Vehicle speed detection based on Gaussian Mixture Model

Mariam Landa Bautista

Juan Paulo Salgado

Alfredo Díaz

Input de video
(videoreader)
(videoinput)

Extracción de datos
(theta,d,H)

Extracción de frames en
Matlab
(hasframe)

Extracción de background y
foreground
(vision.ForegroundDetector)

Filtrado de
imagen(imgaussfilter)

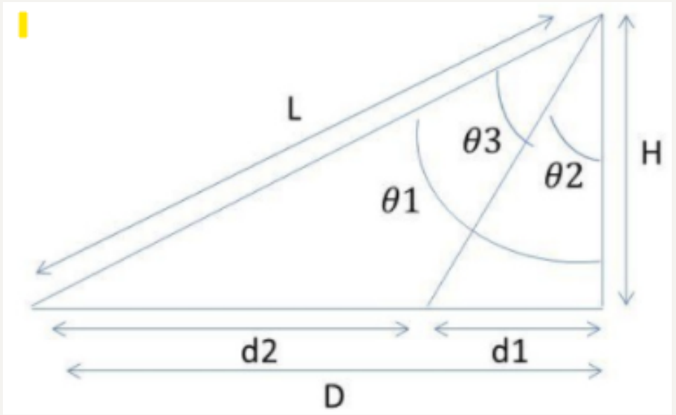
Análisis Blob
(regionprops)
(vision.BlobAnalysis)

Conversión de distancia a unidades reales.

Convierte el diagrama en pixeles, a partir de las imagenes del video, calcula la posición de los vehículos en la carretera.

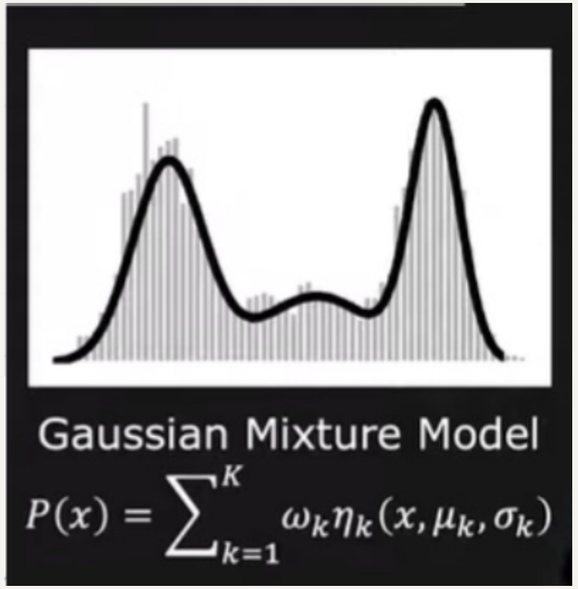
Calculo de tiempo y
velocidad

Resultado en interfaz



Indroducción

En este proyecto usamos Gaussian Mixture Model para la detección de velocidad de un auto, dicho modelo nos ayuda a distinguir la segmentación de fondo y la detección de objetos en movimiento.



Formulas para calcular la velociad

$$d_O = h \tan(\phi + \theta)$$

$$\theta = \tan^{-1} \frac{d_\theta}{h}$$

$$\phi = \tan^{-1}\left(\frac{\frac{N}{2}}{f}\right) - \tan^{-1}\left(\frac{y_t - \frac{N}{2}}{f}\right)$$

$$v = \frac{d_v \times FR \times 3.6}{Frame_t - Frame_{t-1}} \left(\frac{km}{h}\right)$$

Plataforma

Nuestra estructura se realizo de PVC, con una altura aproxmada de 3.30 metros. Y realizamos una base de madera, donde pusimos la raspberry con la bateria.

