

# Índice general

<b>1. Introducción</b>	<b>5</b>
1.1. Procesamiento digital de imágenes . . . . .	5
<b>2. Planteamiento del problema</b>	<b>6</b>
2.1. Objetivo general . . . . .	6
2.2. Objetivo específico . . . . .	6
2.3. Alcance . . . . .	6
2.4. Justificación . . . . .	6
2.5. Especificaciones técnicas . . . . .	6
<b>3. Procedimiento</b>	<b>8</b>
3.1. Lectura del dataset . . . . .	8
3.1.1. La clase CoffeeLeaf . . . . .	9
3.1.2. Lista de objetos CoffeeLeaf . . . . .	9
3.2. Procesado de la imagen . . . . .	9
3.2.1. Conversión de BGR a RGB . . . . .	9
3.2.2. Creación de las regiones de interés . . . . .	10
3.2.3. Creación de la máscara . . . . .	10
3.2.4. Enmascaramiento de las regiones de interés . . . . .	11
3.2.5. Histograma de las regiones de interés . . . . .	11
3.2.6. Segmentación de la imagen . . . . .	11
3.2.7. Clasificación de la hoja . . . . .	12
3.3. Presentación de los datos . . . . .	12
3.3.1. Resumen . . . . .	12
3.3.2. Imagen original . . . . .	12
3.3.3. Máscara . . . . .	13
3.3.4. Regiones de interés . . . . .	13
3.3.5. Imagen segmentada . . . . .	13
3.3.6. Histograma . . . . .	14
<b>4. Resultados</b>	<b>15</b>
4.1. Casos especiales . . . . .	15
4.1.1. Iluminación . . . . .	15
4.1.2. Envés de la hoja . . . . .	15

<b>5. Conclusiones</b>	<b>16</b>
5.1. Conclusiones específicas . . . . .	16
5.2. Conclusiones generales . . . . .	16
<b>Referencias</b>	<b>16</b>

# Índice de tablas

3.1. Anotaciones del dataset . . . . .	8
--	---

# Índice de códigos

1.	Cargar las anotaciones del dataset . . . . .	8
2.	La clase CoffeeLeaf . . . . .	9
3.	Lista de objetos CoffeLeaf . . . . .	10
4.	Convertir imagen BGR a RGB . . . . .	10
5.	Crear máscara . . . . .	10
6.	Enmascarar las regiones de interés . . . . .	11
7.	Cálcular histograma de la región de interés . . . . .	11
8.	Segmentar la región de interés . . . . .	11
9.	Clasificar hoja de café . . . . .	12
10.	Mostrar resumen de la clasificación . . . . .	12
11.	Mostrar imagen original . . . . .	13
12.	Mostrar máscara . . . . .	13
13.	Mostrar regiones de interés . . . . .	13
14.	Mostrar segmentación de la imagen . . . . .	14
15.	Mostrar histograma de la región de interés . . . . .	14

# Capítulo 1

## Introducción

### 1.1. Procesamiento digital de imágenes

## Capítulo 2

# Planteamiento del problema

### 2.1. Objetivo general

Demostrar las habilidades adquiridas durante el seminario *Procesamiento Digital de Imágenes: Fundamentos y Aplicaciones con GNU Octave y Open CV* aplicando los principios y técnicas básicas de manera práctica a un proyecto en particular.

### 2.2. Objetivo específico

Crear un algoritmo que clasifique hojas de café como sanas o infectadas y su nivel de afectación, y evaluar su eficiencia comparando los resultados obtenidos con los proporcionados en el conjunto de datos.

### 2.3. Alcance

A pesar de que el conjunto de datos de prueba contiene seis clasificaciones para las hojas, el algoritmo desarrollado sólo incluirá la clasificación sana y los cuatro niveles de afectación, excluyendo la clasificación *araña roja* debido a las restricciones en el tiempo del proyecto.

### 2.4. Justificación

El algoritmo y las técnicas utilizadas pueden aplicarse de manera directa en el mundo real dentro del área de la agricultura y/o agronomía, e idealmente puede servir como base para desarrollar procesos automatizados para el control de calidad en el campo del café.

### 2.5. Especificaciones técnicas

Se utilizará Python como lenguaje de programación para la implementación del algoritmo debido a su facilidad de uso y al amplio número de bibliotecas disponibles para el

procesamiento de imágenes tales como OpenCV.

