CENTRO DE ENSEÑANZA TÉCNICA INDUSTRIAL

INGENIERÍA EN MECATRÓNICA

Visión artificial

Práctica 7:

https://github.com/thvvad/Vision-2025/tree/770a02db3b3aca8d6172de275ae2849252790cf0/PRACTICA_7_Ruido

Alumna:

Vanessa Aguirre Diaz

Registro:

22310274

Fecha:

27 de mayo del 2025

Objetivo:

Remover ruido lineal y morfológicamente de las imágenes obtenidas con los filtros de color de la practica 6, haciendo uso de operaciones de apertura y cierre, además de las funciones TopHat y BlackHat para obtener una imagen más limpia del objeto que deseamos detectar.

Desarrollo teórico:

En visión artificial, **remover ruido** es uno de los pasos esenciales para mejorar la calidad de una imagen antes del análisis. Existen varias técnicas para hacerlo, y pueden clasificarse en dos grandes grupos:

Remoción de ruido de forma lineal

Estas técnicas utilizan operaciones matemáticas sobre los píxeles basadas en sus valores y los de sus vecinos. Son buenas para reducir ruido aleatorio sin alterar mucho la estructura general de la imagen.

a) Filtro Promedio (Mean filter)

- Reemplaza cada píxel con el promedio de sus vecinos.
- Suaviza la imagen, pero puede emborronar bordes.

```
blurred = cv2.blur(img, (3, 3))  # Usa una máscara de 3x3
```

b) Filtro Gaussiano

- Similar al promedio, pero da más peso a los píxeles cercanos al centro.
- Preserva mejor los bordes que el filtro promedio.

```
blurred = cv2.GaussianBlur(img, (3, 3), 0)
```

c) Filtro de Mediana

- Reemplaza cada píxel con la mediana de sus vecinos.
- Muy eficaz contra el **ruido sal y pimienta**.

```
median = cv2.medianBlur(img, 3)
```

Remoción de ruido morfológica

Las operaciones morfológicas tratan las formas en la imagen, principalmente en imágenes binarias (aunque pueden extenderse a escala de grises). Se basan en un elemento estructurante.

a) Erosión

- Elimina píxeles en los bordes de los objetos.
- Útil para quitar puntos blancos aislados (ruido blanco).

kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))

eroded = cv2.erode(binary_img, kernel, iterations=1)

b) Dilatación

- Añade píxeles a los bordes de los objetos.
- Útil para cerrar huecos negros pequeños.

dilated = cv2.dilate(binary_img, kernel, iterations=1)

c) Apertura

- Erosión seguida de dilatación.
- Elimina pequeños objetos (ruido) sin afectar la forma principal.

opened = cv2.morphologyEx(binary_img, cv2.MORPH_OPEN, kernel)

d) Cierre

- Dilatación seguida de erosión.
- Rellena pequeños huecos o puntos negros en objetos blancos.

closed = cv2.morphologyEx(binary_img, cv2.MORPH_CLOSE, kernel)

¿Cuándo usar cada uno?

Tipo de Ruido	Mejor Técnica
Ruido aleatorio (gaussiano)	Filtro Gaussiano o Promedio
Ruido sal y pimienta	Filtro de Mediana o Apertura
Puntos blancos pequeños	Erosión o Apertura
Huecos negros pequeños	Dilatación o Cierre

Ruido F+ y F-

- **F^+ (Falsos Positivos):** se detecta algo que no está presente en realidad.
- **F^- (Falsos Negativos):** no se detecta algo que sí está presente.

¿Qué significa "remover ruido de la detección F^+ y F^-"?

Se refiere a reducir los errores en la detección o clasificación automática que generan resultados incorrectos debido a:

- Ruido visual en la imagen (por ejemplo, sombras, desenfoque, cambios de iluminación).
- Errores del algoritmo (por ejemplo, mal entrenamiento del modelo).
- Datos mal etiquetados o insuficientes.

Por tanto, remover este “ruido” no implica procesar la imagen directamente, sino ajustar los métodos de detección para mejorar la precisión.

TopHat (cv2.MORPH_TOPHAT)

- **¿Qué hace?**
  Calcula la diferencia entre la imagen original y su apertura.
- **¿Qué resalta?**
  Resalta objetos o detalles más pequeños y brillantes (zonas blancas pequeñas que no forman parte de estructuras grandes).
- **¿Cuándo usarlo?**
  - Cuando quieres detectar pequeños detalles blancos que sobresalen del fondo.
  - Para eliminar el fondo uniforme y mantener solo los objetos claros más pequeños.
  - Por ejemplo: detectar texto blanco, defectos, manchas brillantes, pequeñas piezas, etc.

