VISIÓN ARTIFICIAL

Proyecto. Conteo de automóviles

Ingeniería en Mecatrónica
6to semestre

Mtro. Mauricio Alejandro Cabrera Arellano

Arturo Alejandro Guzmán Pérez – 22110356

Descripción del código:

El sistema realiza el conteo de vehículos que cruzan una línea virtual trazada sobre un video pregrabado, utilizando técnicas de sustracción de fondo y seguimiento de objetos.

Principales bloques funcionales:

1. **Inicialización de video y parámetros**
    - Se abre un video mediante `cv2.VideoCapture`.
    - Se define una línea horizontal que funciona como límite de conteo.
2. **Detección de movimiento**
    - Se utiliza `cv2.createBackgroundSubtractorMOG2` para obtener una máscara binaria de movimiento.
    - Posteriormente se aplica una dilatación y detección de contornos para delimitar los vehículos.
3. **Rastreo**
    - Por cada contorno detectado que cumpla con un tamaño mínimo, se inicia un rastreador de correlación (`dlib.correlation_tracker`).
    - Cada rastreador se actualiza en cada fotograma, obteniendo su posición central.
4. **Conteo de vehículos**
    - Si la coordenada `y` del centro del objeto cruza la línea de conteo dentro de un margen de tolerancia (`offset`), se incrementa el contador y se elimina ese rastreador de la lista.
5. **Visualización**
    - Se muestra en pantalla el video procesado con la línea de conteo, los centros de los objetos rastreados y el conteo acumulado de vehículos.

Código:

Proyecto_conteo_de_vehiculos.py - C:\Users\Arturo Alejandro\Documents\GitHub\VA-2025\...

