ground_truth_input = tf.placeholder(tf.float32,
764                                         [None, class_count],
765                                         name='GroundTruthInput')
766
767   # Organizing the following ops as `final_training_ops` so they're easier
768   # to see in TensorBoard
769   layer_name = 'final_training_ops'
770   with tf.name_scope(layer_name):
771     with tf.name_scope('weights'):
772       initial_value = tf.truncated_normal(
773           [bottleneck_tensor_size, class_count], stddev=0.001)
774
775       layer_weights = tf.Variable(initial_value, name='final_weights')
776
777       variable_summaries(layer_weights)
778     with tf.name_scope('biases'):
779       layer_biases = tf.Variable(tf.zeros([class_count]), name='final_biases')
780       variable_summaries(layer_biases)
781     with tf.name_scope('Wx_plus_b'):
782       logits = tf.matmul(bottleneck_input, layer_weights) + layer_biases
783       tf.summary.histogram('pre_activations', logits)
784
785   final_tensor = tf.nn.softmax(logits, name=final_tensor_name)
786   tf.summary.histogram('activations', final_tensor)
787
788   with tf.name_scope('cross_entropy'):
789     cross_entropy = tf.nn.softmax_cross_entropy_with_logits(
790         labels=ground_truth_input, logits=logits)
791     with tf.name_scope('total'):
792       cross_entropy_mean = tf.reduce_mean(cross_entropy)
793   tf.summary.scalar('cross_entropy', cross_entropy_mean)
794
795   with tf.name_scope('train'):
796     optimizer = tf.train.GradientDescentOptimizer(FLAGS.learning_rate)
797     train_step = optimizer.minimize(cross_entropy_mean)
798
799   return (train_step, cross_entropy_mean, bottleneck_input, ground_truth_input,
800           final_tensor)
```

```
801
802
803 def add_evaluation_step(result_tensor, ground_truth_tensor):
804   """Inserts the operations we need to evaluate the accuracy of our results
805
806   Args:
807     result_tensor: The new final node that produces results.
808     ground_truth_tensor: The node we feed ground truth data
809     into.
810
811   Returns:
812     Tuple of (evaluation step, prediction).
813   """
814   with tf.name_scope('accuracy'):
815     with tf.name_scope('correct_prediction'):
816       prediction = tf.argmax(result_tensor, 1)
817       correct_prediction = tf.equal(
818           prediction, tf.argmax(ground_truth_tensor, 1))
819     with tf.name_scope('accuracy'):
820       evaluation_step = tf.reduce_mean(tf.cast(correct_prediction, tf.float
821   tf.summary.scalar('accuracy', evaluation_step)
822   return evaluation_step, prediction
823
824
825 def save_graph_to_file(sess, graph, graph_file_name):
826   output_graph_def = graph_util.convert_variables_to_constants(
827       sess, graph.as_graph_def(), [FLAGS.final_tensor_name])
828   with gfile.FastGFile(graph_file_name, 'wb') as f:
829     f.write(output_graph_def.SerializeToString())
830   return
831
832
833 def prepare_file_system():
834   # Setup the directory we'll write summaries to for TensorBoard
835   if tf.gfile.Exists(FLAGS.summaries_dir):
836     tf.gfile.DeleteRecursively(FLAGS.summaries_dir)
837   tf.gfile.MakeDirs(FLAGS.summaries_dir)
838   if FLAGS.intermediate_store_frequency > 0:
839     ensure_dir_exists(FLAGS.intermediate_output_graphs_dir)
840   return
```

```
841
842
843 def create_model_info(architecture):
844   """Given the name of a model architecture, returns information about it.
845
846   There are different base image recognition pretrained models that can be
847   retrained using transfer learning, and this function translates from the name
848   of a model to the attributes that are needed to download and train with it.
849
850   Args:
851     architecture: Name of a model architecture.
852
853   Returns:
854     Dictionary of information about the model, or None if the name isn't
855     recognized
856
857   Raises:
858     ValueError: If architecture name is unknown.
859   """
860   architecture = architecture.lower()
861   if architecture == 'inception_v3':
862     # pylint: disable=line-too-long
863     data_url = 'http://download.tensorflow.org/models/image/imagenet/inception-
864     # pylint: enable=line-too-long
865     bottleneck_tensor_name = 'pool_3/_reshape:0'
866     bottleneck_tensor_size = 2048
867     input_width = 299
868     input_height = 299
869     input_depth = 3
870     resized_input_tensor_name = 'Mul:0'
871     model_file_name = 'classify_image_graph_def.pb'
872     input_mean = 128
873     input_std = 128
874   elif architecture.startswith('mobilenet_'):
875     parts = architecture.split('_')
876     if len(parts) != 3 and len(parts) != 4:
877       tf.logging.error("Couldn't understand architecture name '%s'",
878                        architecture)
879       return None
880     version_string = parts[1]
```

```
881     if (version_string != '1.0' and version_string != '0.75' and
882         version_string != '0.50' and version_string != '0.25'):
883       tf.logging.error(
884           """The Mobilenet version should be '1.0', '0.75', '0.50', or '0.
885   but found '%s' for architecture '%s'""",
886           version_string, architecture)
887       return None
888     size_string = parts[2]
889     if (size_string != '224' and size_string != '192' and
890         size_string != '160' and size_string != '128'):
891       tf.logging.error(
892           """The Mobilenet input size should be '224', '192', '160', or '12
893   but found '%s' for architecture '%s'""",
894           size_string, architecture)
895       return None
896     if len(parts) == 3:
897       is_quantized = False
898     else:
899       if parts[3] != 'quantized':
900         tf.logging.error(
901             "Couldn't understand architecture suffix '%s' for '%s'", parts[
902             architecture)
903         return None
904       is_quantized = True
905     data_url = 'http://download.tensorflow.org/models/mobilenet_v1_'
906     data_url += version_string + '_' + size_string + '_frozen.tgz'
907     bottleneck_tensor_name = 'MobilenetV1/Predictions/Reshape:0'
908     bottleneck_tensor_size = 1001
909     input_width = int(size_string)
910     input_height = int(size_string)
911     input_depth = 3
912     resized_input_tensor_name = 'input:0'
913     if is_quantized:
914       model_base_name = 'quantized_graph.pb'
915     else:
916       model_base_name = 'frozen_graph.pb'
917     model_dir_name = 'mobilenet_v1_' + version_string + '_' + size_string
918     model_file_name = os.path.join(model_dir_name, model_base_name)
919     input_mean = 127.5
920     input_std = 127.5
```

