Taller Nº 2 - Segmentación			
Asignatura	Tópicos Especiales - Visión Artificial	**Código**	0756
Profesor	José Carlos Rangel Ortiz		

1. Introducción

Este taller consiste en la aplicación del concepto de segmentación mediante color en OpenCV, con la intención de que los estudiantes obtengan la noción del procedimiento y también del proceso y las operaciones requeridas para la creación y aplicación máscaras utilizando imágenes en OpenCV.

Los procesos de segmentación nos permiten identificar regiones de una imagen y según esa región se puede crear una máscara que nos permitirá de manera efectiva reemplazar partes de una imagen utilizando otra.

En este taller se presentará una aplicación sencilla para reemplazar el fondo de una imagen y superponer esta sobre otra.

Al final del taller se presenta una herramienta de creación de GUI´s que se puede utilizar con OpenCV.

2. Segmentación de Color

1. Iniciamos importando las librerías

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

2. Se procede a cargar la imagen a la cual reemplazaremos el fondo

```
image = cv2.imread('ctree_bluescreen.jpg')
print('Image type: ', type(image),
      'Image Dimensions : ', image.shape)
```

3. Creamos copias de la imagen en formato RGB, para facilitar su visualización con Mat-PlotLib

```
image_copy = np.copy(image)

image_copy = cv2.cvtColor(image_copy, cv2.COLOR_BGR2RGB)
plt.imshow(image_copy)
```

4. Definimos los rangos de colores que utilizaremos

```
lower_blue = np.array([0, 0, 100]) ##[R value, G value, B value]
upper_blue = np.array([120, 100, 255])
```

5. Aplicamos la función de evaluación de los rangos

```
mask = cv2.inRange(image_copy, lower_blue, upper_blue)
plt.imshow(mask, cmap='gray')
```

6. Aplicamos la máscara a la imagen original

```
masked_image = np.copy(image_copy)
masked_image[mask != 0] = [0, 0, 0]
plt.imshow(masked_image)
```

7. Procedemos a cargar la imagen que utilizaremos de fondo y redimensionamos esta al tamaño de nuestra imagen original y aplicamos la máscara

```
background_image = cv2.imread('playa.JPG')
background_image = cv2.cvtColor(background_image, cv2.COLOR_BGR2RGB)


# Nuevo Tamaño
dsize = (500, 281)
# escalar imagen
crop_background = cv2.resize(background_image, dsize)
crop_background[mask == 0] = [0, 0, 0]

plt.imshow(crop_background)
```

8. Realizamos la suma de las imágenes y mostramos el resultado final

```
final_image = crop_background + masked_image
plt.imshow(final_image)
```

3. Segmentación en Tiempo Real

En esta sección aplicaremos los conceptos vistos en la sección anterior, pero se utilizará la cámara web como fuente para fondo nuestra imagen final.

1. Copie el siguiente código, este es similar a la primera parte del código anterior. En este caso no se hacen conversiones a RGB ya que los resultados se presentaran con OpenCV.

```
image = cv2.imread('img2.png')
print('Image type: ', type(image),
      'Image Dimensions : ', image.shape)

escala = 0.80
h,w,c = image.shape
h_final = int(h*escala)
w_final = int(w*escala)

image_copy = np.copy(image)
dsize = (w_final, h_final)
# escalar imagen
image_copy = cv2.resize(image_copy, dsize)

#plt.imshow(image_copy)

# Rangos de Color
lower_blue = np.array([100, 0, 0])
upper_blue = np.array([255, 100, 120])

# Mascara segun Rangos
mask = cv2.inRange(image_copy, lower_blue, upper_blue)
plt.imshow(mask, cmap='gray')

masked_image = np.copy(image_copy)
masked_image[mask != 0] = [0, 0, 0]
#plt.imshow(masked_image[:,:,::-1])
```

2. Esta segunda parte incluye el código para utilizar la cámara web, en la instrucción `cv2.VideoCapture(0)` definimos que cámara utilizaremos en caso de que se tengan varias conectadas al equipo.

3. Las líneas comentadas en el código funcionan para modificar el tamaño de la imagen que envía la cámara. Esto nos permite agrandar o reducir la entrada.

4. Al contrario las líneas que siguen, permiten obtener las dimensiones de la imagen que envía la cámara. En este caso se utilizan para posteriormente hacer un resize de la máscara que se agregará a la imagen de entrada, de esta manera no se modifica el asptec ratio de nuestra imagen original.

```
# Capturar desde la cámara web
video_capture = cv2.VideoCapture(0)

# Al descomentar estas lineas se puede modificar
# la resolucion de la imagen de entrada
# Definir la resolucion para la imagen de entrada
#video_capture.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
#video_capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

# Se obtiene la resolucion de la imagen de entrada
vid_width = int(video_capture.get(cv2.CAP_PROP_FRAME_WIDTH))
vid_height = int(video_capture.get(cv2.CAP_PROP_FRAME_HEIGHT))

vid_dsize = (vid_width, vid_height)

while True:
    # iniciar captura
    ret, frame = video_capture.read()
    background_image = np.copy(frame)

    # se usa el tamaño de la img de entrada
    masked_image = cv2.resize(masked_image, vid_dsize)
    mask = cv2.resize(mask, vid_dsize)

    # Aplicar mascara a la imagen de entrada
    background_image[mask == 0] = [0, 0, 0]

    # Union de fondo y montaje
    final_image = background_image + masked_image

    cv2.imshow('OpenCV Segmentacion de Color', final_image)
    # Presionar 'q' para salir
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


video_capture.release()
cv2.destroyAllWindows()
```

4. TKinter para visualizar imágenes

Tkinter es la biblioteca estándar de Python para crear interfaces gráficas de usuario. Es un conjunto de herramientas que proporciona una forma sencilla de construir aplicaciones con ventanas, botones, etiquetas, cuadros de texto, menús, etc. Tkinter viene preinstalada con Python, por lo que no es necesario instalarla por separado en la mayoría de los entornos Python.

