PROYECTO DE IMAGEN

Grupo 15

Fundamentos de Sonido e Imagen

2021-2022

1. Objetivos

El objetivo principal de este proyecto es crear una serie de funciones que, a partir de una imagen de una fruta, identifique de qué tipo de fruta se trata. Nuestro banco de datos contiene seis imágenes para cada tipo de fruta: manzanas, mandarinas, fresas, plátanos y peras.

2. Funciones

2.1 Mejora de color (tarea 1)

Finalidad:

Con esta función pretendemos mejorar la imagen, maximizando el contraste con el fin de poder identificar con mayor precisión el tipo de fruta.

Sintaxis:

```
imout = MejorarColor(im)

- Parámetros de entrada:

im: imagen que queremos mejorar

- Parámetros de salida:

imout: imagen mejorada
```

Explicación:

Recibimos una imagen de tipo double (es importante que sea tipo double, pero ya nos aseguramos en tareas anteriores que la imagen recibida de este tipo) y la pasamos a escala de grises, para poder trabajar con la luminancia y sacar la los valores máximos y mínimos de esta. Uno de los problemas de calcular la luminancia máxima y mínima es que sus valores pueden estar fuera de rango, por lo que debemos forzarlos dentro del rango [0,1]. Por último devolvemos la imagen con el contraste maximizado, es decir, con la mejora de color.

Antes: **Después:**

2.2 Cálculo de la máscara (tarea 2)

Finalidad:

Obtener a la salida una máscara binaria de la imagen, donde la fruta estará a uno y el fondo a cero.

Sintaxis:

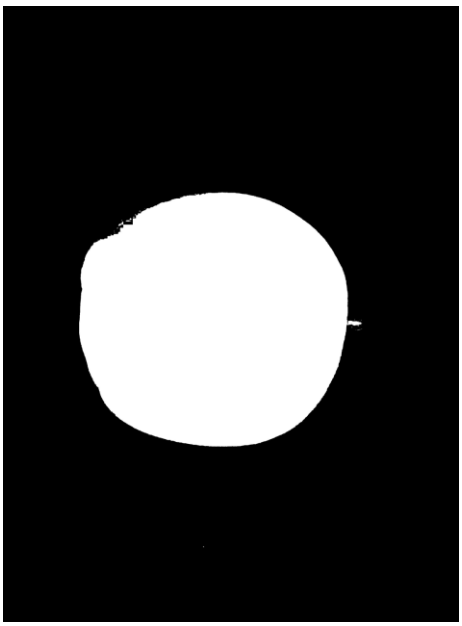
```
mascara = CalcularMascara(im)

- Parámetros de entrada:

im: imagen de la cual queremos calcular la máscara

- Parámetros de salida:

mascara: máscara de la imagen de entrada
```

Explicación:

Primero pasamos la imagen del sistema RGB al HSV (Hue, Saturacion, Value). Después calculamos un umbral que permita diferenciar qué parte de la imagen es fruta y cuál no, usando la función *graythresh*, la cual nos permite calcular un umbral que dependa de cada imagen (en lugar de utilizar un umbral a mano que nunca cambie), luego aplicamos este umbral sobre la componente S (Saturación) utilizando la función *imbinarize*, que reemplaza todos los valores que estén por encima del umbral por 1 y el resto por 0.

Por último, para mejorar la mascara resultante anterior, usamos la función *imfill*, que rellena los posibles espacios que hayan quedado eliminados debido a una alta luminancia.

Antes: **Después:**



2.3 Cálculo del vector de características (tarea 3)

Finalidad:

Función que calcula el valor medio de las componentes RGB, usando solo la parte de la imagen donde hay fruta.