Código MATLAB

```
foregroundDetector = vision.ForegroundDetector('NumGaussians', 3, ...
'NumTrainingFrames', 250, 'MinimumBackgroundRatio', 0.3, ...
'InitialVariance', 110*110);
```

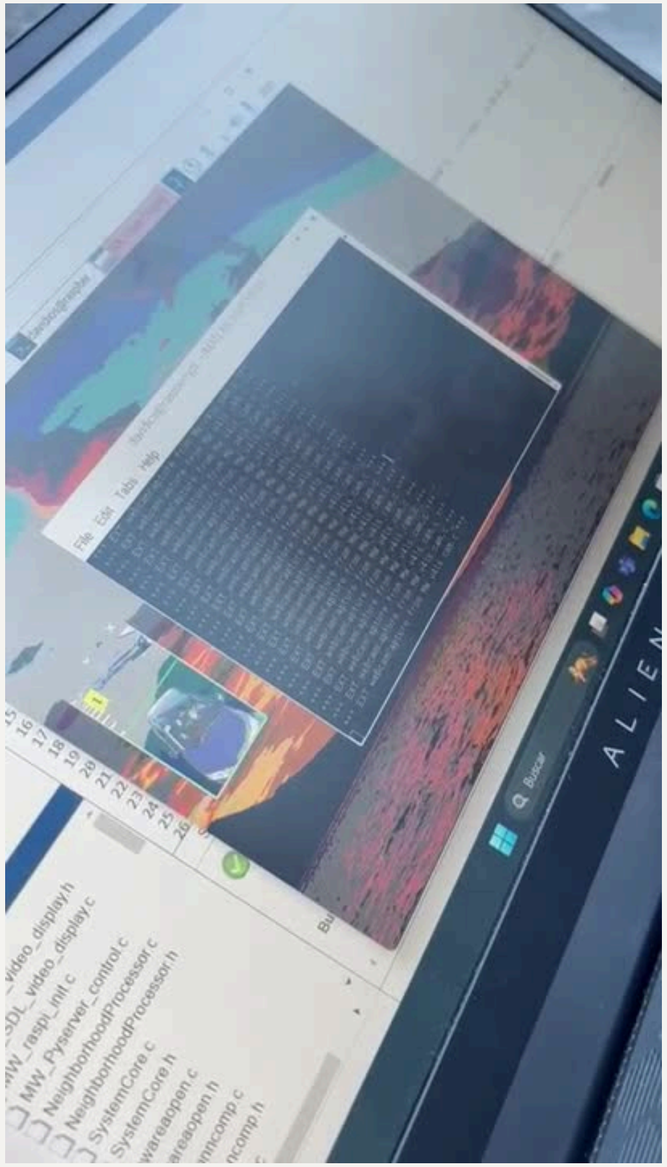
```
% Capturar imagen de la cámara
frame = snapshot(w);
croppedFrame = imcrop(frame, roi);
grayFrame = rgb2gray(croppedFrame);
filteredFrame = imgaussfilt(grayFrame, 2);
filteredFrame = imopen(filteredFrame, seOpen);
foreground = step(foregroundDetector, filteredFrame);
filteredForeground = bwareaopen(foreground, 500);
filteredForeground = imfill(filteredForeground, 'holes');
filteredForeground = imclose(filteredForeground, seClose);
filteredForeground = imdilate(filteredForeground, se);

stats = regionprops(filteredForeground, 'Centroid', 'BoundingBox', 'Area');
minArea = 1000;
stats = stats([stats.Area] >= minArea);
```

