# **Maestría en Inteligencia Artificial Aplicada**

## **Curso: Navegación autónoma**

### Tecnológico de Monterrey

### Dr. David Antonio Torres

#### **Actividad 2.1 - Detección de carriles en video usando transformada de Hough**

**Nombres y matrículas de los integrantes del equipo:**

* Julio Cesar Lynn Jimenez
* Francisco Javier Parga García A01794380
* Carlos Roberto Torres Ferguson
* Fernando Sebastian Sanchez Cardona

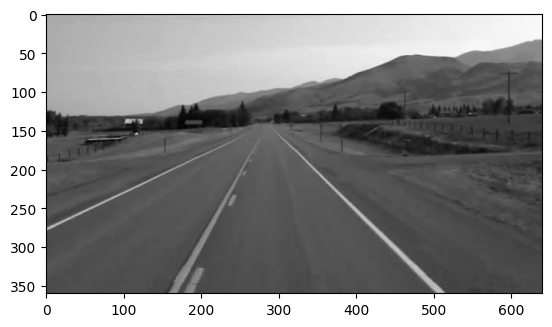
# Importar librerias necesarias

import matplotlib.image as mpimg  
import matplotlib.pyplot as plt  
import numpy as np  
import cv2  
import time  
  
from IPython.display import clear\_output  
from IPython.display import display  
from ipywidgets import Video, Image

# Ajuste de parámetros: Paso a paso

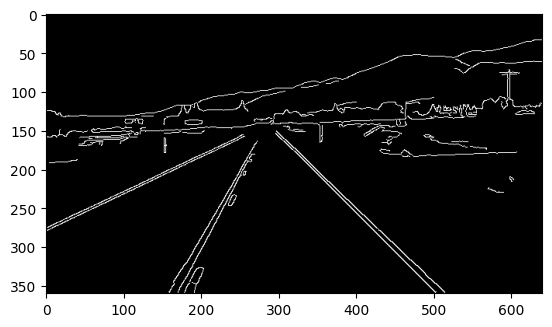
## Cargar video

# Graficamos el primer frame del video como una prueba de que el archivo fue leido correctamente  
video\_file = 'input/test2\_low-res.mp4'  
#video\_file = 'input/test2.mp4'  
cap = cv2.VideoCapture(video\_file)  
  
while cap.isOpened():  
 ret, image\_fr1 = cap.read()  
  
 if not ret:  
 break  
  
 img\_grey\_fr1 = cv2.cvtColor(image\_fr1, cv2.COLOR\_BGR2GRAY) # Convertir la imagen a escala de grises  
 img\_rgb\_fr1 = cv2.cvtColor(image\_fr1, cv2.COLOR\_BGR2RGB)  
 plt.imshow(img\_grey\_fr1, cmap='gray')  
 cap.release() # Libera el video y cumple como condicion de escape del ciclo while



## Detectar bordes en las imagenes

img\_blur = cv2.GaussianBlur(img\_grey\_fr1,(3,3), 0, 0)  
img\_canny = cv2.Canny(img\_blur, 40, 120)   
plt.imshow(img\_canny, cmap='gray');



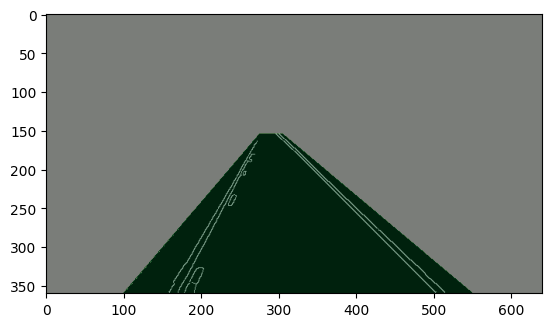
## Obtener medidas del video

img\_arr = np.asarray(img\_grey\_fr1)  
print(f'Alto: {img\_arr.shape[0]} px, ancho: {img\_arr.shape[1]} px.')

Alto: 360 px, ancho: 640 px.

## Definir area de interés y revisar bordes dentro del ROI

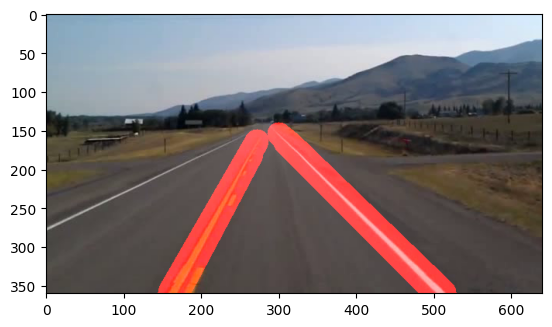
# ROI arbitraria  
#vertices = np.array([[(200,720),(550, 310), (610, 310), (1100,720)]], dtype=np.int32)  
vertices = np.array([[(100,360),(275, 155), (305, 155), (550,360)]], dtype=np.int32)  
# imagen en negro de las mismas dimensiones que la imagen original en escala de grises  
img\_roi = np.zeros\_like(img\_grey\_fr1)  
# la ROI se pinta de blanco sobre la imagen en negro  
cv2.fillPoly(img\_roi, vertices, 255)  
  
img\_mask = cv2.bitwise\_and(img\_canny, img\_roi)  
  
plt.imshow(img\_mask, cmap='gray')  
plt.imshow(img\_roi, cmap='Greens', alpha=0.5);