```
921   else:
922     tf.logging.error("Couldn't understand architecture name '%s'", architecture
923     raise ValueError('Unknown architecture', architecture)
924
925   return {
926       'data_url': data_url,
927       'bottleneck_tensor_name': bottleneck_tensor_name,
928       'bottleneck_tensor_size': bottleneck_tensor_size,
929       'input_width': input_width,
930       'input_height': input_height,
931       'input_depth': input_depth,
932       'resized_input_tensor_name': resized_input_tensor_name,
933       'model_file_name': model_file_name,
934       'input_mean': input_mean,
935       'input_std': input_std,
936   }
937
938
939 def add_jpeg_decoding(input_width, input_height, input_depth, input_mean,
940                       input_std):
941   """Adds operations that perform JPEG decoding and resizing to the graph..
942
943   Args:
944     input_width: Desired width of the image fed into the recognizer graph.
945     input_height: Desired width of the image fed into the recognizer graph.
946     input_depth: Desired channels of the image fed into the recognizer graph.
947     input_mean: Pixel value that should be zero in the image for the graph.
948     input_std: How much to divide the pixel values by before recognition.
949
950   Returns:
951     Tensors for the node to feed JPEG data into, and the output of the
952       preprocessing steps.
953   """
954   jpeg_data = tf.placeholder(tf.string, name='DecodeJPGInput')
955   decoded_image = tf.image.decode_jpeg(jpeg_data, channels=input_depth)
956   decoded_image_as_float = tf.cast(decoded_image, dtype=tf.float32)
957   decoded_image_4d = tf.expand_dims(decoded_image_as_float, 0)
958   resize_shape = tf.stack([input_height, input_width])
959   resize_shape_as_int = tf.cast(resize_shape, dtype=tf.int32)
960   resized_image = tf.image.resize_bilinear(decoded_image_4d,
```

```
961                                                 resize_shape_as_int)
962   offset_image = tf.subtract(resized_image, input_mean)
963   mul_image = tf.multiply(offset_image, 1.0 / input_std)
964   return jpeg_data, mul_image
965
966
967 def main(_):
968   # Needed to make sure the logging output is visible.
969   # See https://github.com/tensorflow/tensorflow/issues/3047
970   tf.logging.set_verbosity(tf.logging.INFO)
971
972   # Prepare necessary directories  that can be used during training
973   prepare_file_system()
974
975   # Gather information about the model architecture we'll be using.
976   model_info = create_model_info(FLAGS.architecture)
977   if not model_info:
978     tf.logging.error('Did not recognize architecture flag')
979     return -1
980
981   # Set up the pre-trained graph.
982   maybe_download_and_extract(model_info['data_url'])
983   graph, bottleneck_tensor, resized_image_tensor = (
984       create_model_graph(model_info))
985
986   # Look at the folder structure, and create lists of all the images.
987   image_lists = create_image_lists(FLAGS.image_dir, FLAGS.testing_percentag
988                                    FLAGS.validation_percentage)
989   class_count = len(image_lists.keys())
990   if class_count == 0:
991     tf.logging.error('No valid folders of images found at ' + FLAGS.image_d
992     return -1
993   if class_count == 1:
994     tf.logging.error('Only one valid folder of images found at ' +
995                      FLAGS.image_dir +
996                      ' - multiple classes are needed for classification.')
997     return -1
998
999   # See if the command-line flags mean we're applying any distortions.
1000  do_distort_images = should_distort_images(
```

```
1001       FLAGS.flip_left_right, FLAGS.random_crop, FLAGS.random_scale,
1002       FLAGS.random_brightness)
1003
1004   with tf.Session(graph=graph) as sess:
1005     # Set up the image decoding sub-graph.
1006     jpeg_data_tensor, decoded_image_tensor = add_jpeg_decoding(
1007         model_info['input_width'], model_info['input_height'],
1008         model_info['input_depth'], model_info['input_mean'],
1009         model_info['input_std'])
1010
1011     if do_distort_images:
1012       # We will be applying distortions, so setup the operations we'll need.
1013       (distorted_jpeg_data_tensor,
1014        distorted_image_tensor) = add_input_distortions(
1015           FLAGS.flip_left_right, FLAGS.random_crop, FLAGS.random_scale,
1016           FLAGS.random_brightness, model_info['input_width'],
1017           model_info['input_height'], model_info['input_depth'],
1018           model_info['input_mean'], model_info['input_std'])
1019     else:
1020       # We'll make sure we've calculated the 'bottleneck' image summaries and
1021       # cached them on disk.
1022       cache_bottlenecks(sess, image_lists, FLAGS.image_dir,
1023                         FLAGS.bottleneck_dir, jpeg_data_tensor,
1024                         decoded_image_tensor, resized_image_tensor,
1025                         bottleneck_tensor, FLAGS.architecture)
1026
1027     # Add the new layer that we'll be training.
1028     (train_step, cross_entropy, bottleneck_input, ground_truth_input,
1029      final_tensor) = add_final_training_ops(
1030          len(image_lists.keys()), FLAGS.final_tensor_name, bottleneck_tensor,
1031          model_info['bottleneck_tensor_size'])
1032
1033     # Create the operations we need to evaluate the accuracy of our new layer.
1034     evaluation_step, prediction = add_evaluation_step(
1035         final_tensor, ground_truth_input)
1036
1037     # Merge all the summaries and write them out to the summaries_dir
1038     merged = tf.summary.merge_all()
1039     train_writer = tf.summary.FileWriter(FLAGS.summaries_dir + '/train',
1040                                          sess.graph)
```