BlackHat (cv2.MORPH_BLACKHAT)

- **¿Qué hace?**
  Calcula la diferencia entre el cierre de la imagen y la imagen original.
- **¿Qué resalta?**
  Resalta detalles pequeños y oscuros en un fondo más claro.
- **¿Cuándo usarlo?**
  - Cuando necesitas ver huecos negros, sombras pequeñas, imperfecciones oscuras.
  - Útil para encontrar grietas, rayones, letras oscuras en fondo claro, etc.

Beneficio	TopHat	BlackHat
Detectar detalles pequeños blancos	✅	❌
Detectar detalles pequeños negros	❌	✅
Eliminar fondo uniforme	✅	✅
Mejorar preprocesamiento para OCR (reconocimiento de texto)	✅	✅
Útiles antes de aplicar contornos o detección de bordes	✅	✅
Ayudan a segmentar imágenes complicadas	✅	✅

Apertura MORPH_OPEN)

La apertura es una operación morfológica que:

1. Elimina ruido blanco pequeño (píxeles aislados de fondo claro).
2. Se compone de:
   ➤ **Erosión** → encoge las regiones blancas.
   ➤ **Dilatación** → las vuelve a expandir.

Útil para eliminar puntitos blancos sueltos que no pertenecen al objeto.

Cierre (MORPH_CLOSE)

La cierre es otra operación morfológica que:

1. Rellena huecos negros pequeños dentro de zonas blancas.
2. Se compone de:
   ➤ **Dilatación** → expande las regiones blancas.
   ➤ **Erosión** → las contrae después.

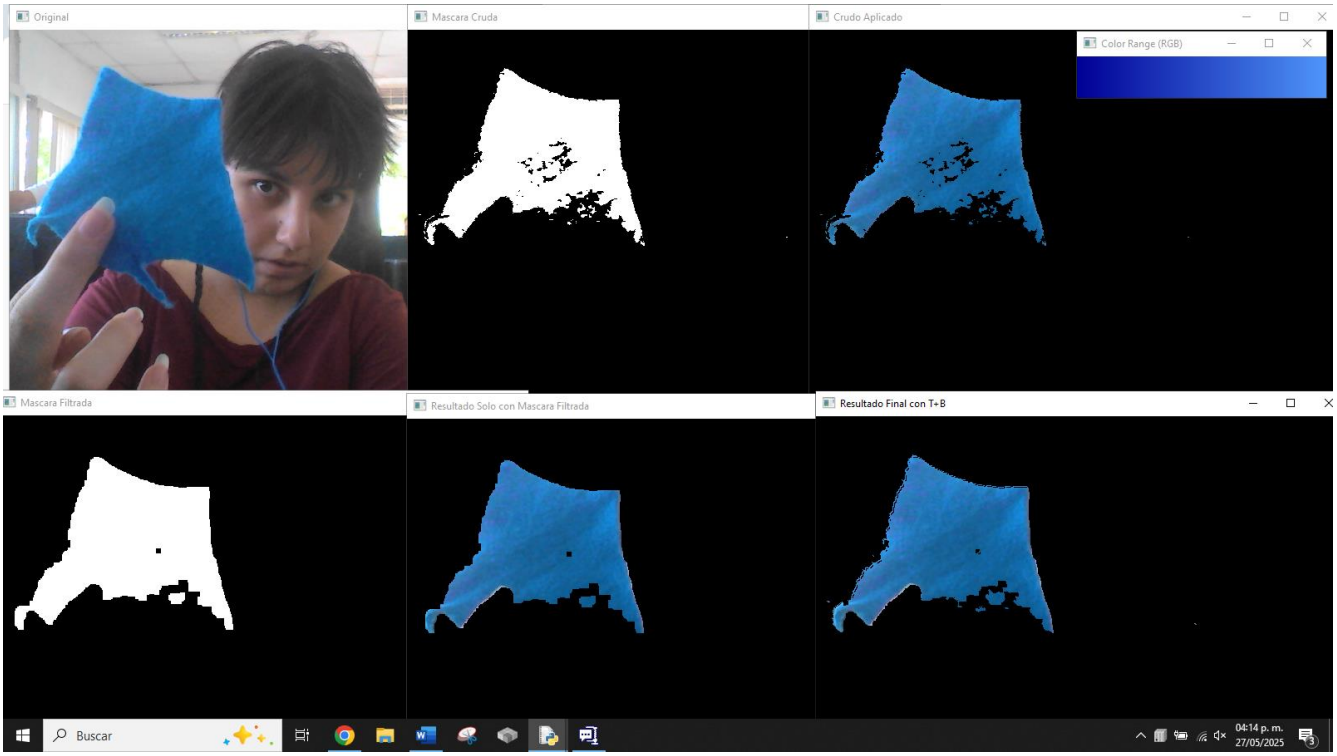
Útil para unir regiones cortadas o completar el objeto detectado.

