Evidencia Final

Rodrigo Bustos Coeto A01645656

Herramientas computacionales: el arte de la programación (Gpo 201)

Problemática

El problema que elegí para resolver fue la detección de la salud de una planta de jitomates basado en sus colores, así como resaltar el fruto que ya es cosechable

Solución

1. Guardar los datos de nuestras imágenes en diferentes variables, dependiendo del modelo de color a utilizar
2. Creamos los rangos de color para segcada dato, después creamos las máscaras y creamos el mapa de dónde están esos colores segmentados
3. Sumamos los pixeles para tener el área segmentada. Después guardamos imágenes combinando los pixeles de los mapas creados y la imagen original.
4. Del numero total de pixeles de la planta calculamos su porcentaje de hojas verde, y calificamos su salud. También calculamos el porcentaje que es fruto cosechable.
5. Imprimimos los resultados del paso anterior. Creamos dos imágenes que representan el área evaluada en cada proceso y una tercera de ambos juntos
6. Ploteamos las 4 imágenes con sus títulos respectivos. La imagen original, la del área de la planta, el área del tomate y el área usada total.

Librerias

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
```

cv2 / OpenCV

Esta librería nos ayuda a la manipulación de imágenes y sus datos mediante diferentes funciones

numPy

Esta librería nos ayuda a crear arreglos para algunas funciones de la librería cv2

MatPlotLib

Esta librería nos ayuda a trazar gráficos, en este caso a dar formato a cómo mostramos las imágenes

1

Lectura de Imagen

Creamos distintas variables para guardar la imagen (archivo de entrada) en diferentes modelos de color utilizando la librería cv2

```
import cv2
import matplotlib.pyplot as plt
import numpy as np

img = cv2.imread("jitomate4.jpg")

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

img_gray_rgb = cv2.cvtColor(img_gray, cv2.COLOR_GRAY2RGB)
```

2

Creación de Máscaras

Usando la librería numPy, creamos 2 arreglos para segmentar colores, siendo ambos arreglos el rango a segmentar. En este caso use valores para el modelo HSV.

Después usamos la librería cv2 para iterar sobre las imágenes y crear una máscara de donde están estos pixeles

```
#Verde plantas saludables
lower_green = np.array([35, 10, 20])
upper_green = np.array([115, 255, 255])

#Amarillo, plantas espezando a morir
lower_yellow = np.array([20, 40, 80])
upper_yellow = np.array([35, 255, 255])

#Cafe, hojas muertas
lower_brown = np.array([10, 50, 20])
upper_brown = np.array([25, 255, 200])

#Tomates
lower_tomato = np.array([0, 0, 105])
upper_tomato = np.array([9, 255, 255])

lower_tomato2 = np.array([170, 0, 125])
upper_tomato2 = np.array([180, 255, 255])

lower_tomato3 = np.array([9, 0, 175])
upper_tomato3 = np.array([20, 255, 255])


#creacion de mascaras
maskVerde = cv2.inRange(img_hsv, lower_green, upper_green)
maskAmarilla = cv2.inRange(img_hsv, lower_yellow, upper_yellow)
maskCafe = cv2.inRange(img_hsv, lower_brown,  upper_brown)

maskTomate1 = cv2.inRange(img_hsv, lower_tomato, upper_tomato)
maskTomate2 = cv2.inRange(img_hsv, lower_tomato2, upper_tomato2)
maskTomate3 = cv2.inRange(img_hsv, lower_tomato3, upper_tomato3)
```

3

Segmentación de Color

```
#Separado de cada color
colorVerde = cv2.bitwise_and(img_rgb, img_rgb, mask=maskVerde)
colorAmarilla = cv2.bitwise_and(img_rgb, img_rgb, mask=maskAmarilla)
colorCafe = cv2.bitwise_and(img_rgb, img_rgb, mask=maskCafe)

#Separando tomates
colorTomate = cv2.bitwise_and(img_rgb, img_rgb, mask=maskTomate1)
colorTomate2 = cv2.bitwise_and(img_rgb, img_rgb, mask=maskTomate2)
colorTomate3 = cv2.bitwise_and(img_rgb, img_rgb, mask=maskTomate3)



#Suma de los pixeles en cada una de los rangos de las mascaras
areaVerde = np.sum(maskVerde > 0)
areaAmarilla = np.sum(maskAmarilla > 0)
areaCafe = np.sum(maskCafe > 0)
areaTomate1 = np.sum(maskTomate1 > 0)
areaTomate2 = np.sum(maskTomate2 > 0)
areaTomate3 = np.sum(maskTomate3 > 0)
areaTomateTotal = areaTomate1 + areaTomate2 + areaTomate3
areaTotal = areaVerde + areaAmarilla + areaCafe
```

Usando la librería cv2 aplicamos las máscaras sobre la imagen original en RGB para segmentar el área que cumplió con los rangos de parámetros de color

Contamos los pixeles Verdaderos de cada máscara para tener la suma total de los pixeles a evaluar de la planta y de los jitomates

4

Interpretación de Segmentación

Usando el área de pixeles Total, obtenemos el porcentaje de cada color en la planta y sobre el dato de pocentaje verde, le asignamos un estado

Después agregamos el Tomate al Área Total y determinamos cuánto de la planta está listo para cosechar

