CONDUCCIÓN AUTÓNOMA DE VEHÍCULOS
(SELF-DRIVING CAR)

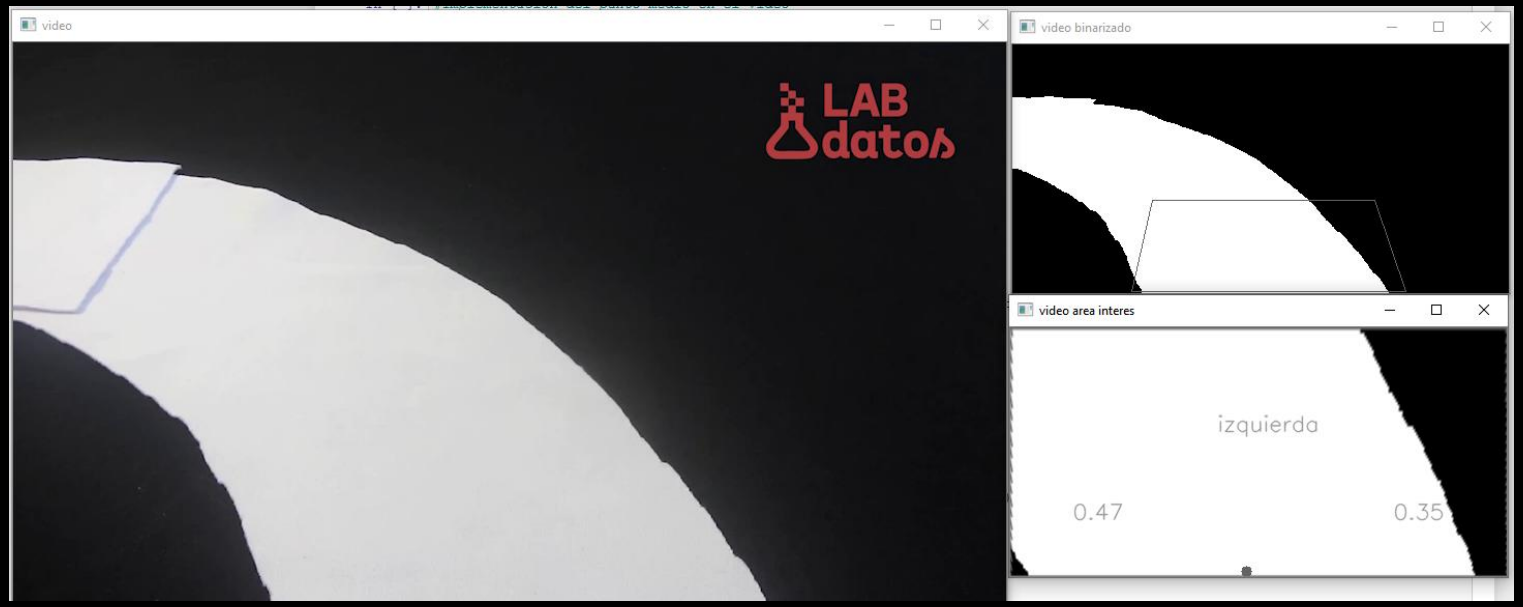
Enrique Camacho

SDC – 6 –
Algoritmo de dirección

UADY
UNIVERSIDAD AUTÓNOMA DE YUCATÁN

LAB datos

FACULTAD DE INGENIERÍA



Ejercicio 1:
Mostrar la dirección de giro

Siga todos los pasos de esta primera parte con el video que se adjunta y la notebook SDC_6_AlgoritmoDeDireccion.ipynb

Recapitulando las funciones

- Cargar librerías generales

```
import matplotlib.pyplot as plt
import numpy as np
import cv2 as cv
```

Cargar librerias

```
import matplotlib.pyplot as plt
import numpy as np
import cv2 as cv
```

Recapitulando las funciones

- Función de binarización

```
#Definir la función de binarización
def binarizacion(imagen):
    img = cv.cvtColor(imagen, cv.COLOR_BGR2RGB)
    img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    img_gauss = cv.GaussianBlur(img_gray,(3,3),0)
    thr, img_thr= cv.threshold(img_gauss ,160 ,255,cv.THRESH_BINARY)
    alto=img.shape[0]
    ancho=img.shape[1]
    ratio=0.2
    img_r = cv.resize(img_thr,(480,240), interpolation=cv.INTER_NEAREST)
    return(img_r)
```