```
1041
1042      validation_writer = tf.summary.FileWriter(
1043          FLAGS.summaries_dir + '/validation')
1044
1045      # Set up all our weights to their initial default values.
1046      init = tf.global_variables_initializer()
1047      sess.run(init)
1048
1049      # Run the training for as many cycles as requested on the command line.
1050      for i in range(FLAGS.how_many_training_steps):
1051        # Get a batch of input bottleneck values, either calculated fresh eve
1052        # time with distortions applied, or from the cache stored on disk.
1053        if do_distort_images:
1054          (train_bottlenecks,
1055           train_ground_truth) = get_random_distorted_bottlenecks(
1056               sess, image_lists, FLAGS.train_batch_size, 'training',
1057               FLAGS.image_dir, distorted_jpeg_data_tensor,
1058               distorted_image_tensor, resized_image_tensor, bottleneck_tenso
1059        else:
1060          (train_bottlenecks,
1061           train_ground_truth, _) = get_random_cached_bottlenecks(
1062               sess, image_lists, FLAGS.train_batch_size, 'training',
1063               FLAGS.bottleneck_dir, FLAGS.image_dir, jpeg_data_tensor,
1064               decoded_image_tensor, resized_image_tensor, bottleneck_tensor,
1065               FLAGS.architecture)
1066        # Feed the bottlenecks and ground truth into the graph, and run a tra
1067        # step. Capture training summaries for TensorBoard with the `merged`
1068        train_summary, _ = sess.run(
1069            [merged, train_step],
1070            feed_dict={bottleneck_input: train_bottlenecks,
1071                       ground_truth_input: train_ground_truth})
1072        train_writer.add_summary(train_summary, i)
1073
1074        # Every so often, print out how well the graph is training.
1075        is_last_step = (i + 1 == FLAGS.how_many_training_steps)
1076        if (i %FLAGS.eval_step_interval) == 0 or is_last_step:
1077          train_accuracy, cross_entropy_value = sess.run(
1078              [evaluation_step, cross_entropy],
1079              feed_dict={bottleneck_input: train_bottlenecks,
1080                         ground_truth_input: train_ground_truth})
```

```
1081            tf.logging.info('%s: Step %d: Train accuracy = %.1f%%' %
1082                            (datetime.now(), i, train_accuracy * 100))
1083            tf.logging.info('%s: Step %d: Cross entropy = %f' %
1084                            (datetime.now(), i, cross_entropy_value))
1085            validation_bottlenecks, validation_ground_truth, _ = (
1086                get_random_cached_bottlenecks(
1087                    sess, image_lists, FLAGS.validation_batch_size, 'validation',
1088                    FLAGS.bottleneck_dir, FLAGS.image_dir, jpeg_data_tensor,
1089                    decoded_image_tensor, resized_image_tensor, bottleneck_tensor,
1090                    FLAGS.architecture))
1091            # Run a validation step and capture training summaries for TensorBoard
1092            # with the `merged` op.
1093            validation_summary, validation_accuracy = sess.run(
1094                [merged, evaluation_step],
1095                feed_dict={bottleneck_input: validation_bottlenecks,
1096                           ground_truth_input: validation_ground_truth})
1097            validation_writer.add_summary(validation_summary, i)
1098            tf.logging.info('%s: Step %d: Validation accuracy = %.1f%% (N=%d)' %
1099                            (datetime.now(), i, validation_accuracy * 100,
1100                             len(validation_bottlenecks)))
1101
1102        # Store intermediate results
1103        intermediate_frequency = FLAGS.intermediate_store_frequency
1104
1105        if (intermediate_frequency > 0 and (i % intermediate_frequency == 0)
1106            and i > 0):
1107            intermediate_file_name = (FLAGS.intermediate_output_graphs_dir +
1108                                      'intermediate_' + str(i) + '.pb')
1109            tf.logging.info('Save intermediate result to : ' +
1110                            intermediate_file_name)
1111            save_graph_to_file(sess, graph, intermediate_file_name)
1112
1113    # We've completed all our training, so run a final test evaluation on
1114    # some new images we haven't used before.
1115    test_bottlenecks, test_ground_truth, test_filenames = (
1116        get_random_cached_bottlenecks(
1117            sess, image_lists, FLAGS.test_batch_size, 'testing',
1118            FLAGS.bottleneck_dir, FLAGS.image_dir, jpeg_data_tensor,
1119            decoded_image_tensor, resized_image_tensor, bottleneck_tensor,
1120            FLAGS.architecture))
```

```
1121      test_accuracy, predictions = sess.run(
1122          [evaluation_step, prediction],
1123          feed_dict={bottleneck_input: test_bottlenecks,
1124                     ground_truth_input: test_ground_truth})
1125      tf.logging.info('Final test accuracy = %.1f%% (N=%d)' %
1126                      (test_accuracy * 100, len(test_bottlenecks)))
1127
1128      if FLAGS.print_misclassified_test_images:
1129        tf.logging.info('=== MISCLASSIFIED TEST IMAGES ===')
1130        for i, test_filename in enumerate(test_filenames):
1131          if predictions[i] != test_ground_truth[i].argmax():
1132            tf.logging.info('%70s  %s' %
1133                            (test_filename,
1134                             list(image_lists.keys())[predictions[i]]))
1135
1136      # Write out the trained graph and labels with the weights stored as
1137      # constants.
1138      save_graph_to_file(sess, graph, FLAGS.output_graph)
1139      with gfile.FastGFile(FLAGS.output_labels, 'w') as f:
1140        f.write('\n'.join(image_lists.keys()) + '\n')
1141
1142
1143  if __name__ == '__main__':
1144    parser = argparse.ArgumentParser()
1145    parser.add_argument(
1146        '--image_dir',
1147        type=str,
1148        default='',
1149        help='Path to folders of labeled images.'
1150    )
1151    parser.add_argument(
1152        '--output_graph',
1153        type=str,
1154        default='/tmp/output_graph.pb',
1155        help='Where to save the trained graph.'
1156    )
1157    parser.add_argument(
1158        '--intermediate_output_graphs_dir',
1159        type=str,
1160        default='/tmp/intermediate_graph/',
```

```
1161        help='Where to save the intermediate graphs.'
1162    )
1163    parser.add_argument(
1164        '--intermediate_store_frequency',
1165        type=int,
1166        default=0,
1167        help="""\
1168            How many steps to store intermediate graph. If "0" then will not
1169            store.\
1170        """
1171    )
1172    parser.add_argument(
1173        '--output_labels',
1174        type=str,
1175        default='/tmp/output_labels.txt',
1176        help='Where to save the trained graph\'s labels.'
1177    )
1178    parser.add_argument(
1179        '--summaries_dir',
1180        type=str,
1181        default='/tmp/retrain_logs',
1182        help='Where to save summary logs for TensorBoard.'
1183    )
1184    parser.add_argument(
1185        '--how_many_training_steps',
1186        type=int,
1187        default=4000,
1188        help='How many training steps to run before ending.'
1189    )
1190    parser.add_argument(
1191        '--learning_rate',
1192        type=float,
1193        default=0.01,
1194        help='How large a learning rate to use when training.'
1195    )
1196    parser.add_argument(
1197        '--testing_percentage',
1198        type=int,
1199        default=10,
1200        help='What percentage of images to use as a test set.'
```

