Índice general

Índice de códigos

Capítulo 1

Introducción

1.1. Procesamiento digital de imágenes

1.2. Objetivo general

Capítulo 2

Planteamiento del problema

2.1. Objetivo del proyecto

2.2. Alcance

2.3. Justificación

2.4. Especificaciones técnicas

Capítulo 3

Procedimiento

El código fuente está disponible en la plataforma de GitHub [1].

3.1. Carga de los datos del dataset

```
import json

annotations_file="RoCoLe.json"

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

Código 1: Cargar las anotaciones del dataset

3.1.1. La clase CoffeeLeaf

3.2. Procesado de la imagen

3.2.1. Conversión de BGR a RGB

3.2.2. Creación de las regiones de interés

3.2.3. Creación de la máscara

3.2.4. Enmascaramiento de las regiones de interés

```python
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

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

Código 3: Convertir imgen BGR a RGB

3.2.5. Histograma de las regiones de interés

3.2.6. Segmentación de la imagen

3.2.7. Clasificación de la hoja

```
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 4: Crear máscara

```
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 5: Enmascarar 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 6: Cálcular histograma de la región de interés

```
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 7: Segmentar la región de interés

```
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 8: 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 9: Mostrar resumen de la clasificación

3.3.2. Imagen original

3.3.3. Máscara

3.3.4. Regiones de interés

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

Código 10: Mostrar imagen original

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

Código 11: Mostrar máscara

```
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 12: 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 13: Mostrar segmentación de la imagen

3.3.6. Histograma

```python
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 14: 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