Función de binarización

```
#Definir la función de binarización
def binarizacion(imagen):
    img = cv.cvtColor(imagen, cv.COLOR_BGR2RGB)
    img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    img_gauss = cv.GaussianBlur(img_gray,(3,3),0)
    thr, img_thr= cv.threshold(img_gauss ,160 ,255,cv.THRESH_BINARY)
    alto=img.shape[0]
    ancho=img.shape[1]
    ratio=0.2
    img_r = cv.resize(img_thr,(480,240), interpolation=cv.INTER_NEAREST)
    return(img_r)
```

Recapitulando las funciones

- Función de área de interés

```
#Poligono de área de interés
pts_poligono = np.array([[135, 150], [350, 150], [380, 238], [115, 238]], np.int32)
pts_poligono = pts_poligono.reshape((-1,1,2))
```

```
#Funcion de área de interés
def area_interes(imagen):
    pts1 = np.float32([[135, 150], [350, 150], [115, 238], [380, 238]])
    pts2 = np.float32([[0, 0], [480, 0], [0, 240], [480, 240]])
    matrix = cv.getPerspectiveTransform(pts1, pts2)
    img_warp = cv.warpPerspective(imagen, matrix, (480, 240))
    return (img_warp)
```


Función de área de interés

```
#Poligono de área de interés
pts_poligono = np.array([[135, 150], [350, 150], [380, 238], [115, 238]], np.int32)
pts_poligono = pts_poligono.reshape((-1,1,2))
```

```
#Funcion de área de interés
def area_interes(imagen):
    pts1 = np.float32([[135, 150], [350, 150], [115, 238], [380, 238]])
    pts2 = np.float32([[0, 0], [480, 0], [0, 240], [480, 240]])
    matrix = cv.getPerspectiveTransform(pts1, pts2)
    img_warp = cv.warpPerspective(imagen, matrix, (480, 240))
    return (img_warp)
```

Recapitulando las funciones

- Función de punto medio

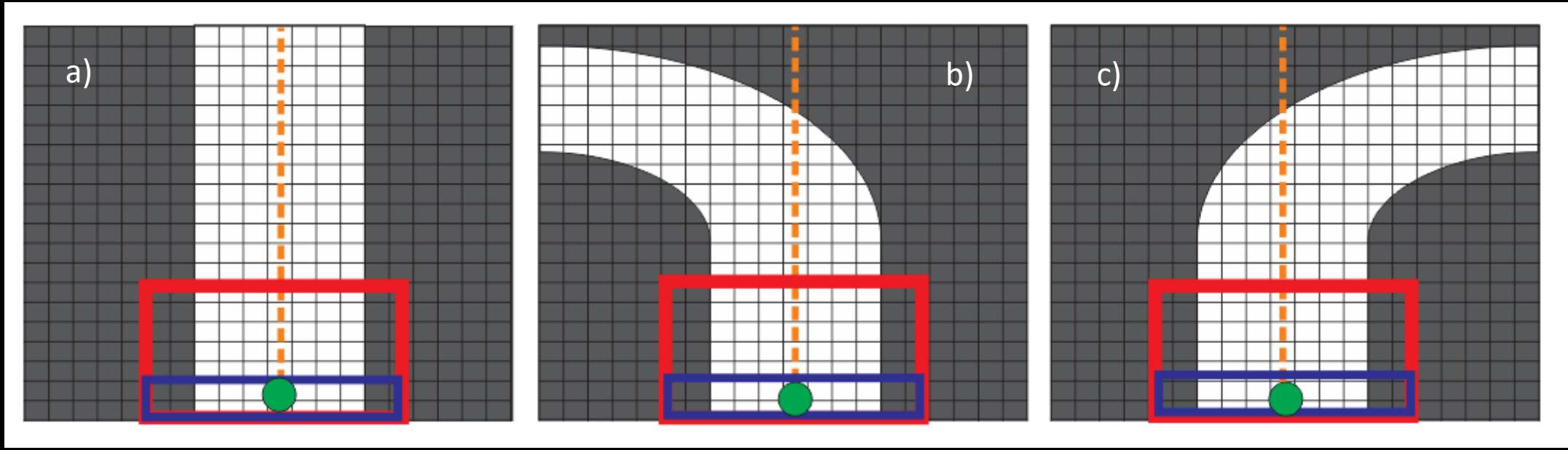
```
#Función para encontrar el punto medio
def punto_medio(imagen):
    img_cercana= imagen[220:, :]
    suma_columnas = img_cercana.sum(axis=0)
    x_pos = np.arange(len(suma_columnas))
    mid_point=int( np.dot(x_pos,suma_columnas) / np.sum( suma_columnas ) )
    return mid_point
```