Es conocida por su simplicidad, facilidad de uso y su capacidad para construir aplicaciones

GUI rápidamente sin la necesidad de aprender una nueva sintaxis o instalar bibliotecas adicionales. Aunque no es tan avanzada o moderna como otras bibliotecas GUI (como PyQt o Kivy), es suficiente para muchas aplicaciones de escritorio.

Dentro de los archivos del taller se encuentra un script(`Taller2_Estudiantes.py`) que realiza una interfaz con dos botones y que permite visualizar una imagen en tiempo real enviada desde la cámara o desde cualquier fuente de vídeo de OpenCV

El siguiente contiene un fragmento del código del script. En en esta porción se pueden apreciar ciertas configuraciones básicas de Tkinter.

```
# funciones que se llaman desde los botones

def open_camera():
       # Proceso para mostrar la imagen en pantalla

def cerrar_camara():
    cap.release()

# Setup de App de TKinter
app = Tk()
app.bind('<Escape>', lambda e: app.quit())
app.title("Aplicar Segmentacion por Color")

# Procesos para Centrar la ventana en la pantalla
window_width = 720
window_height = 560

# get the screen dimension
screen_width = app.winfo_screenwidth()
screen_height = app.winfo_screenheight()

# find the center point
center_x = int(screen_width/2 - window_width / 2)
center_y = int(screen_height/2 - window_height / 2)

# set the position of the window to the center of the screen
app.geometry(f'{window_width}x{window_height}+{center_x}+{center_y}')
app.resizable(False, False)

label_widget = Label(app)
label_widget.grid(row=0, column=0, columnspan=3)

button1 = Button(app, text="Open Camera", command=open_camera)
button1.grid(row=1, column=0)

button2 = Button(app, text="Close Camera", command=cerrar_camara)
button2.grid(row=1, column=1)

# Create an infinite loop for displaying app on screen
app.mainloop()
```

1. Entre la línea 3-7 se definen las funciones que se llamarán cuando se oprima algún botón en la interfaz.

```
# funciones que se llaman desde los botones

def open_camera():
       # Proceso para mostrar la imagen en pantalla

def cerrar_camara():
    cap.release()
```

2. En la línea 10 se crea una instancia de la clase TKinter, esta instancia es nuestra ventana o interfaz en la cual se agregarán los diferentes elementos que necesitamos en la ventana.

```
# Setup de App de TKinter
app = Tk()
app.bind('<Escape>', lambda e: app.quit())
app.title("Aplicar Segmentacion por Color")
```

3. Los códigos en las líneas 14-28, permiten configurar el tamaño y ubicación de nuestra ventana al iniciar el programa.

4. En la línea 30, se crea un objeto Label, el cual será el contenedor de nuestra imagen, cuando se crea un elemento de la interfaz se debe indicar a que ventana se esta agregando en este caso *app*.

```
label_widget = Label(app)
```

5. La línea 31 toma el objeto label recien creado y establece la ubicación del mismo dentro de la ventana. En este caso usando el comando *grid()* le indicamos al sistema que deseamos que nuestra ventana se divida en celdas y que se ubique este elemento en al celda *(0,0)* de la ventana. El termino *colspam* indica cuantas columnas ocupara este objeto en el grid de la ventana.

```
label_widget.grid(row=0, column=0, columnspan=3)
```

6. Las líneas 33-37 agregan botones a la interfaz y se especifica la ubicación que tendrá cada botón en la interfaz, mediante el comando *grid*. En el caso especial de los botones estos tienen el parámetro *command*, este le indica al botón que función de debe llamar cuando el boton es presionado en la interfaz.

```
button1 = Button(app, text="Open Camera", command=open_camera)
button1.grid(row=1, column=0)
```

Observación

Dentro de los archivos también encontrará el script `Taller2_Flet_Estudiantes.py` El cual utiliza la librería Flet (Web de Flet), basada en Flutter para desarrollar interfaces gráficas, las cual pueden ser también desplegadas en un explorador.
Para utilizar esta librería deberá instalar Flet en su computador, escribiendo lo siguiente en una consola:

```
pip install flet
```

Puede utilizar Tkinter o Flet para realizar la ultima entrega solicitada del taller.

Entregables del taller

1. Configure el programa para que pueda guardar la imagen final en su computadora.
2. Ejecute cada programa y suba en un documento pdf con la imagen generada para cada sección del taller.
3. Para el primer programa utilice las imágenes suministradas.
4. Para el segundo programa cree su propia imagen o máscara, buscando una imagen con transparencia y modificando su color de fondo. Coloque esta máscara sobre la salida de la cámara web o vídeo que emule el tiempo real. Puede modificar el funcionamiento y colocar su mascara solo sobre una esquina o cualquier posición de su imagen proveniente de la cámara.
5. Tomando como base el script de TKinter y el el código de Segmentación en Tiempo Real, agregue un botón al script de TKinter que active y desactive la aplicación del montaje de su imagen en la entrada de la cámara o vídeo.
6. Agregue en su informe la imagen que utilizará de máscara en su programa.