Sintaxis:

```
vectorCaracteristicas = CalcularCaracteristicas(im, mascara)
```

- Parámetros de entrada:

im: imagen de la cual calcular su vector de características

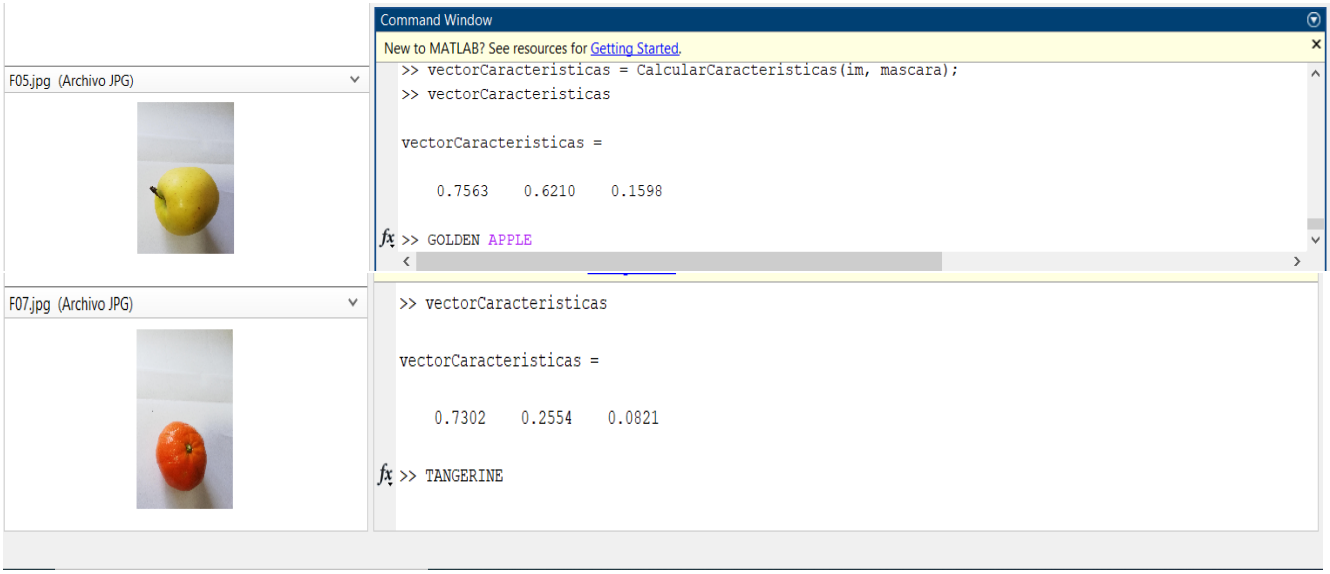
mascara: mascara de la imagen

- Parámetros de salida:

vectorCaracteristicas: vector con el valor medio de cada componente RGB

Explicación:

De la imagen que estamos analizando, sacamos sus tres componentes (RGB). A continuación hacemos la media de cada componente (RGB) aplicando la máscara, es decir, solo en las partes de la imagen donde hay fruta. Por último, con esos tres valores construimos el vector de características.



2.4 Cálculo de los prototipos (tarea 4)

Finalidad:

El objetivo de esta función es, usando la mitad de las imágenes que hay por cada tipo de fruta (tres imágenes en nuestro caso), crear un vector de características que sea el promedio de los vectores de esas tres imágenes, de manera que lo podamos usar más adelante como baremo para comprobar de qué tipo de fruta se trata.

Sintaxis:

```
prototipo = CrearPrototipo(nombre_im1, nombre_im2, nombre_im3)
```

- Parámetros de entrada:

nombre_imx: tres imágenes de la galería (del mismo tipo)

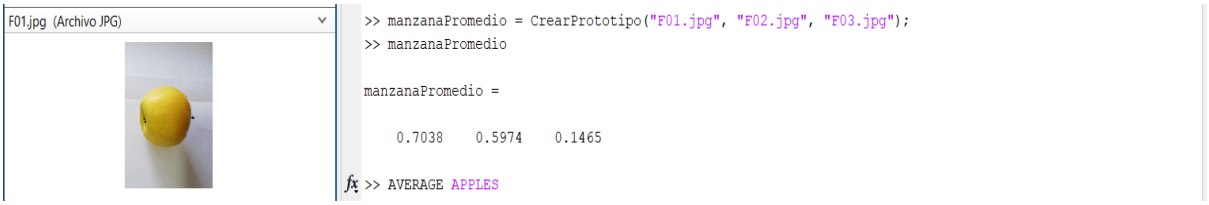
- Parámetros de salida:

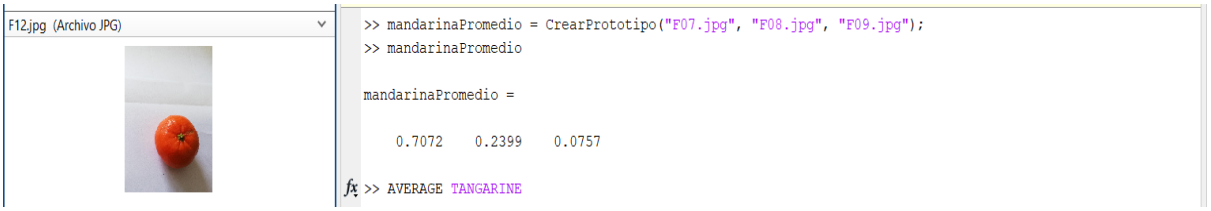
prototipo: vector de características promedio de las tres imágenes

Explicación:

Leemos cada una de las imágenes que tomamos como muestra para hacer el prototipo, y, como en los casos anteriores, trabajamos con la imagen en formato double. Para cada una de las imágenes llamamos a las tareas 1 y 2 para calcular las máscaras con las que trabajar.

Una vez tenemos las tres máscaras, llamamos a la tarea 3 para sacar tres vectores característicos, uno de cada imagen. Por último, para calcular el prototipo, promediamos estos tres vectores en uno solo.





2.5 Reconocimiento de fruta (tarea 5)

Finalidad:

Detectar el tipo de fruta.

Sintaxis:

```
clase = ReconocerClase(vectorCaracteristicas,
manzanaPromedio, mandarinaPromedio, fresaPromedio,
platanoPromedio, peraPromedio)
```

- Parámetros de entrada:

vectorCaracteristicas: vector de características del que decidir de qué clase se trata

frutaPromedio: vector de características promedio de cada tipo de fruta

- Parámetros de salida:

clase: tipo de fruta que se ha reconocido

Explicación:

Primero se calcula la distancia (norma) entre el vector dado y cada fruta promedio. Luego se comparan todas las distancias entre sí, y la menor se considerará como la correcta.

2.6 Programa principal

Finalidad:

Devuelve por pantalla el tipo de fruta de la imagen reconocida.

Sintaxis:

`clase = reconocimiento_frutas(nombre_imagen)`

`- Parámetros de entrada:`

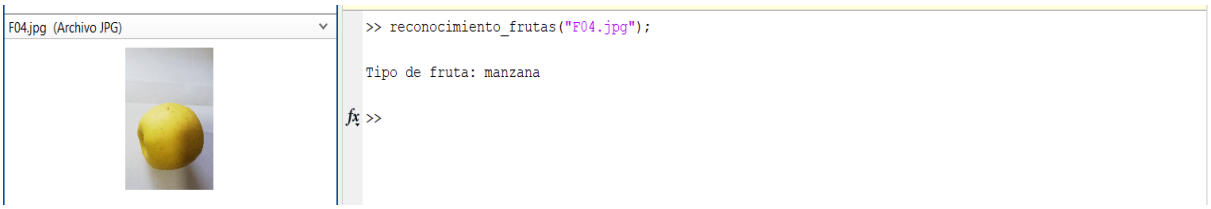
`nombre imagen: nombre del archivo (imagen) a analizar`

`- Parámetros de salida:`

`clase: tipo de fruta`

Explicación:

Primero se lee la imagen pasada como parámetro de entrada y se pasa a double (este paso es necesario como ya hemos comentado en tareas anteriores). Luego se le aplica la mejora de color (tarea 1), se calcula su máscara (tarea 2), se calcula su vector de características (tarea 3), se calcula el prototipo de cada tipo de fruta (tarea 4), se compara el vector característico con los prototipos para saber cual es el más parecido (tarea 5) y por último se muestra de qué fruta se trata.