```
1201    )
1202    parser.add_argument(
1203        '--validation_percentage',
1204        type=int,
1205        default=10,
1206        help='What percentage of images to use as a validation set.'
1207    )
1208    parser.add_argument(
1209        '--eval_step_interval',
1210        type=int,
1211        default=10,
1212        help='How often to evaluate the training results.'
1213    )
1214    parser.add_argument(
1215        '--train_batch_size',
1216        type=int,
1217        default=100,
1218        help='How many images to train on at a time.'
1219    )
1220    parser.add_argument(
1221        '--test_batch_size',
1222        type=int,
1223        default=-1,
1224        help="""\
1225        How many images to test on. This test set is only used once, to evalu
1226        the final accuracy of the model after training completes.
1227        A value of -1 causes the entire test set to be used, which leads to n
1228        stable results across runs.\
1229        """
1230    )
1231    parser.add_argument(
1232        '--validation_batch_size',
1233        type=int,
1234        default=100,
1235        help="""\
1236        How many images to use in an evaluation batch. This validation set is
1237        used much more often than the test set, and is an early indicator of
1238        accurate the model is during training.
1239        A value of -1 causes the entire validation set to be used, which lead
1240        more stable results across training iterations, but may be slower on
```

```
1241        training sets.\
1242        """
1243    )
1244    parser.add_argument(
1245        '--print_misclassified_test_images',
1246        default=False,
1247        help="""\
1248        Whether to print out a list of all misclassified test images.\
1249        """,
1250        action='store_true'
1251    )
1252    parser.add_argument(
1253        '--model_dir',
1254        type=str,
1255        default='/tmp/imagenet',
1256        help="""\
1257        Path to classify_image_graph_def.pb,
1258        imagenet_synset_to_human_label_map.txt, and
1259        imagenet_2012_challenge_label_map_proto.pbtxt.\
1260        """
1261    )
1262    parser.add_argument(
1263        '--bottleneck_dir',
1264        type=str,
1265        default='/tmp/bottleneck',
1266        help='Path to cache bottleneck layer values as files.'
1267    )
1268    parser.add_argument(
1269        '--final_tensor_name',
1270        type=str,
1271        default='final_result',
1272        help="""\
1273        The name of the output classification layer in the retrained graph.\
1274        """
1275    )
1276    parser.add_argument(
1277        '--flip_left_right',
1278        default=False,
1279        help="""\
1280        Whether to randomly flip half of the training images horizontally.\
```

```
1281        """,
1282        action='store_true'
1283    )
1284    parser.add_argument(
1285        '--random_crop',
1286        type=int,
1287        default=0,
1288        help="""\
1289        A percentage determining how much of a margin to randomly crop off th
1290        training images.\
1291        """
1292    )
1293    parser.add_argument(
1294        '--random_scale',
1295        type=int,
1296        default=0,
1297        help="""\
1298        A percentage determining how much to randomly scale up the size of th
1299        training images by.\
1300        """
1301    )
1302    parser.add_argument(
1303        '--random_brightness',
1304        type=int,
1305        default=0,
1306        help="""\
1307        A percentage determining how much to randomly multiply the training i
1308        input pixels up or down by.\
1309        """
1310    )
1311    parser.add_argument(
1312        '--architecture',
1313        type=str,
1314        default='inception_v3',
1315        help="""\
1316        Which model architecture to use. 'inception_v3' is the most accurate,
1317        also the slowest. For faster or smaller models, chose a MobileNet wit
1318        form 'mobilenet_<parameter size>_<input_size>[_quantized]'. For examp
1319        'mobilenet_1.0_224' will pick a model that is 17 MB in size and takes
1320        pixel input images, while 'mobilenet_0.25_128_quantized' will choose
```

```
1321        less accurate, but smaller and faster network that's 920 KB on disk and
1322        takes 128x128 images. See https://research.googleblog.com/2017/06/mobilen
1323        for more information on Mobilenet.\
1324        """)
1325    FLAGS, unparsed = parser.parse_known_args()
1326    tf.app.run(main=main, argv=[sys.argv[0]] + unparsed)
```

```
1 # Copyright 2017 The TensorFlow Authors. All Rights Reserved.
2 #
3 # Licensed under the Apache License, Version 2.0 (the "License");
4 # you may not use this file except in compliance with the License.
5 # You may obtain a copy of the License at
6 #
7 #     http://www.apache.org/licenses/LICENSE-2.0
8 #
9 # Unless required by applicable law or agreed to in writing, software
10 # distributed under the License is distributed on an "AS IS" BASIS,
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 # See the License for the specific language governing permissions and
13 # limitations under the License.
14 # ==============================================================================
15
16 from __future__ import absolute_import
17 from __future__ import division
18 from __future__ import print_function
19
20 import argparse
21 import sys
22 import time
23
24 import numpy as np
25 import tensorflow as tf
26
27 def load_graph(model_file):
28   graph = tf.Graph()
29   graph_def = tf.GraphDef()
30
31   with open(model_file, "rb") as f:
32     graph_def.ParseFromString(f.read())
33   with graph.as_default():
```

```
34      tf.import_graph_def(graph_def)
35
36    return graph
37
38  def read_tensor_from_image_file(file_name, input_height=299, input_width=29
39                                  input_mean=0, input_std=255):
40    input_name = "file_reader"
41    output_name = "normalized"
42    file_reader = tf.read_file(file_name, input_name)
43    if file_name.endswith(".png"):
44      image_reader = tf.image.decode_png(file_reader, channels = 3,
45                                         name='png_reader')
46    elif file_name.endswith(".gif"):
47      image_reader = tf.squeeze(tf.image.decode_gif(file_reader,
48                                                    name='gif_reader'))
49    elif file_name.endswith(".bmp"):
50      image_reader = tf.image.decode_bmp(file_reader, name='bmp_reader')
51    else:
52      image_reader = tf.image.decode_jpeg(file_reader, channels = 3,
53                                          name='jpeg_reader')
54    float_caster = tf.cast(image_reader, tf.float32)
55    dims_expander = tf.expand_dims(float_caster, 0);
56    resized = tf.image.resize_bilinear(dims_expander, [input_height, input_wi
57    normalized = tf.divide(tf.subtract(resized, [input_mean]), [input_std])
58    sess = tf.Session()
59    result = sess.run(normalized)
60
61    return result
62
63  def load_labels(label_file):
64    label = []
65    proto_as_ascii_lines = tf.gfile.GFile(label_file).readlines()
66    for l in proto_as_ascii_lines:
67      label.append(l.rstrip())
68    return label
69
70  if __name__ == "__main__":
71    file_name = "tf_files/lipids/Lauric_Acid/DSC_1704.jpg"
72    model_file = "tf_files/retrained_graph.pb"
73    label_file = "tf_files/retrained_labels.txt"
```