Desarrollo práctico

Para esta practica se tomo como base los resultados obtenidos en la practica 6 de filtros de colores y se integraron operaciones morfológicas para mejorar la detección del objeto y eliminar ruido del fondo y dentro del mismo objeto.

En la primera ventana se ve el frame de la cámara de video, en la segunda se ve la máscara del filtro del color y en la tercera ventana la imagen de la cámara al aplicar el filtro, dejando ver solo los pixeles azules que entran dentro del rango.

Inicialmente en la práctica, se hicieron primero las operaciones de apertura y cierre y después las de TopHat y BlackHat (las cuales se aplican a la máscara), obteniendo el siguiente resultado:



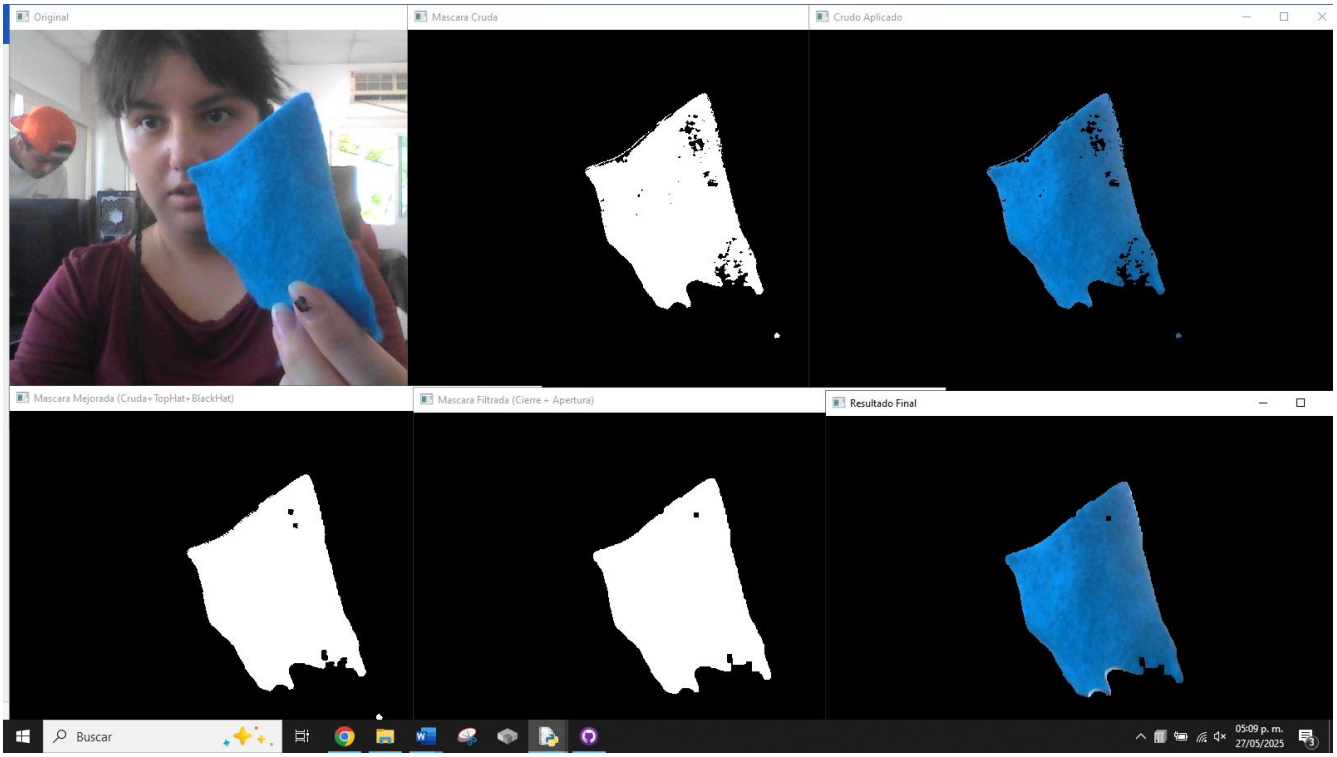
Con estas operaciones podemos ver que se eliminan pixeles azules que se encuentran en el fondo. Estos son detectados por el filtro de color por el simple hecho de ser azules, pero pueden llegar a interferir en el propósito de nuestro programa, por lo que debemos eliminarlos. Además, no solo eliminamos el ruido externo al objeto sino también el ruido negro ocasionado por sombras en el relieve del objeto que hacen que ciertos pixeles dentro del objeto se salgan levemente del rango de color.