## Capítulo 3

# Procedimiento

A continuación se describen los procesos del algoritmo que permiten solucionar el problema especificado. El código fuente está disponible de manera digital en la plataforma de GitHub [1].

### 3.1. Lectura del dataset

Se comienza leyendo el dataset que contiene información que ha sido etiquetada de manera manual por los autores del mismo. Este proceso se describe en Código 1.

```
import json

annotations_file = "RoCoLe.json"

with open(annotations_file, "r") as f:
    annotations = json.load(f)
```

Código 1: Cargar las anotaciones del dataset

Utilizando la biblioteca `json` de Python leemos el archivo `RoCoLe.json` (originalmente `Annotations/RoCoLe-json.json`). La variable `annotations` contiene la información necesaria para contruir nuestro conjunto de datos de prueba (véase Tabla 3.1).

Anotación	Descripción
ID	Identificador de la hoja
Label.Leaf.0.state	Estado de la hoja como saludable o infectada
Label.classification	Clasificación de la hoja o nivel de afectación
Label.Leaf.0.geometry	Puntos (x,y) que determinan el contorno de la hoja

Tabla 3.1: Anotaciones del dataset



### 3.1.1. La clase CoffeeLeaf

A continuación creamos una clase llamada `CoffeeLeaf` la cual se encarga de contener los datos proporcionados en las anotaciones y que representa a una hoja de café. Los atributos de esta clase pueden observarse en Código 2.

```
class CoffeeLeaf:
    def __init__(self, leaf_id, state, classification, image_bgr, geometry):
        self.id = leaf_id
        self.state_manual = state
        self.state_computed = None
        self.classification_manual = classification
        self.classification_computed = None
        self.image_bgr = image_bgr
        self.image_rgb = None
        self.roi_rgb = None
        self.roi_hsv = None
        self.masked_roi_rgb = None
        self.masked_roi_hue = None
        self.mask = None
        self.area = None
        self.affected_percentage = None
        self.histogram_hue = None
        self.limit_below = None
        self.limit_above = None
        self.binary = None
        self.contours = None
        self.contours_canvas = None
        self.polygon = None
        self.geometry = geometry
        self._processed = False
```

Código 2: La clase `CoffeeLeaf`

### 3.1.2. Lista de objetos `CoffeeLeaf`

## 3.2. Procesado de la imagen

### 3.2.1. Conversión de BGR a RGB

```

IMAGES_PATH = "../rocole_photos/"

coffee_leaves = []

for annotation in annotations:
    classification = annotation["Label"]["classification"]
    if classification != "red_spider_mite":
        leaf_id = annotation["ID"]
        state = annotation["Label"]["Leaf"][0]["state"]
        geometry = annotation["Label"]["Leaf"][0]["geometry"]
        leaf_image = f"{IMAGES_PATH}/{leaf_id}.jpeg"
        image_bgr = cv.imread(leaf_image)
        coffee_leaf = CoffeeLeaf(leaf_id, state,
                                classification, image_bgr, geometry)
        coffee_leaves.append(coffee_leaf)

```

Código 3: Lista de objetos CoffeLeaf

```

def _generate_image_rgb(self):
    self.image_rgb = cv.cvtColor(self.image_bgr, cv.COLOR_BGR2RGB)

```

Código 4: Convertir imagen BGR a RGB

### 3.2.2. Creación de las regiones de interés

### 3.2.3. Creación de la máscara

```

def _create_mask(self):
    self.mask = np.zeros(self.roi_hsv.shape[:2], np.uint8)
    polygon_start = self.polygon.min(axis=0)
    polygon_at_zero = self.polygon - polygon_start
    CONTOURS = -1 # All contours
    COLOR = (255, 255, 255) # White
    THICKNESS = -1 # Fill
    self.mask = cv.drawContours(
        self.mask, [polygon_at_zero], CONTOURS, COLOR, THICKNESS
    )
    self.area = cv.countNonZero(self.mask)

```

Código 5: Crear máscara

### 3.2.4. Enmascaramiento de las regiones de interés

```
def _create_masked_roi(self):
    self.masked_roi_rgb = cv.bitwise_and(self.roi_rgb,
                                         self.roi_rgb, mask=self.mask)
    self.masked_roi_hue = cv.bitwise_and(self.roi_hsv[:, :, 0], self.mask)
```