```
74  input_height = 299
75  input_width = 299
76  input_mean = 0
77  input_std = 255
78  input_layer = "Mul"
79  output_layer = "final_result"
80
81  parser = argparse.ArgumentParser()
82  parser.add_argument("--image", help="image to be processed")
83  parser.add_argument("--graph", help="graph/model to be executed")
84  parser.add_argument("--labels", help="name of file containing labels")
85  parser.add_argument("--input_height", type=int, help="input height")
86  parser.add_argument("--input_width", type=int, help="input width")
87  parser.add_argument("--input_mean", type=int, help="input mean")
88  parser.add_argument("--input_std", type=int, help="input std")
89  parser.add_argument("--input_layer", help="name of input layer")
90  parser.add_argument("--output_layer", help="name of output layer")
91  args = parser.parse_args()
92
93  if args.graph:
94    model_file = args.graph
95  if args.image:
96    file_name = args.image
97  if args.labels:
98    label_file = args.labels
99  if args.input_height:
100   input_height = args.input_height
101 if args.input_width:
102   input_width = args.input_width
103 if args.input_mean:
104   input_mean = args.input_mean
105 if args.input_std:
106   input_std = args.input_std
107 if args.input_layer:
108   input_layer = args.input_layer
109 if args.output_layer:
110   output_layer = args.output_layer
111
112 graph = load_graph(model_file)
113 t = read_tensor_from_image_file(file_name,
```

```
                                    input_height=input_height,
                                    input_width=input_width,
                                    input_mean=input_mean,
                                    input_std=input_std)

input_name = "import/" + input_layer
output_name = "import/" + output_layer
input_operation = graph.get_operation_by_name(input_name);
output_operation = graph.get_operation_by_name(output_name);

with tf.Session(graph=graph) as sess:
  start = time.time()
  results = sess.run(output_operation.outputs[0],
                     {input_operation.outputs[0]: t})
  end=time.time()
results = np.squeeze(results)

top_k = results.argsort()[-5:][::-1]
labels = load_labels(label_file)