Obtuve que con este orden en las operaciones se veía los bordes desconectados en algunas partes, por lo que primero se sumaron los bordes obtenidos de las operaciones TopHat y BlackHat y después se realizaron las operaciones de apertura y cierre, ya que son estas operaciones la que integran los pixeles sueltos cercanos a la imagen principal que pueden corresponder a la figura y borran aquellos dispersos en el resto de la imagen que pueden ser considerados ruido.

Código final mejorado (TopHat y BlackHat primero, cierre y apertura después)

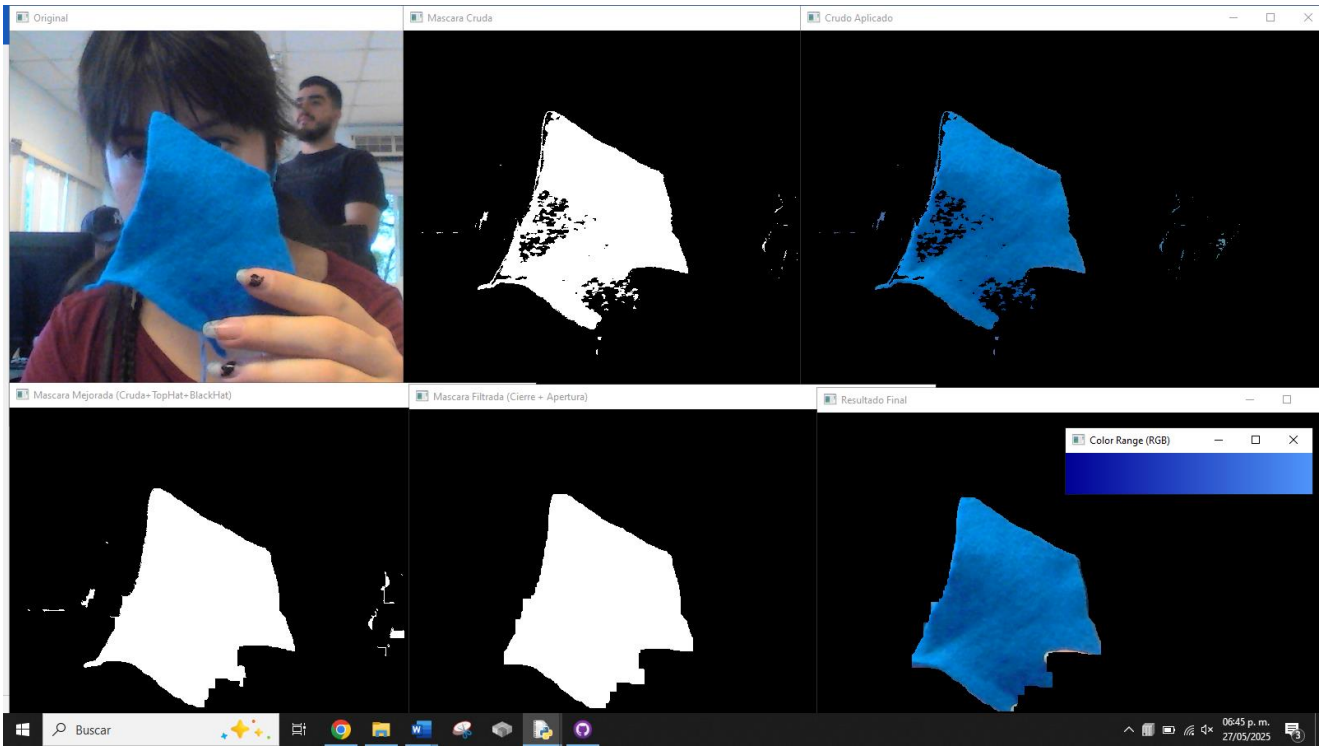
```
# Practica 7: Morfología con suma de TopHat y BlackHat antes del cierre y apertura

import cv2                  # Librería OpenCV para visión artificial
import numpy as np          # Librería NumPy para manejo de matrices

# Inicializar la cámara principal
cap = cv2.VideoCapture(0)

# Rango de azul definido en RGB
min_RGB = np.array([0, 0, 150])
max_RGB = np.array([80, 150, 250])

# Convertir RGB a BGR (OpenCV usa BGR)
bgr_min = min_RGB[::-1]
bgr_max = max_RGB[::-1]

# Crear barra de colores para mostrar rango
def create_color_bar(min_color, max_color, width=300, height=50):
    bar = np.zeros((height, width, 3), dtype=np.uint8)
    for i in range(width):
        alpha = i / width
        color = (1 - alpha) * min_color + alpha * max_color
        bar[:, i] = color
    return bar.astype(np.uint8)

color_bar = create_color_bar(bgr_min, bgr_max)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # 1. Máscara cruda (blanco donde hay azul)
    mask = cv2.inRange(frame, bgr_min, bgr_max)
```

```
    # 2. Aplicar máscara cruda directamente
    res_crudo = cv2.bitwise_and(frame, frame, mask=mask)

    # 3. Crear kernel estructurante
    kernel = np.ones((6, 6), np.uint8)

    # 4. TopHat y BlackHat desde la máscara cruda (antes de cierre y apertura)
    tophat = cv2.morphologyEx(mask, cv2.MORPH_TOPHAT, kernel)
    blackhat = cv2.morphologyEx(mask, cv2.MORPH_BLACKHAT, kernel)

    # 5. Sumar TopHat y BlackHat a la máscara cruda
    mascara_mejorada = cv2.add(mask, tophat)
    mascara_mejorada = cv2.add(mascara_mejorada, blackhat)

    # 6. Aplicar cierre + apertura a la máscara mejorada
    closing = cv2.morphologyEx(mascara_mejorada, cv2.MORPH_CLOSE, kernel)
    opening = cv2.morphologyEx(closing, cv2.MORPH_OPEN, kernel)

    # 7. Aplicar solo la máscara filtrada (opening) a la imagen original
    resultado_filtrado = cv2.bitwise_and(frame, frame, mask=opening)

    # 8. Aplicar la máscara mejorada (después de cierre + apertura) al frame
    resultado_final = cv2.bitwise_and(frame, frame, mask=opening)

    # Mostrar ventanas
    cv2.imshow('Original', frame)
    cv2.imshow('Mascara Cruda', mask)
    cv2.imshow('Crudo Aplicado', res_crudo)
    cv2.imshow('Mascara Mejorada (Cruda+TopHat+BlackHat)', mascara_mejorada)
    cv2.imshow('Mascara Filtrada (Cierre + Apertura)', opening)
    #cv2.imshow('Resultado Solo con Mascara Filtrada', resultado_filtrado)
    #cv2.imshow('TopHat (blancos pequeños)', tophat)
    #cv2.imshow('BlackHat (negros pequeños)', blackhat)
    cv2.imshow('Resultado Final', resultado_final)
    cv2.imshow('Color Range (RGB)', color_bar)

    # Tecla ESC para salir
    if cv2.waitKey(5) & 0xFF == 27:
        break

# Liberar recursos
cap.release()
cv2.destroyAllWindows()
```