2.7 Matriz de confusión (tarea 6)

Finalidad:

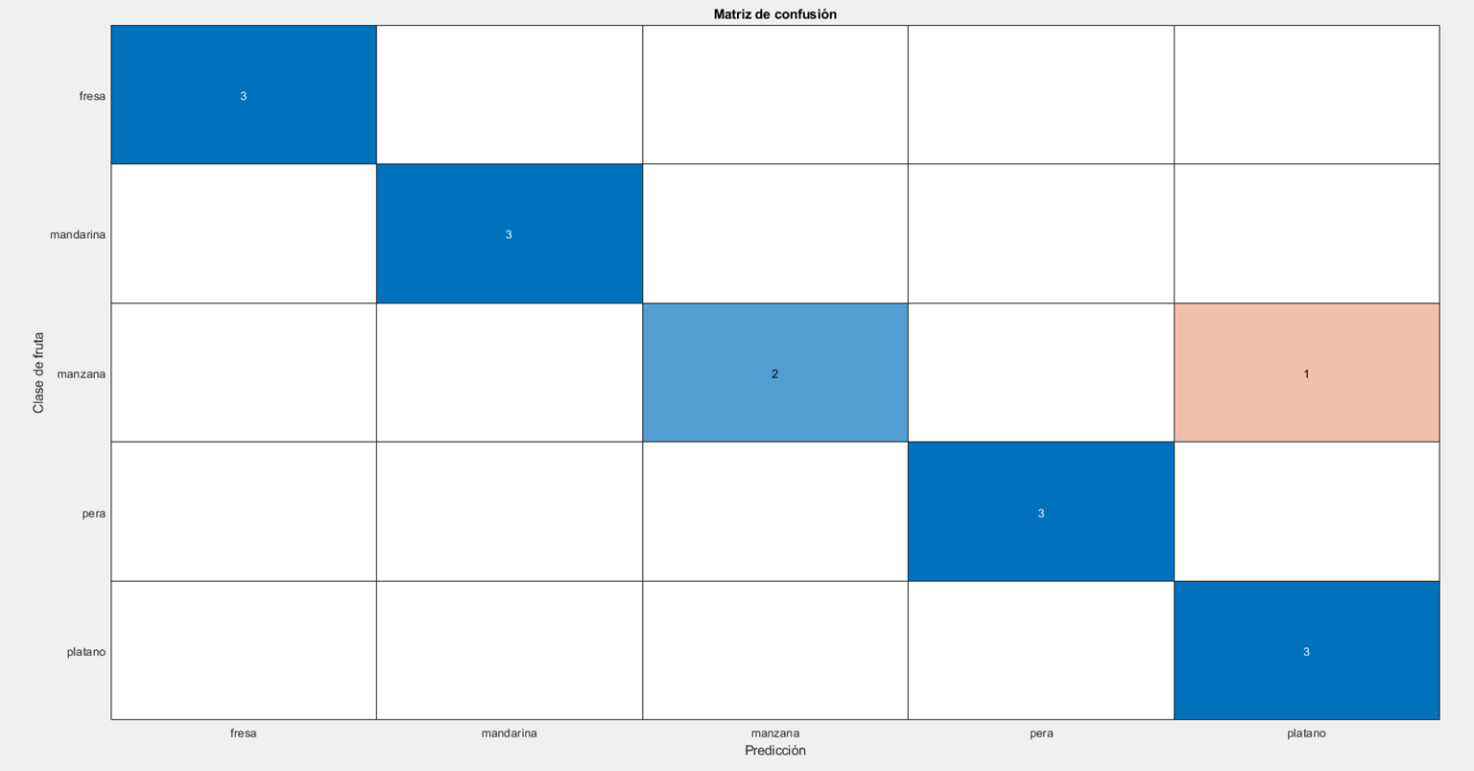
Calcula y muestra la matriz de confusión a partir de la mitad de las imágenes del banco de datos.

Sintaxis:

```
MatrizConfusion()
```

Explicación:

En primer lugar se crea un vector con los valores que se espera obtener. En nuestro caso, usamos tres imágenes para cada tipo de fruta. A continuación se crea otro vector con los valores dados por el programa principal, es decir, usamos el programa para intentar detectar de qué tipo de fruta se trata cada imagen.



Por último, usando la función *confussionchart*, calculamos la matriz de confusión y la mostramos por pantalla. En un caso de funcionamiento óptimo, la matriz debería mostrar todo ceros salvo en la diagonal principal.