VISIÓN ARTIFICIAL

Práctica 6. Filtrado de Color con OpenCV y Python

Ingeniería en Mecatrónica
6to semestre

Mtro. Mauricio Alejandro Cabrera Arellano
Arturo Alejandro Guzman Perez 22110356 6G

Práctica 6. Filtrado de Color con OpenCV y Python

Objetivo

Implementar un filtro de color utilizando OpenCV en Python para detectar y aislar un color específico en tiempo real mediante el uso de máscaras y operaciones bit a bit.

El filtrado de color es una técnica fundamental en el procesamiento de imágenes que permite aislar objetos o regiones de una imagen basándose en su color. OpenCV facilita esta tarea mediante la conversión de espacios de color y la aplicación de máscaras.

Espacio de Color HSV

El espacio de color HSV (Hue, Saturation, Value) es más adecuado para la detección de colores que el espacio BGR, ya que separa la información de color (tono) de la intensidad (valor), lo que permite una detección más robusta bajo diferentes condiciones de iluminación.

- *Hue (Tono):* Representa el tipo de color.
- *Saturation (Saturación):* Indica la pureza del color.
- *Value (Valor):* Corresponde al brillo del color.

Máscaras y Operaciones Bit a Bit

Una máscara es una imagen binaria que indica las regiones de interés en la imagen original. Las operaciones bit a bit, como `cv2.bitwise_and()`, permiten combinar la imagen original con la máscara para resaltar o aislar las regiones deseadas.

Desarrollo de la Práctica

- *Configuración Inicial*

Se importan las bibliotecas necesarias y se inicia la captura de video desde la cámara web:

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)
```

- *Conversión a HSV y Creación de la Máscara*

Dentro de un bucle, se lee cada cuadro de video, se convierte al espacio de color HSV y se crea una máscara para el rango de color deseado:

```
while True:
    _, frame = cap.read()
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    lower_red = np.array([30, 150, 50])
    upper_red = np.array([255, 255, 180])

    mask = cv2.inRange(hsv, lower_red, upper_red)
    res = cv2.bitwise_and(frame, frame, mask=mask)
```

- *Visualización de Resultados*

Se muestran la imagen original, la máscara y el resultado del filtrado:

```
   cv2.imshow('frame', frame)
    cv2.imshow('mask', mask)
    cv2.imshow('res', res)

    k = cv2.waitKey(5) & 0xFF
    if k == 27:
        break

cv2.destroyAllWindows()
cap.release()
```

Resultados

Al ejecutar el programa, se observa en tiempo real la detección del color rojo en la imagen capturada por la cámara web. La máscara muestra en blanco las regiones donde se detecta el color rojo y en negro las demás. El resultado final muestra únicamente las áreas de la imagen original que corresponden al color rojo, mientras que el resto se oscurece.

Conclusión

La práctica permitió comprender y aplicar técnicas de filtrado de color utilizando OpenCV en Python. Se logró:

- Convertir imágenes del espacio de color BGR a HSV para una detección de color más efectiva.
- Crear máscaras para aislar colores específicos.
- Aplicar operaciones bit a bit para resaltar las regiones de interés.