File Edit Format Run Options Window Help

```
import cv2
import dlib
import imutils
from imutils.video import VideoStream
import time

# Inicializa video
cap = cv2.VideoCapture("highway.mp4")  # Cambia esto por 0 si usas cámara web

# Línea de conteo (y-coordinate)
count_line_position = 250
vehicle_count = 0

# Inicializa detector de movimiento y rastreador
detector = cv2.createBackgroundSubtractorMOG2()
tracker_list = []
trackers = dlib.correlation_tracker
track_id = 0

# Umbral de detección
min_width = 80
min_height = 80
offset = 6  # margen para contar solo una vez

def center_handle(x, y, w, h):
    x1 = int(w / 2)
    y1 = int(h / 2)
    cx = x + x1
    cy = y + y1
    return cx, cy

detects = []

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

    frame = imutils.resize(frame, width=800)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

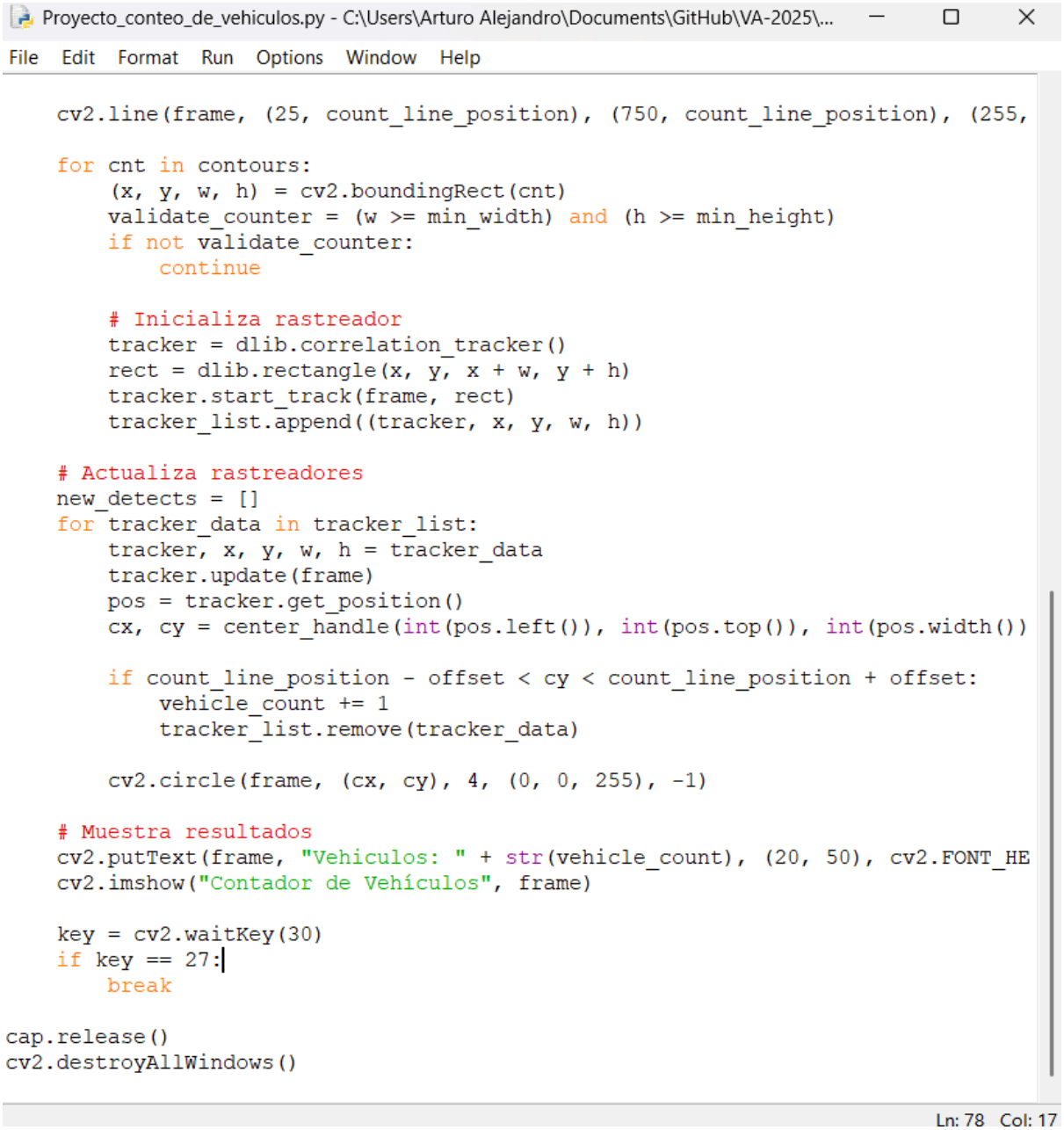
Ln: 1 Col: 0

Proyecto_conteo_de_vehiculos.py - C:\Users\Arturo Alejandro\Documents\GitHub\VA-2025\...

File Edit Format Run Options Window Help

```
    frame = imutils.resize(frame, width=800)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    fgmask = detector.apply(frame)
    dilated = cv2.dilate(fgmask, None, iterations=2)
    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMP

    cv2.line(frame, (25, count_line_position), (750, count_line_position), (255,

    for cnt in contours:
        (x, y, w, h) = cv2.boundingRect(cnt)
        validate_counter = (w >= min_width) and (h >= min_height)
        if not validate_counter:
            continue

        # Inicializa rastreador
        tracker = dlib.correlation_tracker()
        rect = dlib.rectangle(x, y, x + w, y + h)
        tracker.start_track(frame, rect)
        tracker_list.append((tracker, x, y, w, h))

    # Actualiza rastreadores
    new_detects = []
    for tracker_data in tracker_list:
        tracker, x, y, w, h = tracker_data
        tracker.update(frame)
        pos = tracker.get_position()
        cx, cy = center_handle(int(pos.left()), int(pos.top()), int(pos.width())

        if count_line_position - offset < cy < count_line_position + offset:
            vehicle_count += 1
            tracker_list.remove(tracker_data)

        cv2.circle(frame, (cx, cy), 4, (0, 0, 255), -1)