Función de punto medio

```
#Función para encontrar el punto medio
def punto_medio(imagen):
    img_cercana= imagen[220:, :]
    suma_columnas = img_cercana.sum(axis=0)
    x_pos = np.arange(len(suma_columnas))
    mid_point=int( np.dot(x_pos,suma_columnas) / np.sum( suma_columnas ) )
    return mid_point
```

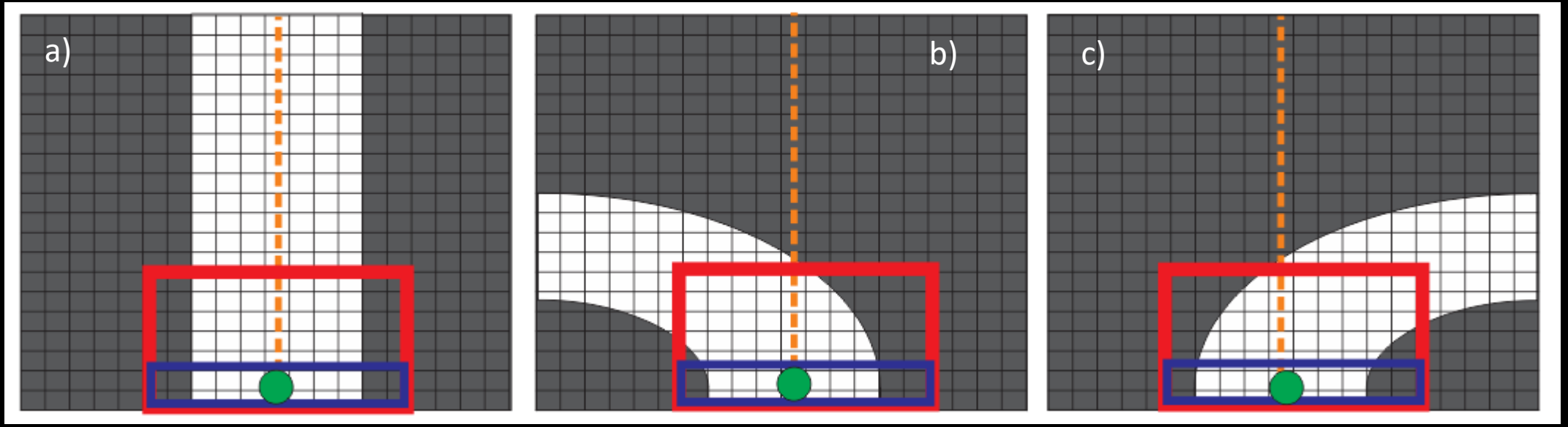
Método para encontrar la dirección de giro

- En este ejemplo se muestran 3 posibles escenarios, cuando hay una recta, una vuelta a la izquierda y una vuelta a la derecha
- En este escenario en el área de interés aun no se aprecia una vuelta en b) y c)
- En rojo esta el área de interés
- En azul la región que nos sirve para encontrar el punto medio (región cercana al observador)
- La circunferencia verde nos muestra el punto medio



Método para encontrar la dirección de giro

- En este punto el vehículo ha avanzado lo suficiente de manera que en el área de interés ya se aprecia una vuelta b) y c)



Método para encontrar la dirección de giro

- Analizamos los pixeles para el caso de las curvas
- Sumamos las columnas de la parte izquierda y derecha que resultan de la separación utilizando el punto medio

Suma_col_izquierda= [1020 , 1785 , 1785 , 1785]

Suma_col_derecha= [1785 , 1530 , 1020 , 0]

Suma_col_izquierda= [0 , 1020 , 1530 , 1785]

Suma_col_derecha= [1785 , 1785 , 1785 , 1020]