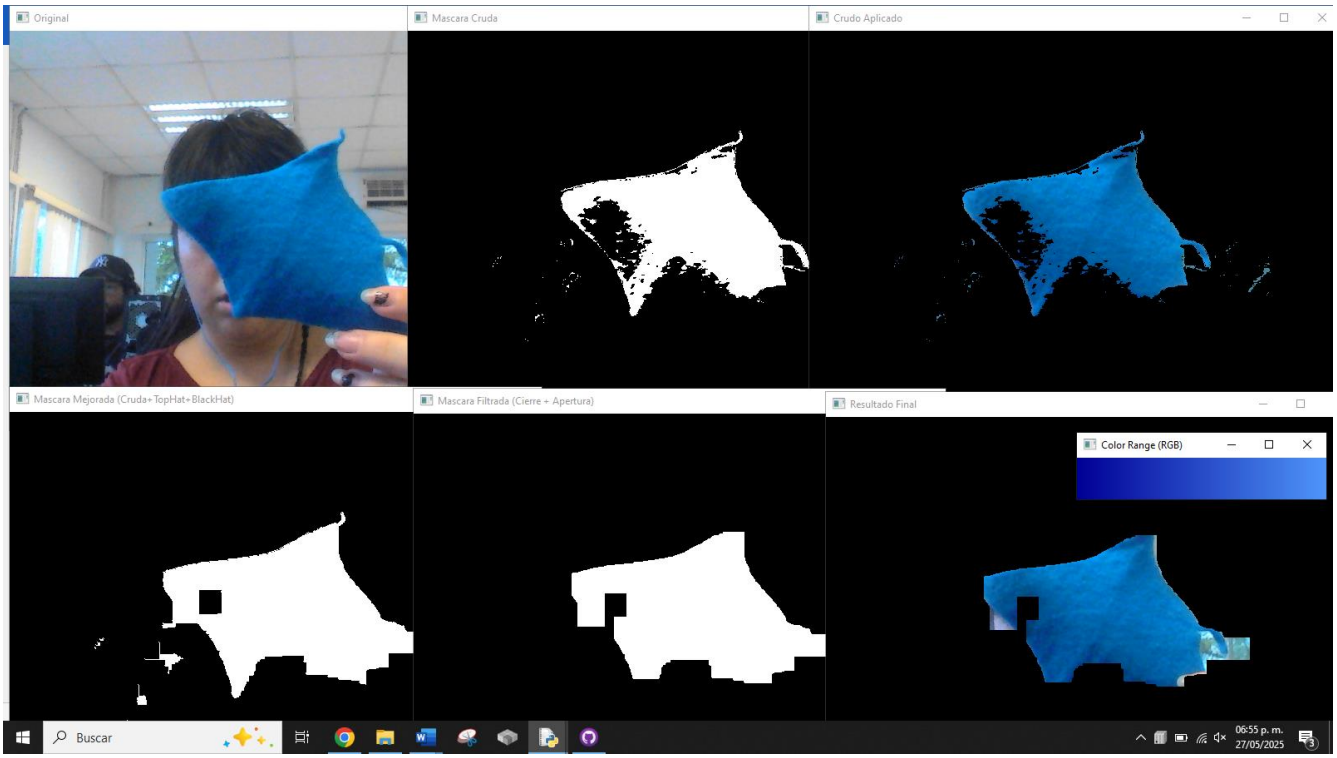
Con la modificacion de este orden podemos ver como antes de las operaciones de cierre y apertura se añade el delineado de la figura, pero al aplicar estas operaciones, estos bordes se integran, haciendo que se vea mejor el objeto. Si no tuviera las operaciones de TopHat y BlackHat, los bordes se verian con mucho ruido.

