2025

Reporte

PRACTICA 7

ALEXIS DANIELCORTES TAVERA

Introducción

La visión artificial es un campo de la inteligencia artificial que permite a las computadoras analizar y procesar imágenes digitales.

OpenCV es una de las bibliotecas más utilizadas para estas tareas, ofreciendo herramientas para realizar operaciones aritméticas en imágenes. Estas operaciones incluyen suma, resta, multiplicación y división de imágenes, así como transformaciones como la transpuesta, la rotación y la traslación. En esta práctica, exploramos diversas operaciones sobre imágenes utilizando OpenCV en Python.

Desarrollo Teórico y Practico

En estas prácticas se usaron diversas herramientas para su realización que serán explicadas a continuación:

librerías:

- cv2: Biblioteca OpenCV, usada para el procesamiento de imágenes, captura de video en tiempo real, creación de máscaras y aplicación de filtros morfológicos.

- numpy: Biblioteca para manejo de arreglos y cálculos numéricos, útil para crear rangos de colores en el espacio HSV y generar operaciones sobre matrices de píxeles.

Las operaciones morfológicas y lineales aplicadas sobre las máscaras de color tienen como objetivo **limpiar el ruido** y mejorar la precisión en la detección de colores. Estas son:

- **Suavizado (filtro gaussiano):** Aplica un desenfoque controlado a la máscara para reducir puntos dispersos y bordes ruidosos.
- **Apertura morfológica:** Elimina pequeñas manchas blancas aisladas en la máscara, es decir, posibles falsos positivos.

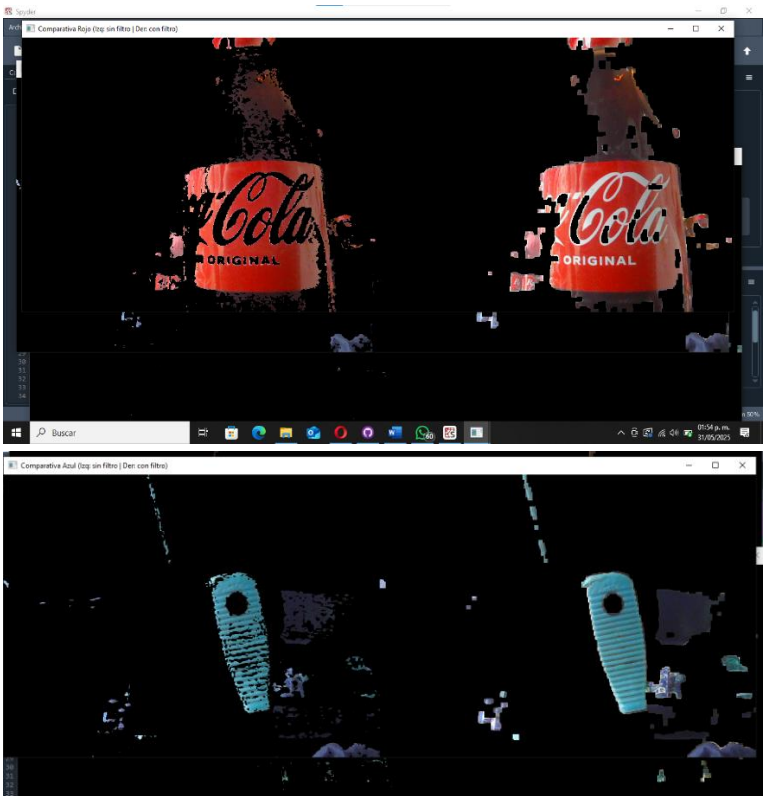
- **Cierre morfológico:** Rellena huecos oscuros dentro de objetos detectados, corrigiendo falsos negativos.

Estas técnicas permiten mejorar la segmentación y obtener resultados más estables al detectar colores específicos en tiempo real.

Practica 1:

Se reutiliza el código de la **Práctica 6**, en el cual se capturaban imágenes desde la cámara y se filtraban tres colores (rojo, verde y azul) utilizando rangos en el espacio HSV. A partir de ahí, se aplicaron las siguientes mejoras:

- A cada **máscara de color** generada se le aplicó un **filtro de remoción de ruido**, el cual consiste en una cadena de operaciones: suavizado → apertura → cierre.
- Después, se generó una imagen resultado usando cv2.bitwise_and() tanto para la **máscara original sin filtros** como para la **máscara con filtros aplicados**.
- Finalmente, se unieron las dos versiones (sin filtros y con filtros) en una **comparativa horizontal**, lo que permite visualizar claramente la mejora.



