Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería Informática

Task 2: Sistema de Detección, Seguimiento y Conteo de Vehículos

Big Data – Curso Académico 2025–2026

Trabajo Individual

Autor:

Adrián Ojeda Viera (DNI: 45364780V)
Laura Aguiar Pérez (DNI: xxxxxxxx)

Repositorio:

Click para ver el repositorio

Fecha de Entrega: 22 de Noviembre, 2025

"Este documento presenta la implementación y análisis de un sistema de visión por computador para el conteo y clasificación de tráfico mediante técnicas de sustracción de fondo y operaciones morfológicas."

1. Implementación del Sistema de Detección y Conteo de Vehículos

Para la realización de este trabajo se ha utilizado exclusivamente la librería `OpenCV (cv2)` junto con `numpy` para el manejo de matrices y operaciones matemáticas. El objetivo del código es procesar un flujo de video de tráfico, detectar vehículos en movimiento, realizar un seguimiento (*tracking*) de los mismos y contarlos según crucen líneas virtuales predefinidas en diferentes carriles y direcciones.

1.1. Preprocesamiento y Funciones Auxiliares

El primer paso fundamental es el acondicionamiento de la imagen. Dado que el video original puede tener una resolución elevada (1920x1080), se define la función `rescale` para redimensionar cada frame a una escala más manejable (0.6x). Esto no solo facilita la visualización en pantalla, sino que reduce significativamente la carga computacional, permitiendo un procesamiento más fluido en tiempo real.

```
def rescale(frame, scale=SCALE_FACTOR):
    return cv2.resize(frame, (int(frame.shape[1]*scale), int(frame.shape
    [0]*scale)), interpolation=cv2.INTER_AREA)
```

Listing 1: Función de reescalado

Adicionalmente, se implementan funciones geométricas clave para la lógica de conteo: `point_line_signed_distance` y `point_to_segment_proj_dist`. Estas funciones permiten calcular la posición relativa de un punto (centroide del vehículo) respecto a un segmento de línea, determinando si ha ocurrido un cruce y la distancia exacta al segmento, lo cual es vital para evitar falsos positivos.

1.2. Detección de Movimiento y Sustracción de Fondo

El núcleo del sistema de detección recae en el algoritmo de sustracción de fondo MOG2 (`createBackgroundSubtractorMOG2`). Este método modela cada píxel como una mezcla de gaussianas, lo que le permite adaptarse a cambios de iluminación graduales y separar los objetos en movimiento del fondo estático.

```
object_detector = cv2.createBackgroundSubtractorMOG2(history=250,
    varThreshold=32, detectShadows=True)
```

Listing 2: Configuración del Detector MOG2

Se establece un periodo de calibración inicial de frames (definido en `BG_STABILIZATION_FRAMES`) donde el sistema procesa el video sin realizar conteo. Esto permite que el modelo de fondo se estabilice y aprenda la escena estática, evitando detecciones erróneas al inicio de la ejecución.

1.3. Procesamiento Morfológico y Limpieza de Máscara

La máscara binaria obtenida del detector MOG2 suele contener ruido (movimiento de árboles, reflejos) y fragmentación (vehículos divididos en partes). Para solucionar esto, se aplica una cadena de procesamiento de imagen robusta:

- **Suavizado Gaussiano (GaussianBlur):** Se aplica al frame en escala de grises antes de la detección para reducir el ruido de alta frecuencia del asfalto.
- **Umbralizado (threshold):** Se aplica un corte estricto (200-250) a la máscara resultante para eliminar sombras grises (si `detectShadows=True`, las sombras se marcan en gris), dejando solo los píxeles blancos correspondientes a objetos sólidos.
- **Operaciones Morfológicas:**
  - *Erosión (erode):* Con un kernel de 3x3, se eliminan pequeños píxeles de ruido aislados.
  - *Dilatación (dilate):* Se expanden las regiones blancas restantes para recuperar el volumen del vehículo.
  - *Cierre (morphologyEx con MORPH_ CLOSE):* Con un kernel más grande (11x11), se fusionan componentes cercanos. Esto es crítico para detectar camiones o vehículos largos como un único objeto en lugar de múltiples fragmentos (cabina + remolque).

```
_, bin_img = cv2.threshold(mask, BINARY_THRESHOLD, 255, cv2.
    THRESH_BINARY)
bin_img = cv2.morphologyEx(bin_img, cv2.MORPH_OPEN, KERNEL_ERODE)
bin_img = cv2.morphologyEx(bin_img, cv2.MORPH_CLOSE, KERNEL_CLOSE)
```

Listing 3: Aplicación de Umbral y Morfología

1.4. Detección de Contornos y Filtrado

Sobre la máscara binaria limpia, se utiliza `cv2.findContours` para identificar los objetos. Se aplica un filtrado riguroso basado en el área (`MIN_CONTOUR_AREA`), dimensiones mínimas y relación de aspecto. Además, se introduce un filtro de horizonte (ignorar si $y <$ umbral), descartando cualquier movimiento en la parte superior de la imagen (cielo o vegetación lejana) que no corresponda a la carretera.