Por su parte, analicemos que pasa si aumentamos el valor de kernel:



Al subir el valor de kernel, que es el tamaño de la matriz o numero de pixeles que se toman en cuenta para hacer cada operación de apertura y cierre, se mejora la detección del objeto y se elimina mejor el ruido, tanto el de fondo como el que esta dentro del objeto, pero si se sube demasiado este valor, puede perderse la imagen o distorsionarse la forma del objeto que queremos detectar.

Esta variable también se usa para las operaciones de TopHat y BlackHat, por lo que al ser muy grande el valor, los bordes también se ven afectados, dejando de seguir la línea del borde del objeto y dejando pasar partes de la imagen que ya no corresponden al objeto.



Conclusión:

En esta práctica se exploraron técnicas morfológicas aplicadas al procesamiento de imágenes en tiempo real para mejorar la detección de objetos basados en color, específicamente en tonos azules. El objetivo fue limpiar la imagen, eliminar ruido externo y rellenar imperfecciones internas para resaltar la figura de interés de manera más precisa. Se aplicaron las operaciones morfológicas en el siguiente orden: TopHat, apertura, cierre y BlackHat. Este orden permitió, en primer lugar, resaltar los detalles pequeños y claros dentro del objeto con TopHat; después, la apertura eliminó puntos blancos de ruido fuera del objeto; el cierre rellenó huecos negros dentro del área detectada; y finalmente, BlackHat ayudó a corregir detalles oscuros residuales, afinando los bordes del objeto. Se concluyó que el orden de las operaciones morfológicas influye directamente en la calidad del resultado final y que su combinación adecuada permite obtener una máscara más limpia y fiel a la forma original del objeto, lo cual es útil en aplicaciones de visión por computadora, reconocimiento de objetos y sistemas de inspección automatizada.

Fuentes:

https://support.ptc.com/help/mathcad/r10.0/es/index.html#page/PTC_Mathcad_Help/opening_and_closing.html

https://users.exa.unicen.edu.ar/catedras/pdi/FILES/TE/Teorica5.pdf