##### Resaltar lineas encontradas en el ROI

Para resaltar las caracteristicas de las lineas dentro del ROI, utilizamos la transformada de Hough para lineas

rho = 1 # resolución de rho en pixeles  
theta = np.pi/180 # resolución de theta en radianes   
threshold = 70 # mínimo número de votos para ser considerado una línea   
min\_line\_len = 20 # mínimo número de pixeles para que se forme una línea  
max\_line\_gap = 70 # máximo espacio en pixeles entre segmentos de línea   
lines = cv2.HoughLinesP(img\_mask, rho, theta, threshold, np.array([]), minLineLength=min\_line\_len, maxLineGap=max\_line\_gap)  
# se crea un fondo negro del tamaño de la imagen con bordes  
img\_lines = np.zeros((img\_mask.shape[0], img\_mask.shape[1], 3), dtype=np.uint8)  
# se dibujan cada una de las líneas sobre la imagen con fondo negro   
for line in lines:  
 for x1,y1,x2,y2 in line:   
 cv2.line(img\_lines, (x1, y1), (x2, y2), [255, 0, 0], 30)  
   
  
alpha = 1  
beta = 1  
gamma = 1  
img\_lane\_lines = cv2.addWeighted(img\_rgb\_fr1, alpha, img\_lines, beta, gamma)  
plt.imshow(img\_lane\_lines);



# Integración en una función

En esta función se realiza el seguiminto de las líneas detectadas en cada frame del video

def genera\_img\_hough\_lines(  
 img\_mask,  
 rho = 2, # resolución de rho en pixeles  
 theta = np.pi/180, # resolución de theta en radianes   
 threshold = 70, # mínimo número de votos para ser considerado una línea   
 min\_line\_len = 20, # mínimo número de pixeles para que se forme una línea  
 max\_line\_gap = 120, # máximo espacio en pixeles entre segmentos de línea   
):  
   
 lines = cv2.HoughLinesP(img\_mask, rho, theta, threshold, np.array([]), minLineLength=min\_line\_len, maxLineGap=max\_line\_gap)  
 # se crea un fondo negro del tamaño de la imagen con bordes  
 img\_lines = np.zeros((img\_mask.shape[0], img\_mask.shape[1], 3), dtype=np.uint8)  
 # se dibujan cada una de las líneas sobre la imagen con fondo negro   
 for line in lines:  
 for x1,y1,x2,y2 in line:   
 cv2.line(img\_lines, (x1, y1), (x2, y2), [255, 0, 0], 30)  
  
 # se retorna la imagen generada  
 return img\_lines

## Implementación de la función

cap = cv2.VideoCapture(video\_file)  
  
frames = []  
  
while(1):  
 try:  
 ret, image = cap.read()  
   
 # Convertir la imagen a escala de grises  
 img\_grey = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 #vertices = np.array([[(200,720),(550, 310), (610, 310), (1100,720)]], dtype=np.int32)  
 vertices = np.array([[(100,360),(275, 155), (305, 155), (550,360)]], dtype=np.int32)  
 img\_roi = np.zeros\_like(img\_grey\_fr1)  
 cv2.fillPoly(img\_roi, vertices, 255)  
   
 img\_blur = cv2.GaussianBlur(img\_grey,(3,3), 0, 0)  
 img\_canny = cv2.Canny(img\_blur, 40, 120)   
 img\_mask = cv2.bitwise\_and(img\_canny, img\_roi)  
  
 # imagen generada a partir de las lineas encontradas usando la transformada de Hough  
 img\_lines = genera\_img\_hough\_lines(img\_mask)  
  
 alpha = 1  
 beta = 1  
 gamma = 1  
 img\_lane\_lines = cv2.addWeighted(img\_rgb, alpha, img\_lines, beta, gamma)  
 img\_lane\_lines = cv2.cvtColor(img\_lane\_lines, cv2.COLOR\_RGB2BGR)  
  
 frames.append(img\_lane\_lines)  
 except Exception:  
 break  
  
#Establecer parámetros para guardar el video procesado  
width = int(cap.get(3))  
height = int(cap.get(4))  
  
filename = 'output/test2\_detect.mp4'  
  
fourcc = cv2.VideoWriter\_fourcc(\*'avc1')  
writer = cv2.VideoWriter(filename, fourcc, 25, (width, height))  
  
#Escribir cuadro por cuadro  
for frame in frames:  
 writer.write(frame)  
  
cap.release()  
writer.release()

[ERROR:0@17.958] global cap\_ffmpeg\_impl.hpp:2991 open Could not find encoder for codec\_id=27, error: Encoder not found  
[ERROR:0@17.959] global cap\_ffmpeg\_impl.hpp:3066 open VIDEOIO/FFMPEG: Failed to initialize VideoWriter

## Revisión del video

video2 = Video.from\_file(filename)  
video2

{"model\_id":"ef82f29e99944a44bc0d847692e45233","version\_major":2,"version\_minor":0}