Rodrigo Hernández Zavala

26 de mayo de 2025

Ing. Mecatrónica

22310215

Remover ruido, F+, F-, Dilatación Erosión

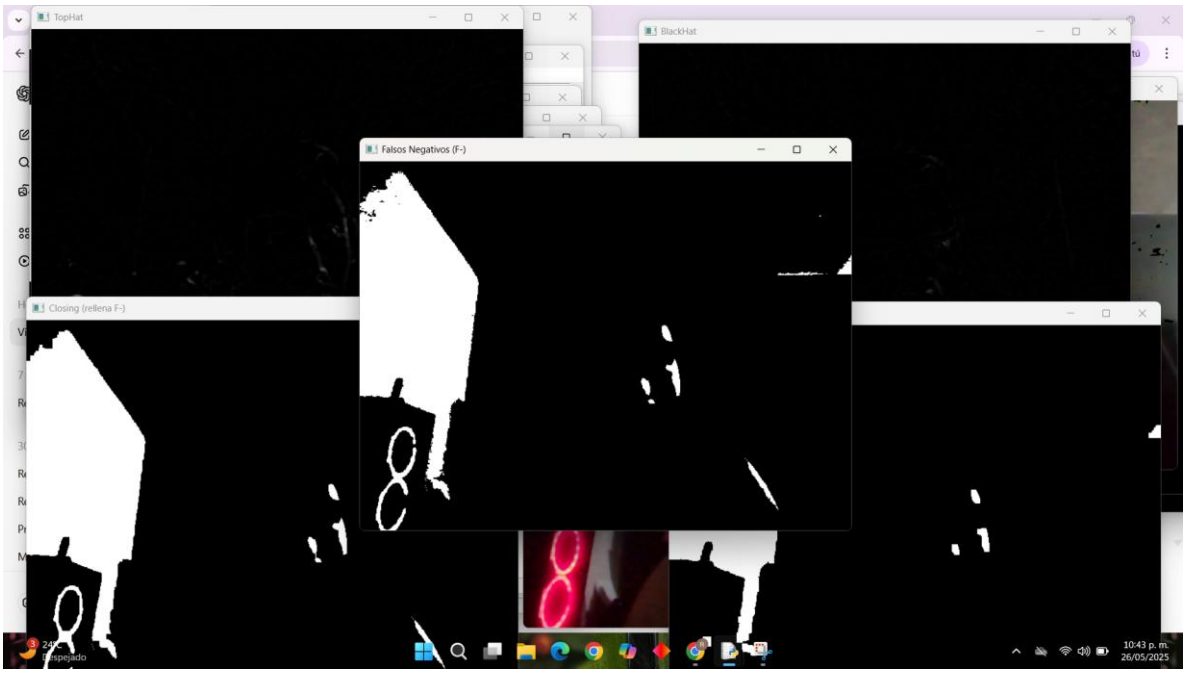
Visión Artificial

```
import cv2
import numpy as np

# ------------------- Captura de video -------------------
cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print("Error: No se puede acceder a la cámara.")
    exit()

# Kernel morfológico
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))

while True:
    ret, frame = cap.read()
    if not ret:
        print("Error al capturar el video.")
        break

    frame = cv2.resize(frame, (640, 480))
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # ------------------- Filtros HSV -------------------
    # Rojo (dos rangos)
    lower_red1 = np.array([0, 120, 70])
    upper_red1 = np.array([10, 255, 255])
    mask_red1 = cv2.inRange(hsv, lower_red1, upper_red1)
```

```
lower_red2 = np.array([170, 120, 70])
upper_red2 = np.array([180, 255, 255])
mask_red2 = cv2.inRange(hsv, lower_red2, upper_red2)

mask_red = cv2.add(mask_red1, mask_red2)

# Verde
lower_green = np.array([40, 40, 40])
upper_green = np.array([70, 255, 255])
mask_green = cv2.inRange(hsv, lower_green, upper_green)

# Azul
lower_blue = np.array([100, 150, 0])
upper_blue = np.array([140, 255, 255])
mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)

# Aplicar máscaras
res_red = cv2.bitwise_and(frame, frame, mask=mask_red)
res_green = cv2.bitwise_and(frame, frame, mask=mask_green)
res_blue = cv2.bitwise_and(frame, frame, mask=mask_blue)

# ------------------- Green Screen -------------------
mask_inv_green = cv2.bitwise_not(mask_green)
green_removed = cv2.bitwise_and(frame, frame, mask=mask_inv_green)

# ------------------- YUV -------------------
yuv = cv2.cvtColor(frame, cv2.COLOR_BGR2YUV)

# ------------------- MORFOLOGÍA -------------------
```

```
# Binarizar para operaciones
_, binary = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)

# Falsos positivos (F+): puntos blancos extra
fp_sim = binary.copy()
cv2.circle(fp_sim, (100, 100), 10, 255, -1)

# Falsos negativos (F−): huecos negros simulados
fn_sim = binary.copy()
cv2.rectangle(fn_sim, (300, 200), (320, 220), 0, -1)

# MORPH: TopHat y BlackHat
tophat = cv2.morphologyEx(gray, cv2.MORPH_TOPHAT, kernel)
blackhat = cv2.morphologyEx(gray, cv2.MORPH_BLACKHAT, kernel)

# Erosión y Dilatación
erosion = cv2.erode(fn_sim, kernel, iterations=1)
dilation = cv2.dilate(fp_sim, kernel, iterations=1)

# Opening (elimina F+), Closing (rellena F−)
opening = cv2.morphologyEx(fp_sim, cv2.MORPH_OPEN, kernel)
closing = cv2.morphologyEx(fn_sim, cv2.MORPH_CLOSE, kernel)

# ---------- Mostrar Resultados ----------
cv2.imshow('Original', frame)
cv2.imshow('Gris', gray)
cv2.imshow('Filtro Rojo', res_red)
cv2.imshow('Filtro Verde', res_green)
cv2.imshow('Filtro Azul', res_blue)
```

```
    cv2.imshow('Green Screen (Verde eliminado)', green_removed)
    cv2.imshow('YUV', yuv)

    # Resultados morfológicos
    cv2.imshow('Falsos Positivos (F+)', fp_sim)
    cv2.imshow('Falsos Negativos (F-)', fn_sim)
    cv2.imshow('Dilatación', dilation)
    cv2.imshow('Erosión', erosion)
    cv2.imshow('Opening (quita F+)', opening)
    cv2.imshow('Closing (rellena F-)', closing)
    cv2.imshow('TopHat', tophat)
    cv2.imshow('BlackHat', blackhat)

    # Salida con tecla 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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