Código 6: Enmascarar las regiones de interés

### 3.2.5. Histograma de las regiones de interés

```
def _compute_histogram(self):
    hsv_normalizer = Normalize(vmin=0, vmax=179)
    self.hsv_mappable = ScalarMappable(norm=hsv_normalizer, cmap='hsv')
    self.histogram_hue = cv.calcHist([self.masked_roi_hue],
                                     [0], self.mask, [180], [0, 180])
```

Código 7: Calcular histograma de la región de interés

### 3.2.6. Segmentación de la imagen

```
def _binarize(self):
    GREEN_LIMIT_BELOW = 30
    GREEN_LIMIT_ABOVE = 120
    _, segments_below_green = cv.threshold(self.masked_roi_hue,
                                           GREEN_LIMIT_BELOW,
                                           179,
                                           cv.THRESH_BINARY)
    _, segments_above_green = cv.threshold(self.masked_roi_hue,
                                           GREEN_LIMIT_ABOVE,
                                           179,
                                           cv.THRESH_BINARY_INV)
    self.binary = cv.bitwise_and(segments_below_green,
                                 segments_above_green)
```

Código 8: Segmentar la región de interés

### 3.2.7. Clasificación de la hoja

```
def _categorize(self):
    healthy_area = cv.countNonZero(self.binary)
    affected_area = self.area - healthy_area
    self.affected_percentage = int((affected_area / self.area) * 100)
    if self.affected_percentage < 1:
        self.state_computed = "healthy"
        self.classification_computed = "healthy"
    elif self.affected_percentage < 6:
        self.state_computed = "unhealthy"
        self.classification_computed = "rust_level_1"
    elif self.affected_percentage < 21:
        self.state_computed = "unhealthy"
        self.classification_computed = "rust_level_2"
    elif self.affected_percentage < 51:
        self.state_computed = "unhealthy"
        self.classification_computed = "rust_level_3"
    else:
        self.state_computed = "unhealthy"
        self.classification_computed = "rust_level_4"
```

Código 9: Clasificar hoja de café

## 3.3. Presentación de los datos

### 3.3.1. Resumen

```
def show_summary(self):
    original_class = self.classification_manual
    new_class = self.classification_computed
    title_md = f"### {original_class} --> " \
               "{new_class} ({self.affected_percentage} %)"
    display(Markdown(title_md))
```

Código 10: Mostrar resumen de la clasificación

### 3.3.2. Imagen original

```
def show_original_image(self):
    plt.imshow(self.image_rgb)
    plt.title("Imagen Original")
    plt.axis("off")
    plt.show()
```

Código 11: Mostrar imagen original

### 3.3.3. Máscara

```
def show_mask(self):
    plt.imshow(self.mask, cmap="gray")
    plt.title("Máscara")
    plt.axis("off")
    plt.show()
```

Código 12: Mostrar máscara

### 3.3.4. Regiones de interés

```
def show_roi(self, hue=False):
    if hue:
        colorspace = "Hue"
        plt.imshow(self.masked_roi_hue, cmap="hsv")
    else:
        colorspace = "RGB"
        plt.imshow(self.masked_roi_rgb)
    plt.title(f"Región de Interés ({colorspace})")
    plt.axis("off")
    plt.show()
```

Código 13: Mostrar regiones de interés

### 3.3.5. Imagen segmentada

```
def show_binary(self):
    plt.imshow(self.binary, cmap="gray")
    plt.title(f"Segmentación")
    plt.axis("off")
    plt.show()
```

Código 14: Mostrar segmentación de la imagen

### 3.3.6. Histograma

```
def show_histogram(self):
    fig, ax = plt.subplots()
    ax.plot(self.histogram_hue)
    colorbar = plt.colorbar(self.hsv_mappable, ax=ax, location="bottom")
    colorbar.set_ticks([])
    idx_max = np.argmax(self.histogram_hue)
    plt.title(f"Histograma (Hue) Máx={idx_max}")
    plt.margins(x=0)
    plt.show()
```

Código 15: Mostrar histograma de la región de interés

## Capítulo 4

# Resultados

### 4.1. Casos especiales

#### 4.1.1. Iluminación

#### 4.1.2. Envés de la hoja

## Capítulo 5

# Conclusiones

5.1. Conclusiones específicas

5.2. Conclusiones generales



# Referencias

- [1] L. Dominguez, “Coffee Leaves Classification,” 07 2025. [Online]. Available: <https://github.com/LindermanDgz/coffee-leaves-classification>