1.5. Seguimiento (Tracking) y Lógica de Conteo

El sistema implementa un algoritmo de seguimiento basado en la distancia Euclidiana entre centroides. Para cada frame, se comparan los centroides detectados con los objetos

ya registrados en `tracking_objs`. Si la distancia es menor a `MAX_DISTANCE`, se actualiza la posición del objeto existente; de lo contrario, se registra como un nuevo vehículo con un ID único.

El conteo se realiza mediante líneas virtuales definidas por coordenadas. Se evalúa si la trayectoria de un vehículo cruza estas líneas analizando el cambio de signo en la distancia con signo entre su posición anterior y la actual. Se han definido 5 líneas de conteo estratégicas para cubrir diferentes flujos de tráfico:

- **Norte y Sur:** Tráfico principal.
- **INC Norte y INC Sur:** Incorporaciones laterales.
- **DECEL:** Carril de desaceleración.

Cada vehículo cuenta con un registro para asegurar que solo sea contabilizado una vez por cada línea que cruce.

1.6. Visualización de Resultados

Finalmente, el sistema genera una visualización en tiempo real compuesta por tres capas de información:

1. **Video Original:** Referencia visual base.
2. **Máscara Procesada:** Visualización de la "visiónïnterna del algoritmo tras los filtros morfológicos.
3. **Monitor Final:** Overlay con las cajas delimitadoras (*bounding boxes*), IDs, velocidad estimada, líneas de conteo y un panel de estadísticas.

2. Anexo: Configuración de Parámetros

A continuación se presenta la configuración final de parámetros extraídos para el ajuste fino (*tuning*) del algoritmo:

```
# 1. CONFIGURACION DEL VIDEO
VIDEO_PATH = 'trafico.mp4'
SCALE_FACTOR = 0.6

# 2. FISICA Y VELOCIDAD
PX_TO_METERS = 0.15
SPEED_LOOKBACK = 6

# 3. DETECCION DE FONDO
BG_HISTORY = 250
BG_VAR_THRESH = 32
BG_SHADOWS = True
BG_LEARNING_RATE = 0.003
BG_STABILIZATION_FRAMES = 150

# 4. FILTROS DE IMAGEN
MEDIAN_BLUR_SIZE = 5
BINARY_THRESHOLD = 200

# 5. MORFOLOGIA
KERNEL_ERODE_SIZE = (3, 3)
KERNEL_CLOSE_SIZE = (11, 11)

# 6. FILTROS DE OBJETOS
MIN_CONTOUR_AREA = 100
MIN_BOX_WIDTH = 10
MIN_BOX_HEIGHT = 10
MIN_ASPECT_RATIO = 0.2
MAX_ASPECT_RATIO = 5.0

# 7. TRACKING
MAX_TRACKING_DIST = 80
MAX_HISTORY_LEN = 20

# 8. LINEAS DE CONTEO Y ZONAS
LINES_CFG = [
    {"id": "north", "label": "NORTE (P)", "pts": ((550, 450), (650, 400)), "col": (255, 0, 0),   "th": 25, "tol": (0, 1)},
    {"id": "south", "label": "SUR (P)",   "pts": ((750, 425), (850, 375)), "col": (0, 0, 255),   "th": 25, "tol": (0, 1)},
    {"id": "inc_n", "label": "INC NORTE", "pts": ((300, 550), (450, 500)), "col": (255, 255, 0), "th": 40, "tol": (0, 1)},
```

```
    {"id": "inc_s", "label": "INC SUR",   "pts": ((880, 290), (940, 250)
    ), "col": (0, 255, 255), "th": 25, "tol": (0, 1)},
    {"id": "decel", "label": "DECEL",     "pts": ((190, 260), (90, 130))
    ,  "col": (255, 0, 255), "th": 60, "tol": (-0.2, 1.2)}
]
```

Listing 4: Parámetros Globales del Sistema