```python
import cv2
import numpy as np

# ------------------- Captura de video -------------------
cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print("Error: No se puede acceder a la cámara.")
    exit()

# Kernel morfológico
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (7, 7))

while True:
    ret, frame = cap.read()
    if not ret:
        print("Error al capturar el video.")
        break

    frame = cv2.resize(frame, (640, 480))
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # ------------------- Filtros HSV -------------------
    # Rojo (dos rangos)
    lower_red1 = np.array([0, 120, 70])
    upper_red1 = np.array([10, 255, 255])
    mask_red1 = cv2.inRange(hsv, lower_red1, upper_red1)

    lower_red2 = np.array([170, 120, 70])
    upper_red2 = np.array([180, 255, 255])
    mask_red2 = cv2.inRange(hsv, lower_red2, upper_red2)

    mask_red = cv2.add(mask_red1, mask_red2)

    # Verde
    lower_green = np.array([40, 40, 40])
    upper_green = np.array([70, 255, 255])
    mask_green = cv2.inRange(hsv, lower_green, upper_green)
```

```python
# Azul
lower_blue = np.array([100, 150, 0])
upper_blue = np.array([140, 255, 255])
mask_blue = cv2.inRange(hsv, lower_blue, upper_blue)

# Aplicar máscaras
res_red = cv2.bitwise_and(frame, frame, mask=mask_red)
res_green = cv2.bitwise_and(frame, frame, mask=mask_green)
res_blue = cv2.bitwise_and(frame, frame, mask=mask_blue)

# ------------------- Green Screen -------------------
mask_inv_green = cv2.bitwise_not(mask_green)
green_removed = cv2.bitwise_and(frame, frame, mask=mask_inv_green)

# ------------------- YUV --------------------
yuv = cv2.cvtColor(frame, cv2.COLOR_BGR2YUV)

# ------------------- MORFOLOGÍA -------------------
# Binarizar para operaciones
_, binary = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)

# Falsos positivos (F+): puntos blancos extra
fp_sim = binary.copy()
cv2.circle(fp_sim, (100, 100), 10, 255, -1)

# Falsos negativos (F-): huecos negros simulados
fn_sim = binary.copy()
cv2.rectangle(fn_sim, (300, 200), (320, 220), 0, -1)

# MORPH: TopHat y BlackHat
tophat = cv2.morphologyEx(gray, cv2.MORPH_TOPHAT, kernel)
blackhat = cv2.morphologyEx(gray, cv2.MORPH_BLACKHAT, kernel)
```

```
# Erosión y Dilatación
erosion = cv2.erode(fn_sim, kernel, iterations=1)
dilation = cv2.dilate(fp_sim, kernel, iterations=1)

# Opening (elimina F+), Closing (rellena F-)
opening = cv2.morphologyEx(fp_sim, cv2.MORPH_OPEN, kernel)
closing = cv2.morphologyEx(fn_sim, cv2.MORPH_CLOSE, kernel)

# ------------------- Mostrar Resultados -------------------
cv2.imshow('Original', frame)
cv2.imshow('Gris', gray)
cv2.imshow('Filtro Rojo', res_red)
cv2.imshow('Filtro Verde', res_green)
cv2.imshow('Filtro Azul', res_blue)
cv2.imshow('Green Screen (Verde eliminado)', green_removed)
cv2.imshow('YUV', yuv)

# Resultados morfológicos
cv2.imshow('Falsos Positivos (F+)', fp_sim)
cv2.imshow('Falsos Negativos (F-)', fn_sim)
cv2.imshow('Dilatación', dilation)
cv2.imshow('Erosión', erosion)
cv2.imshow('Opening (quita F+)', opening)
cv2.imshow('Closing (rellena F-)', closing)
cv2.imshow('TopHat', tophat)
cv2.imshow('BlackHat', blackhat)

# Salida con tecla 'q'
if cv2.waitKey(1) & 0xFF == ord('q'):
    break

.release()
.destroyAllWindows()
```

Link:

https://github.com/Yiodrigo/Artificial_Visi-n.git

¿Qué hace este código?

- Captura video en tiempo real.
- Detecta **colores rojo, verde y azul**.
- Simula errores de detección (**F+ y F−**).

- Aplica filtros **morfológicos** para limpiar el ruido.
- Muestra **efectos de TopHat y BlackHat** para resaltar detalles.

¿Qué verás?

- **F+** simulado como puntos blancos extra.
- **F−** simulado como huecos negros dentro de objetos.
- Cómo **Opening** y **Closing** eliminan ese ruido.
- **Dilatación** agranda objetos blancos.
- **Erosión** elimina detalles finos blancos.
- **TopHat** resalta regiones **claras pequeñas**.
- **BlackHat** resalta regiones **oscuras pequeñas**.