Maestro:
Mauricio Alejandro Cabrera Arellano

P3

```
# -*- coding: utf-8 -*-
"""
Created on Fri May 30 19:26:11 2025
@author: taver
"""
import cv2
import numpy as np
# Captura de video desde la cámara
cap = cv2.VideoCapture(0)
# Kernel estructurante
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))
def limpiar_mascara(mask):
    # Aplica suavizado + apertura + cierre
    mask_blur = cv2.GaussianBlur(mask, (5, 5), 0)
    apertura = cv2.morphologyEx(mask_blur, cv2.MORPH_OPEN, kernel)
    cierre = cv2.morphologyEx(apertura, cv2.MORPH_CLOSE, kernel)
    return cierre
while True:
    ret, frame = cap.read()
    if not ret:
        break
    # Convertir a HSV
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    # Rangos para rojo (dos bandas)
    lower_red1 = np.array([0, 100, 40])
```

```
    upper_red1 = np.array([10, 255, 255])
    lower_red2 = np.array([170, 60, 40])
    upper_red2 = np.array([180, 255, 255])
    mask_red = cv2.inRange(hsv, lower_red1, upper_red1) | cv2.inRange(hsv,
lower_red2, upper_red2)
    # Verde
    lower_green = np.array([36, 60, 40])
    upper_green = np.array([86, 255, 255])
    mask_green = cv2.inRange(hsv, lower_green, upper_green)
    # Azul
    lower_blue = np.array([94, 60, 2])
    upper_blue = np.array([126, 255, 255])
    mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)
    # Aplicar limpieza a las máscaras
    mask_red_clean = limpiar_mascara(mask_red)
    mask_green_clean = limpiar_mascara(mask_green)
    mask_blue_clean = limpiar_mascara(mask_blue)
    # Aplicar máscaras (sin filtro y con filtro)
    red_raw = cv2.bitwise_and(frame, frame, mask=mask_red)
    red_clean = cv2.bitwise_and(frame, frame, mask=mask_red_clean)
    green_raw = cv2.bitwise_and(frame, frame, mask=mask_green)
    green_clean = cv2.bitwise_and(frame, frame, mask=mask_green_clean)
    blue_raw = cv2.bitwise_and(frame, frame, mask=mask_blue)
    blue_clean = cv2.bitwise_and(frame, frame, mask=mask_blue_clean)
    # Comparativas por color (lado a lado)
    comp_red = np.hstack((red_raw, red_clean))
    comp_green = np.hstack((green_raw, green_clean))
```

```
    comp_blue = np.hstack((blue_raw, blue_clean))
    # Mostrar comparativas finales
    cv2.imshow('Comparativa Rojo (Izq: sin filtro | Der: con filtro)', comp_red)
    cv2.imshow('Comparativa Verde (Izq: sin filtro | Der: con filtro)', comp_green)
    cv2.imshow('Comparativa Azul (Izq: sin filtro | Der: con filtro)', comp_blue)
    # Salir con 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Conclusión

Esta práctica resultó bastante útil para reforzar lo aprendido en la detección de colores, ya que permitió mejorar los resultados aplicando técnicas de limpieza que son muy utilizadas en procesamiento de imágenes. El hecho de que se pueda reducir el ruido con unas pocas líneas de código demuestra lo potente que es OpenCV, especialmente al trabajar con video en tiempo real.

Además, aplicar operaciones como el suavizado, la apertura y el cierre ayuda a entender cómo se puede mejorar la precisión de los sistemas de visión artificial en situaciones reales, donde siempre habrá ruido o interferencias. Lo mejor es que su implementación es sencilla, y con un poco de lógica se pueden lograr resultados mucho más limpios y profesionales.

Sin duda, este tipo de prácticas no solo ayudan a entender los conceptos, sino que también muestran lo cercanas que están estas tecnologías a aplicaciones actuales como cámaras inteligentes, robots móviles o sistemas de seguimiento

Referencias:

Ibm. (2025, 6 enero). Computer Vision. *Vision Artificial*. https://www.ibm.com/mx-es/topics/computer-vision