```
#Calculo del porcentaje
porcentajeVerde = areaVerde / areaTotal * 100
porcentajeAmarillo = areaAmarilla / areaTotal * 100
porcentajeCafe = areaCafe / areaTotal * 100


#Evaluacion de la planta
estados = ["Saludable","Posible Estres",
                    "Seca","Severamente deshidratada"]

if porcentajeVerde > 70:
    estado = estados[0]
elif porcentajeVerde > 40:
    estado = estados[1]
elif porcentajeVerde > 15:
    estado = estados[2]
else:
    estado = estados[3]

areaTotal += areaTomateTotal
porcentajeTomate = areaTomateTotal / areaTotal * 100
```

5

Impresión de Datos

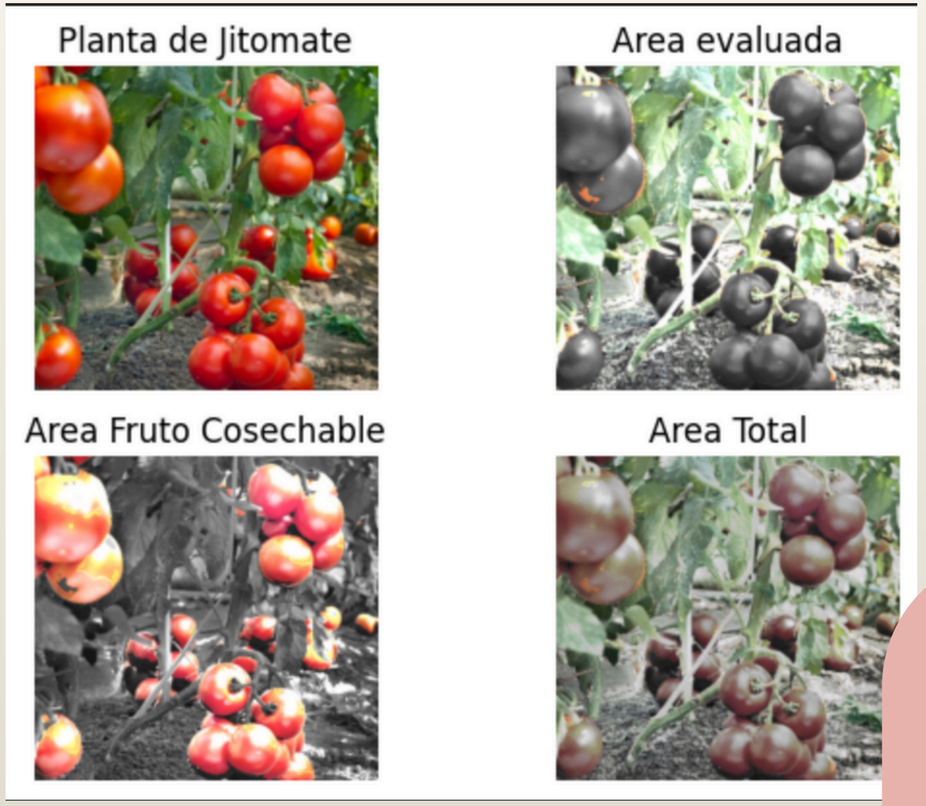
Después de interpretar los datos, los imprimimos con formato de 2 decimales junto con el estado de la planta

```python
#Impresion de resultados para la planta
print(f"Total de pixeles de plantas: --{areaTotal}-- Pixeles\n\n")
print(f"{porcentajeVerde:.2f}% de la planta es verde")
print(f"{porcentajeAmarillo:.2f}% de la planta es amarilla")
print(f"{porcentajeCafe:.2f}% de plantas esta seca\n")

print(f"El estado de la planta es: {estado}\n\n")

print(f"{porcentajeTomate:.2f}% de la planta es fruto cosechable")
```

```
Total de pixeles de plantas: --242471-- Pixeles


76.90% de la planta es verde
10.73% de la planta es amarilla
12.37% de plantas esta seca

El estado de la planta es: Saludable


35.06% de la planta es fruto cosechable
```

6

Muestra Final de Imágenes




```python
#Ploteo
plt.figure()

plt.subplot(2, 2, 1)
plt.imshow(img_rgb)
plt.axis("off")
plt.title("Planta de Jitomate")

plt.subplot(2, 2, 2)
plt.imshow(result)
plt.axis("off")
plt.title("Area evaluada")


plt.subplot(2,2,3)
plt.imshow(resultTomate)
plt.axis("off")
plt.title("Area Fruto Cosechable")

plt.subplot(2,2,4)
plt.imshow(resultTotal)
plt.axis("off")
plt.title("Area Total")
```

Finalmente usamos la librería matPlotLib para mostrar nuestras imágenes de manera ordenada, con sus títulos correspondientes

Conclusiones

Las librerías para manipulación de imágenes son muy útiles ya que permiten hacer diseñar con las funciones diferentes algoritmos para la disitintos fines

Gracias