    # Muestra resultados
    cv2.putText(frame, "Vehiculos: " + str(vehicle_count), (20, 50), cv2.FONT_HE
    cv2.imshow("Contador de Vehículos", frame)

    key = cv2.waitKey(30)
    if key == 27:
```

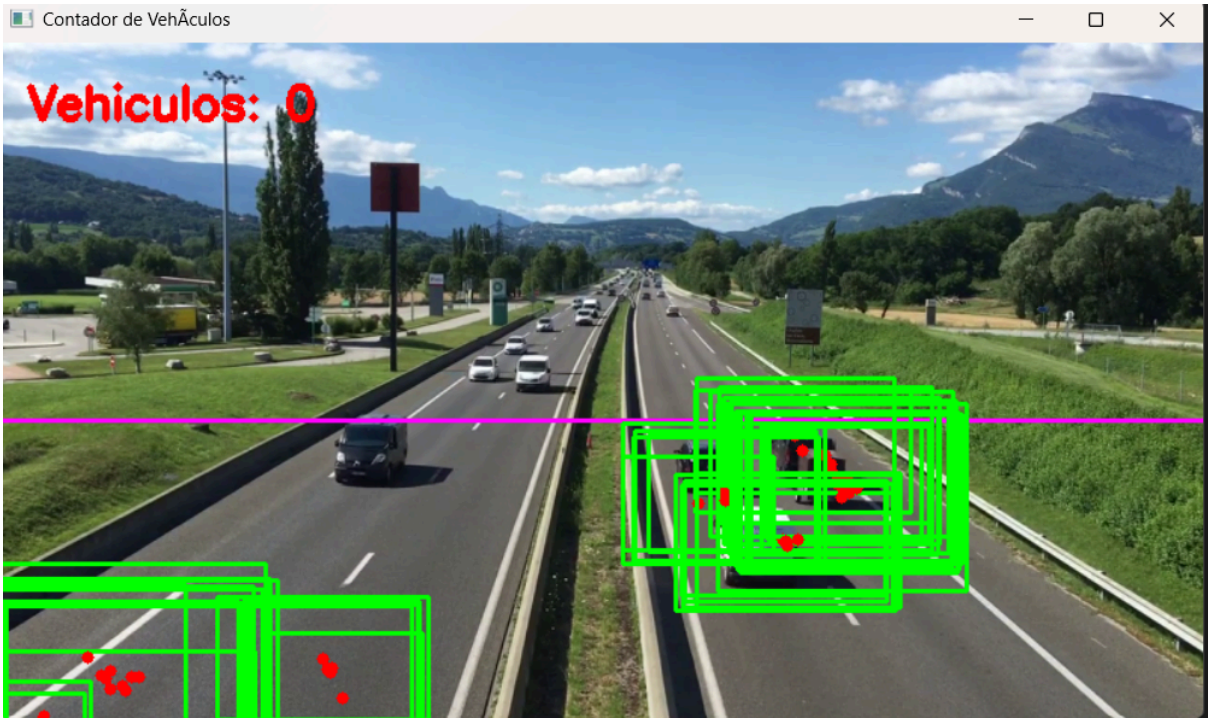
Ln: 40 Col: 50



Resultados:

El sistema permite contar de manera eficiente los vehículos que atraviesan la línea definida en el video. Se mostraron los centros de los objetos rastreados mediante círculos rojos y el total acumulado en pantalla.

Captura de funcionamiento:



Conclusiones:

- Es posible implementar un sistema funcional de conteo de objetos móviles mediante visión artificial usando Python y librerías accesibles como OpenCV y Dlib.
- La precisión del conteo depende de factores como la iluminación, la posición de la cámara, la velocidad de los objetos y el tamaño mínimo de detección.
- Mejoras posibles:
    - Implementar lógica para evitar contar el mismo objeto más de una vez en caso de rebases o detenciones.
    - Incluir clasificación de vehículos por tamaño o tipo.
    - Optimizar la gestión de rastreadores para reducir la carga de memoria y aumentar la eficiencia en tiempo real.