print('\nEvaluation time (1-image): {:.3f}s\n'.format(end-start))
template = "{} (score={:0.5f})"
for i in top_k:
  print(template.format(labels[i], results[i]))
```

Bibliografía

[1] Catur Hermanto, Oscar S Opina, Marina P Natural, et al. Assessment of fungicide resistance of a population of mycosphaerella spp. on señorita banana variety (sucrier group). *Tree and Forestry Science and Biotecnology*, 4:85–90, 2010.

[2] LV Medeiros, DB Maciel, VV Medeiros, LM Houllou Kido, NT Oliveira, et al. pelb gene in isolates of colletotrichum gloeosporioides from several hosts. *Genetics and Molecular Research*, 9(2):661–673, 2010.

[3] Marta L Marulanda, Liliana Isaza, and Ana María Ramirez. Identificación de la especie de colletotrichum responsable de la antracnosis en la mora de castilla en la región cafetera. *Scientia et technica*, 13(37), 2007.

[4] S Sreenivasaprasad, K Sharada, AE Brown, and PR Mills. Pcr-based detection of colletotrichum acutatum on strawberry. *Plant pathology*, 45(4):650–655, 1996.

[5] C. V. Raman. A new radiation. *Proceedings of the Indian Academy of Sciences - Section A*, 37(3):333–341, 1928.

[6] D. C Harris and M. D. Bertolucci. *Symmetry and spectroscopy: an introduction to vibrational and electronic spectroscopy*. Courier Corporatin, 1978.

[7] R C Maher, L F Cohen, J C Gallop, E C Le Ru, and P G Etchegoin. Temperature-Dependent Anti-Stokes/Stokes Ratios under Surface-Enhanced Raman Scattering Conditions. *The Journal of Physical Chemistry B*, 110(13):6797–6803, apr 2006.

[8] Stephen J. Free. Chapter two - fungal cell wall organization and biosynthesis. volume 81 of *Advances in Genetics*, pages 33 – 82. Academic Press, 2013.

[9] Hemanth Noothalapati, Takahiro Sasaki, Tomohiro Kaino, Makoto Kawamukai, Masahiro Ando, Hiro-o Hamaguchi, and Tatsuyuki Yamamoto. Label-free Chemical Imaging of Fungal Spore Walls by Raman Microscopy and Multivariate Curve Resolution Analysis. *Scientific Reports*, 6:27789, jun 2016.

[10] Yan Ke and R. Sukthankar. Pca-sift: a more distinctive representation for local image descriptors. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II, June 2004.

[11] Daoqiang Zhang and Zhi-Hua Zhou. (2d)2pca: Two-directional two-dimensional pca for efficient face representation and recognition. *Neurocomputing*, 69(1):224 – 231, 2005. Neural Networks in Signal Processing.

[12] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *arXiv preprint arXiv:1712.00559*, 2017.

[13] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2(6), 2017.

[14] German (Portafolio) Duque. Colombia, con riesgo medio de plagas en la agricultura, 2016.

[15] The University of Sydney. Mycology - structure and function - wall composition. (n.d.).

[16] Eun-Min; Yang Sung Ik; Ganbold Erdene-Ochir; Jun Jehoon; Cho Kwang-Hwi; Lee, Cheol Min; Cho. Raman spectroscopy and density functional theory calculations of β-glucans and chitins in fungal cell walls. *Bulletin of the Korean Chemical Society*, 34(3):943–945, 2013.

[17] Almar F. Palonpon, Jun Ando, Hiroyuki Yamakoshi, Kosuke Dodo, Mikiko Sodeoka, Satoshi Kawata, and Katsumasa Fujita. Raman and sers microscopy for molecular imaging of live cells. *Nat. Protocols*, 8(4):677–692, Apr 2013.

[18] George N. Agrios. *Plant Pathology, Fifth Edition*. Academic Press, 5 edition, 2005.

[19] R. Dean, J. A. Van Kan, Z. A. Pretorius, K. E. Hammond-Kosack, A. Di Pietro, P. D. Spanu, J. J. Rudd, M. Dickman, R. Kahmann, J. Ellis, and G. D. Foster. The Top 10 fungal pathogens in molecular plant pathology. *Mol. Plant Pathol.*, 13(4):414–430, May 2012.

[20] Philipp H. Fesel and Alga Zuccaro. β-glucan: Crucial component of the fungal cell wall and elusive {MAMP} in plants. *Fungal Genetics and Biology*, 90:53 – 60, 2016.

[21] PORTAFOLIO. Colombia aumentó en 3,8 % el área de cultivos de banano. *Portafolio*, Feb 2018.

[22] Hector A Rodriguez, Esperanza Rodriguez-Arango, Juan G Morales, Gert Kema, and Rafael E Arango. Defense Gene Expression Associated with Biotrophic Phase of Mycosphaerella fijiensis M. Morelet Infection in Banana. *Plant Disease*, 100(6):1170–1175, jan 2016.

[23] José Vicente Lazo, José Alfredo Muñoz, and Aníbal Escalona. Evaluación experimental del clorotalonil en el control de la sigatoka negra (mycosphaerella fijiensis) en plantaciones de plátano (musa spp. aab). *Bioagro*, 24(2):127–134, 2012.

[24] S.P. Verma. Method of using resonance raman spectroscopy for detection of malignancy disease, May 23 1989. US Patent 4,832,483.

[25] Stefan Harmsen, Ruimin Huang, Matthew A. Wall, Hazem Karabeber, Jason M. Samii, Massimiliano Spaliviero, Julie R. White, Sebastien Monette, Rachael O'Connor, Kenneth L. Pitter, Stephen A. Sastra, Michael Saborowski, Eric C. Holland, Samuel Singer, Kenneth P. Olive, Scott W. Lowe, Ronald G. Blasberg, and Moritz F. Kircher. Surface-enhanced resonance raman scattering nanostars for high-precision cancer imaging. *Science Translational Medicine*, 7(271):271ra7–271ra7, 2015.

[26] Fa-Ke Lu, David Calligaris, Olutayo I. Olubiyi, Isaiah Norton, Wenlong Yang, Sandro Santagata, X. Sunney Xie, Alexandra J. Golby, and Nathalie Y. R. Agar. Label-free neurosurgical pathology with stimulated raman imaging. *Cancer Research*, 2016.

[27] Maciej Roman, Katarzyna M. Marzec, Ewa Grzebelus, Philipp W. Simon, Malgorzata Baranska, and Rafal Baranski. Composition and (in)homogeneity of carotenoid crystals in carrot cells revealed by high resolution raman imaging. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 136, Part C:1395 – 1400, 2015.

[28] Sneha Polisetti, Amber N. Bible, Jennifer L. Morrell-Falvey, and Paul W. Bohn. Raman chemical imaging of the rhizosphere bacterium pantoea sp. yr343 and its co-culture with arabidopsis thaliana. *Analyst*, 141:2175–2182, 2016.

[29] Petr Vitek, Katerina Novotna, Petra Hodanova, Barbora Rapantova, and Karel Klem. Detection of herbicide effects on pigment composition and psii photochemistry in helianthus annuus by raman spectroscopy and chlorophyll a fluorescence. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 170:234 – 241, 2017.

[30] Rekha Dhayakaran, Suresh Neethirajan, and Xuan Weng. Investigation of the antimicrobial activity of soy peptides by developing a high throughput drug screening assay. *Biochemistry and Biophysics Reports*, 6:149 – 157, 2016.

[31] Group Xie. Stimulated raman scattering microscopy, 2002.