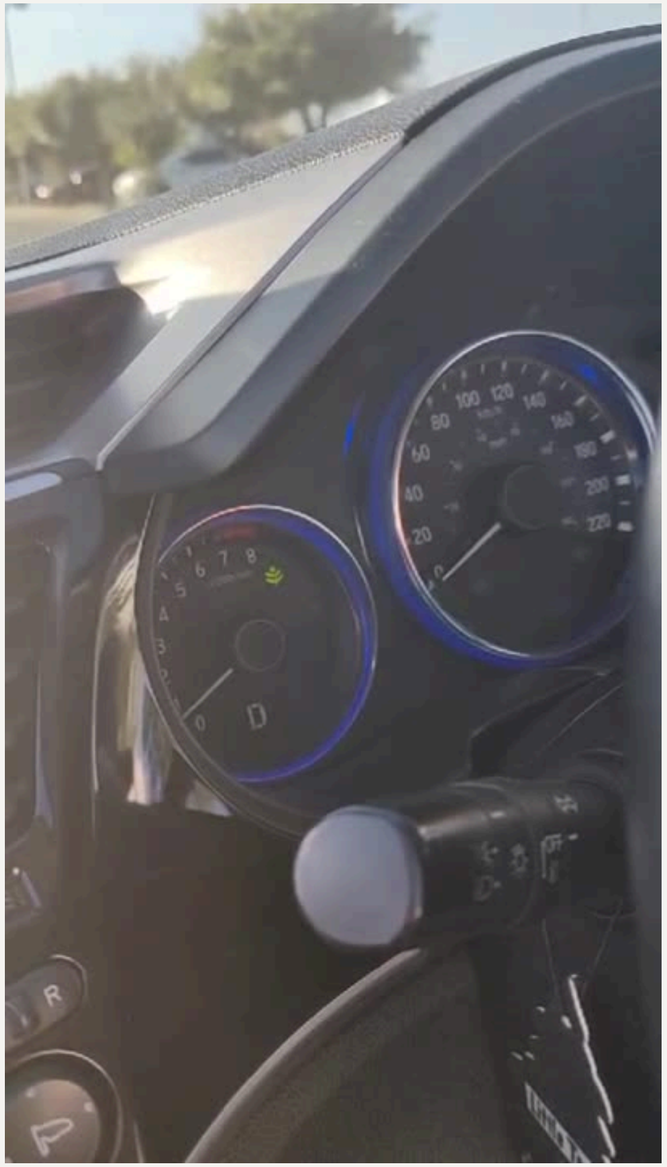
Código MATLAB

```
% Si el buffer aún no ha alcanzado el tamaño máximo, agregar el centroide
if bufferIndex <= N
    centroidBuffer(bufferIndex, :) = centroid;
    bufferIndex = bufferIndex + 1;
else
    % Si el buffer ya está lleno, hacer un "desplazamiento" para agregar el nuevo centroide
    centroidBuffer = circshift(centroidBuffer, [-1, 0]);
    centroidBuffer(N, :) = centroid;
end
```

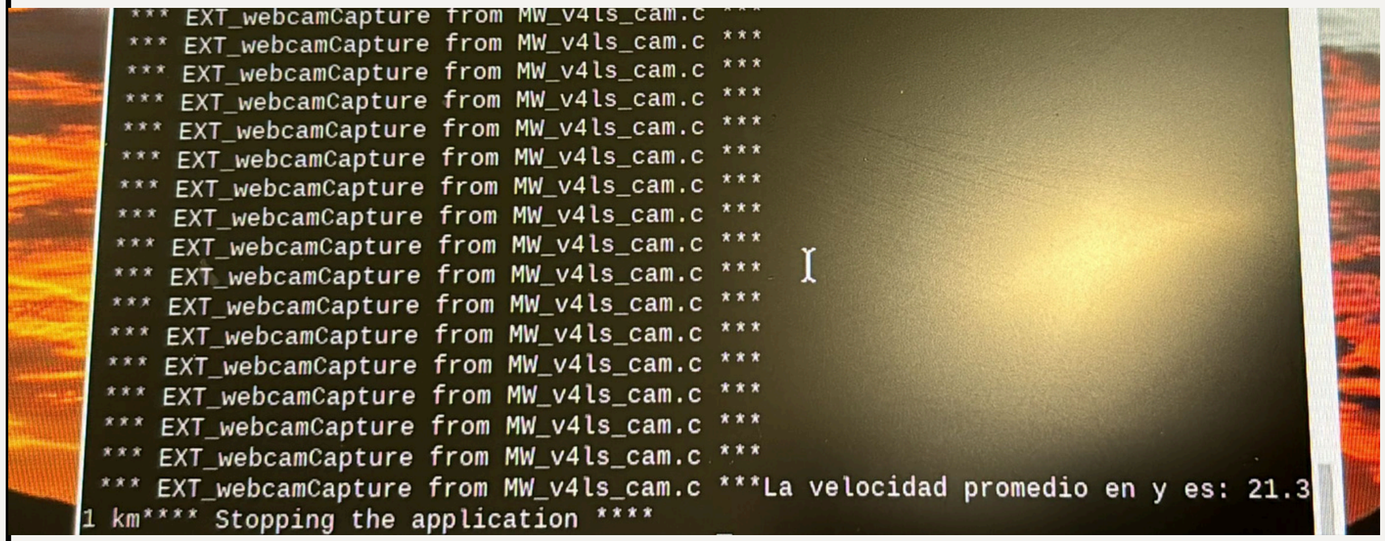
```
% Calcular el ángulo phi basado en la posición y del centroide suavizado
phi = atan((roi(4) / 2) / f) - atan((smoothedCentroid(2) - roi(4) / 2) / f);

% Calcular la distancia al objeto
dO = h * tan(phi + theta);
```