[32] J.R. Lombardi and R.L. Birke. A unified approach to surface-enhanced raman spectroscopy. *Journal of Physical Chemistry C*, 112(14):5605–5617, 2008. cited By 325.

[33] P. Matousek and A.W. Parker. Bulk raman analysis of pharmaceutical tablets. *Applied Spectroscopy*, 60(12):1353–1357, 2006. cited By 90.

[34] L.D. Barron, F. Zhu, L. Hecht, G.E. Tranter, and N.W. Isaacs. Raman optical activity: An incisive probe of molecular chirality and biomolecular structure. *Journal of Molecular Structure*, 834-836(SPEC. ISS.):7–16, 2007. cited By 51.

[35] Adolf Smekal. Zur quantentheorie der dispersion. *Naturwissenschaften*, 11(43):873–875, 1923.

[36] H. A. Kramers and W. Heisenberg. Über die streuung von strahlung durch atome. *Zeitschrift für Physik*, 31(1):681–708, Feb 1925.

[37] The quantum theory of dispersion. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 114(769):710–728, 1927.

[38] G Landsberg and L Mandelstam. Eine neue erscheinung bei der lichtzerstreuung in krystallen. *Naturwissenschaften*, 16(28):557–558, 1928.

[39] Chandrasekhara Venkata Raman. A new radiation. 1928.

[40] Rajinder Singh. C. v. raman and the discovery of the raman effect. *Physics in Perspective*, 4(4):399–420, 2002.

[41] T. H. MAIMAN. Stimulated optical radiation in ruby. *Nature*, 187(4736):493–494, August 1960.

[42] Y. R. Shen and N. Bloembergen. Theory of stimulated brillouin and raman scattering. *Phys. Rev.*, 137:A1787–A1805, Mar 1965.

[43] V.V. Yakovlev, G.I. Petrov, H.F. Zhang, G.D. Noojin, M.L. Denton, R.J. Thomas, and M.O. Scully. Stimulated raman scattering: Old physics, new applications. *Journal of Modern Optics*, 56(18-19):1970–1973, 2009. cited By 14.

[44] P.F. Bernath. *Spectra of Atoms and Molecules*. Oxford University Press, 2005.

[45] Andreas C Albrecht. On the theory of raman intensities. *The Journal of Chemical Physics*, 34(5):1476–1484, 1961.

[46] Jeanne L McHale. *Molecular spectroscopy*. CRC Press, 2017.

[47] David Bohm. *Quantum theory*. Courier Corporation, 2012.

[48] Nouredine Zettili. Quantum mechanics: concepts and applications, 2003.

[49] Rodney Loudon. *The Quantum Theory of Light, 3rd ed. (Oxford Science Publications)*. Oxford University Press, USA, 3 edition, 2000.

[50] Wolfgang Kiefer. The raman effect—a unified treatment of the theory of raman scattering by molecules. derek a. long, john wiley & sons, ltd., 2002, pp 597. isbn 0-471-49028-8. *Journal of Raman Spectroscopy*, 34(2):180–180, 2003.

[51] Daniel F. Walls. Quantum theory of the raman effect. *Zeitschrift für Physik*, 237(3):224–233, 1970.

[52] Daniel F. Walls. Quantum theory of the raman effect. *Zeitschrift für Physik*, 244(2):117–128, 1971.

[53] Yariv A. *Quantum electronics*. Wiley, 3ed. edition, 1989.

[54] Shikuan Yang, Xianming Dai, Birgitt Boschitsch Stogin, and Tak-Sing Wong. Ultrasensitive surface-enhanced raman scattering detection in common fluids. *Proceedings of the National Academy of Sciences*, 113(2):268–273, 2016.

[55] Geoffrey Dent. *Modern Raman spectroscopy: a practical approach*. Wiley, 2005.

[56] Bernhard Schrader. *Infrared and Raman spectroscopy: methods and applications*. John Wiley & Sons, 2008.

[57] Horiba Jovin Yvon. Raman data and analysis raman spectroscopy for analysis and monitoring. urlhttp://www.horiba.com/fileadmin/uploads/Scientific/Documents/Raman/bands.pdf, 2018. Accessed 01-10-2019.

[58] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[59] Ragav Venkatesan. Cnn tutorial. urlhttps://tf-lenet.readthedocs.io/en/latest/tutorial/index.html, 2017. Accessed 01-14-2019.

[60] Dominik Scherer, Andreas Müller, and Sven Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks–ICANN 2010*, pages 92–101. Springer, 2010.

[61] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[62] Leonardo Araujo Santos. Artificial intelligence. urlhttps://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/, 2017. Accessed 01-14-2019.

[63] David W Hahn and Nicoló Omenetto. Laser-induced breakdown spectroscopy (libs), part ii: review of instrumental and methodological approaches to material analysis and applications to different fields. *Applied spectroscopy*, 66(4):347–419, 2012.

[64] Heinz W Siesler, Yukihiro Ozaki, Satoshi Kawata, and H Michael Heise. *Near-infrared spectroscopy: principles, instruments, applications*. John Wiley & Sons, 2008.

[65] Jorge Gastón Fernández, Fernández Baldo, Martín Alejandro, Gabriela Sansone, Viviana Calvente, Delia Aurora Benuzzi, Eloy Salinas, Julio Raba, and Maria Isabel Sanz Ferramola. Effect of temperature on the morphological characteristics of botrytis cinerea and its correlated with the genetic variability. 2014.

[66] Juan Gabriel Vanegas López, J Jesús Merlos García, and César Medardo Mayorga Abril. Flower export barriers: A comparative study in colombia, mexico and ecuador. *Latin American Business Review*, 18(3-4):227–250, 2017.

[67] Sybren Ruurds De Groot and Peter Mazur. *Non-equilibrium thermodynamics*. Courier Corporation, 2013.

[68] Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999.

[69] Jan Palach. *Parallel Programming with Python*. Packt Publishing Ltd, 2014.

[70] N Binyamini and Mina Schiffmann-Nadel. Latent infection in avocado fruit due to colletotrichum gloeosporioides. *Phytopathology*, 62(6):592–594, 1972.

[71] Thorlabs. Thorlabs laser line filter FL532-3, 2019.

[72] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[73] Max Born and Emil Wolf. *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. Elsevier, 2013.

[74] Ronald J Clarke and Anna Oprysa. Fluorescence and light scattering. *Journal of chemical education*, 81(5):705, 2004.

[75] Daniel C Harris and Michael D Bertolucci. *Symmetry and spectroscopy: an introduction to vibrational and electronic spectroscopy*. Courier Corporation, 1989.

[76] FLIR: Machine Vision. Flea3 5.0 MP Color GigE Vision (Sony ICX655), 2019.

[77] ST McHugh. Dynamic range in digital photography. *Cambridge in Colour. Available from: http://www. cambridgeincolour. com/tutorial s/dynamic-range. htm.[Accessed 24 July 2008]*, 2007.

[78] B&W Tech. B&W Tech Exemplar Datasheet, 2019.

[79] Zuojun Wei, Ruofei Pan, Yaxin Hou, Yao Yang, and Yingxin Liu. Graphene-supported pd catalyst for highly selective hydrogenation of resorcinol to 1, 3-cyclohexanedione through giant π-conjugate interactions. *Scientific reports*, 5:15664, 2015.

[80] D Sfyris, GI Sfyris, and C Galiotis. Stress intrepretation of graphene e-2g and a-1g vibrational modes: theoretical analysis. *arXiv preprint arXiv:1706.04465*, 2017.

[81] L Escobar-Alarcón, ME Espinosa-Pesqueira, DA Solis-Casados, J Gonzalo, J Solis, M Martinez-Orts, and E Haro-Poniatowski. Two-dimensional carbon nanostructures obtained by laser ablation in liquid: effect of an ultrasonic field. *Applied Physics A*, 124(2):141, 2018.

[82] R Escribano, JJ Sloan, N Siddique, N Sze, and T Dudev. Raman spectroscopy of carbon-containing particles. *Vibrational Spectroscopy*, 26(2):179–186, 2001.

[83] Michael L Ramírez-Cedeño, Natalie Gaensbauer, Hilsamar Félix-Rivera, William Ortiz-Rivera, Leonardo Pacheco-Londoño, and Samuel P Hernández-Rivera. Fiber optic coupled raman based detection of hazardous liquids concealed in commercial products. *International Journal of Spectroscopy*, 2012, 2012.

[84] Jonathan Henry W Jacobsen, Lindsay M Parker, Arun V Everest-Dass, Erik P Schartner, Georgios Tsiminis, Vasiliki Staikopoulos, Mark R Hutchinson, and Sanam Mustafa. Novel imaging tools for investigating the role of immune signalling in the brain. *Brain, behavior, and immunity*, 58:40–47, 2016.

[85] S Venkateswaran. Raman spectrum of hydrogen peroxide. *Nature*, 127(3202):406, 1931.

[86] Chi-Wah Kok and Wing-Shan Tam. Digital image.

[87] Axel J Ganzhorn, Pascale Vincendon, and John T Pelton. Structural characterization of myo-inositol monophosphatase from bovine brain by secondary structure prediction, fluorescence, circular dichroism and raman spectroscopy. *Biochimica et Biophysica Acta (BBA)-Protein Structure and Molecular Enzymology*, 1161(2-3):303–310, 1993.

[88] Dale Purves, George J Augustine, David Fitzpatrick, Lawrence C Katz, Anthony-Samuel LaMantia, James O McNamara, and S Mark Williams. Neurotransmission in the visceral motor system. In *Neuroscience. 2nd edition*. Sinauer Associates, 2001.

[89] Latha K Parthasarathy, L Ratnam, S Seelan, Carmelita Tobias, Manuel F Casanova, and Ranga N Parthasarathy. Mammalian inositol 3-phosphate synthase: its role in the biosynthesis of brain inositol and its clinical use as a psychoactive agent. In *Biology of Inositols and Phosphoinositides*, pages 293–314. Springer, 2006.

[90] Yinpeng Jin, Laura M Fayad, and Andrew F Laine. Contrast enhancement by multiscale adaptive histogram equalization. In *Wavelets: Applications in Signal and Image Processing IX*, volume 4478, pages 206–214. International Society for Optics and Photonics, 2001.

[91] Jeffrey A Hart, Stefanie Ann Lenway, and Thomas Murtha. A history of electroluminescent displays. *Indiana University*, pages 1–18, 1999.

[92] George Chikvaidze, Nina Mironova-Ulmane, A Plaude, and O Sergeev. Investigation of silicon carbide polytypes by raman spectroscopy. *Latvian Journal of Physics and Technical Sciences*, 51, 07 2014.

[93] Bruce L Bramfitt and Arlan O Benscoter. *Metallographer's guide: practice and procedures for irons and steels*. Asm International, 2001.

[94] Sujay Kumar Dutta and Dharmesh R Lodhari. *Extraction of Nuclear and Non-ferrous Metals*. Springer, 2018.

[95] FI Obahiagbon et al. A review: aspects of the african oil palm (elaeis guineesis jacq.) and the implications of its bioactives in human health. *American Journal of Biochemistry and Molecular Biology*, 2(3):106–119, 2012.

[96] JL Beare-Rogers, A Dieffenbacher, and JV Holm. Lexicon of lipid nutrition (iupac technical report). *Pure and Applied Chemistry*, 73(4):685–744, 2001.

[97] Krzysztof Czamara, Katarzyna Majzner, Marta Z Pacia, K Kochan, A Kaczor, and M Baranska. Raman spectroscopy of lipids: a review. *Journal of Raman Spectroscopy*, 46(1):4–20, 2015.

[98] Joke De Gelder, Kris De Gussem, Peter Vandenabeele, and Luc Moens. Reference database of raman spectra of biological molecules. *Journal of Raman Spectroscopy: An International Journal for Original Work in all Aspects of Raman Spectroscopy, Including Higher Order Processes, and also Brillouin and Rayleigh Scattering*, 38(9):1133–1147, 2007.

[99] Martin V Dutton and Christine S Evans. Oxalate production by fungi: its role in pathogenicity and ecology in the soil environment. *Canadian journal of microbiology*, 42(9):881–895, 1996.

[100] National Institute of Advanced Industrial Science (AIST) and Technology. Spectral Database for Organic Compounds. 2019.

[101] NA Marley, P Bennett, DR Janecky, and JS Gaffney. Spectroscopic evidence for organic diacid complexation with dissolved silica in aqueous systems—i. oxalic acid. *Organic Geochemistry*, 14(5):525–528, 1989.

[102] Ian R Lewis and Howell Edwards. *Handbook of Raman spectroscopy: from the research laboratory to the process line*. CRC Press, 2001.

[103] Lubert Stryer. Biosynthesis of membrane lipids and steroids. *Biochemistry*, 4:685–712, 1995.

[104] A Harvey Richard and R Ferrier Denise. Biochemistry lippincott's illustrated reviews, 2010.

[105] Beulah JM Rajkumar and V Ramakrishnan. Infrared and raman spectra of l-valine nitrate and l-leucine nitrate. *Journal of Raman Spectroscopy*, 31(12):1107–1112, 2000.

[106] Facanha PF Filho, PTC Freire, KCV Lima, FEA Melo, PS Pizani, et al. Raman spectra of l-leucine crystals. *arXiv preprint arXiv:0704.2792*, 2007.

[107] George Socrates. *Infrared and Raman characteristic group frequencies: tables and charts*. John Wiley & Sons, 2004.

[108] David M Carey and Gerald M Korenowski. Measurement of the raman spectrum of liquid water. *The Journal of chemical physics*, 108(7):2669–2675, 1998.

[109] Qiang Sun and Chaojian Qin. Raman oh stretching band of water as an internal standard to determine carbonate concentrations. *Chemical Geology*, 283(3-4):274–278, 2011.

[110] Marcin Wysokowski, Vasilii V Bazhenov, Mikhail V Tsurkan, Roberta Galli, Allison L Stelling, Hartmut Stöcker, Sabine Kaiser, Elke Niederschlag, Günter Gärtner, Thomas Behm, et al. Isolation and identification of chitin in three-dimensional skeleton of aplysina fistularis marine sponge. *International journal of biological macromolecules*, 62:94–100, 2013.

[111] Lucia Afanador-Kafuri, Dror Minz, Marcel Maymon, and Stanley Freeman. Characterization of colletotrichum isolates from tamarillo, passiflora, and mango in colombia and identification of a unique species from the genus. *Phytopathology*, 93(5):579–587, 2003.

[112] Wen-Hsin Chung, Wen-Chuan Chung, Mun-Tsu Peng, Hong-Ren Yang, and Jenn-Wen Huang. Specific detection of benzimidazole resistance in colletotrichum gloeosporioides from fruit crops by pcr-rflp. *New biotechnology*, 27(1):17–24, 2010.