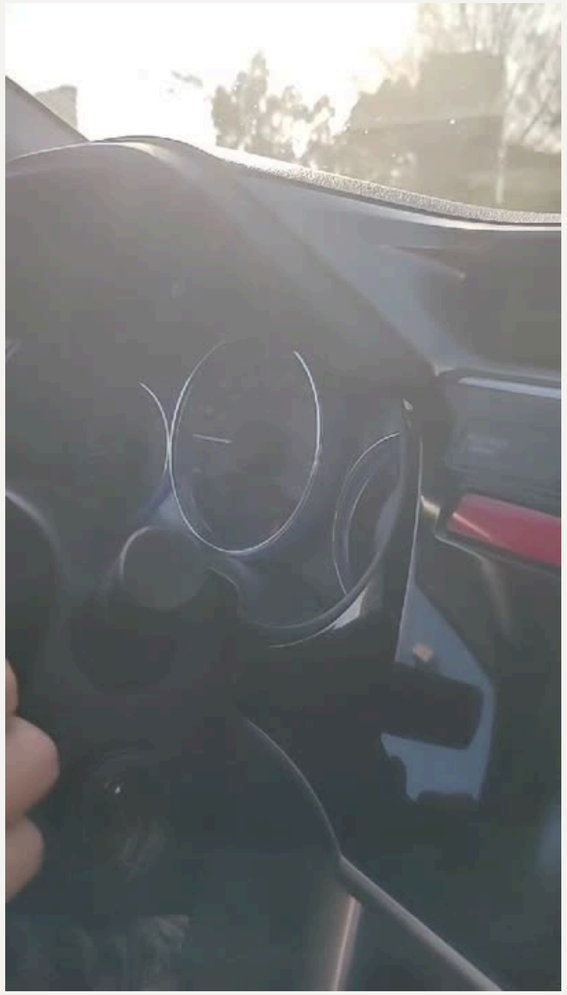
PRUEBAS 15km



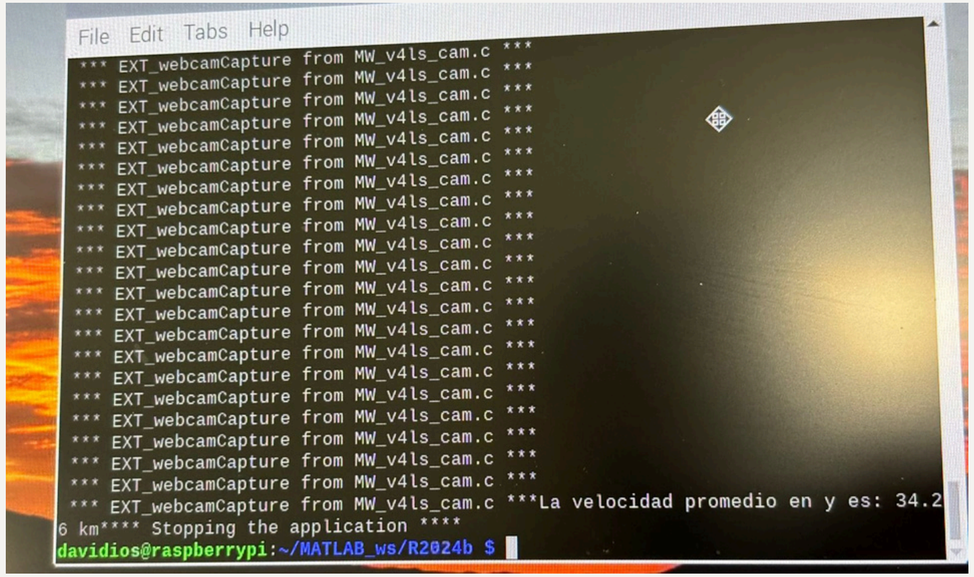


PRUEBAS 20 KM





PRUEBAS 30 K


File Edit Tabs Help
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***
*** EXT_webcamCapture from MW_v4ls_cam.c ***La velocidad promedio en y es: 34.2
6 km**** Stopping the application ****
davidios@raspberrypi:~/MATLAB_ws/R2024b $

Conclusión

¡Gracias por su atención!