- Encontramos la suma de los valores de cada arreglo

Suma_izquierda= 1020 + 1785 + 1785 + 1785 = 7395

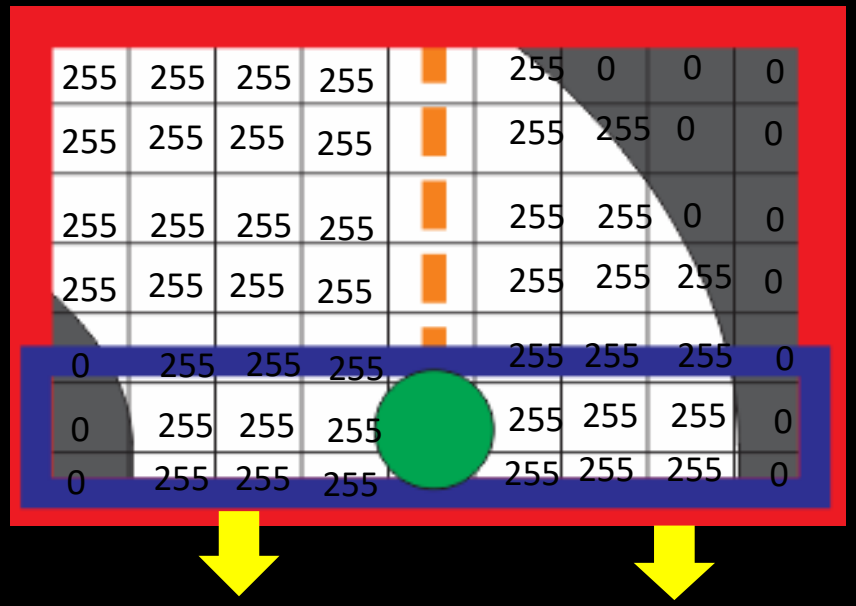
Suma_derecha= 1785 + 1530 + 1020 + 0 = 4335

Suma_izquierda= 0 + 1020 + 1530 + 1785 = 4335

Suma_izquierda= 1785 + 1785 + 1785 + 0 = 7395

Método para encontrar la dirección de giro

- Normalizamos con respecto a toda el área de interés, suponiendo que todos tienen pixeles en color blanco. Para nuestro ejemplo 7(alto) x 9 (ancho) x 255 (pixel en color blanco) = 16,065



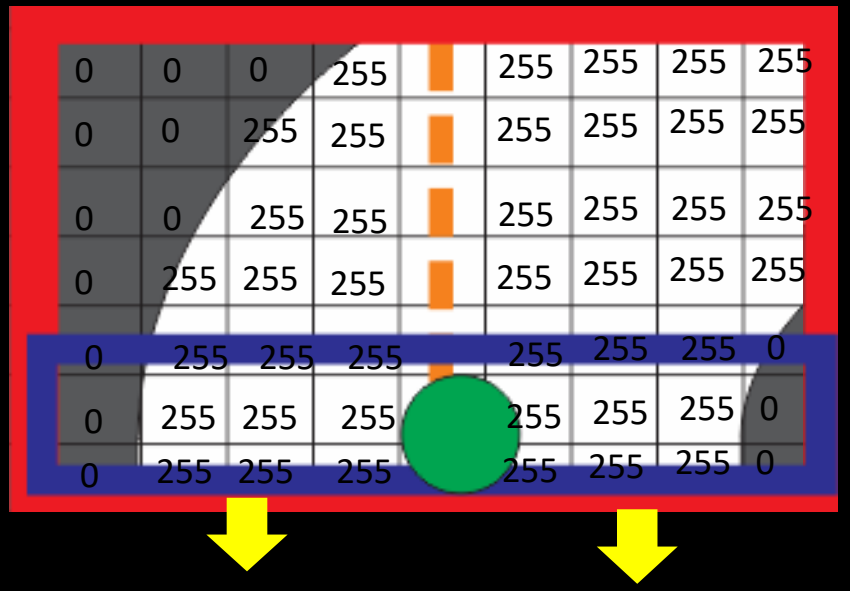
Suma_izquierda= 7395

Suma_derecha= 4335

Suma_izq_norm= 7395/16065 = 0.46

Suma_der_norm= 4335/16065 = 0.27

Delta= Suma_izq_norm – Suma_der_norm = 0.46 – 0.27 = 0.19



Suma_izquierda = 4335

Suma_izquierda= 7395

Suma_izq_norm= 4335/16065 = 0.27

Suma_der_norm= 7395/16065 = 0.46

Delta= Suma_izq_norm – Suma_der_norm = 0.27 – 0.46 = -0.19

- Para este ejemplo se podría decir que si Delta > 0.19 se tendrá una vuelta a la izquierda, si Delta es < -0.19 se tendrá una vuelta a la derecha, de otro modo el vehículo seguirá avanzando hacia adelante.
- *ATENCIÓN: El valor de Delta va a depender para cada configuración.*

Implementación del método para encontrar la dirección de giro

- Agregar un ejemplo de texto que servirá para ajustar y mostrar la dirección de giro

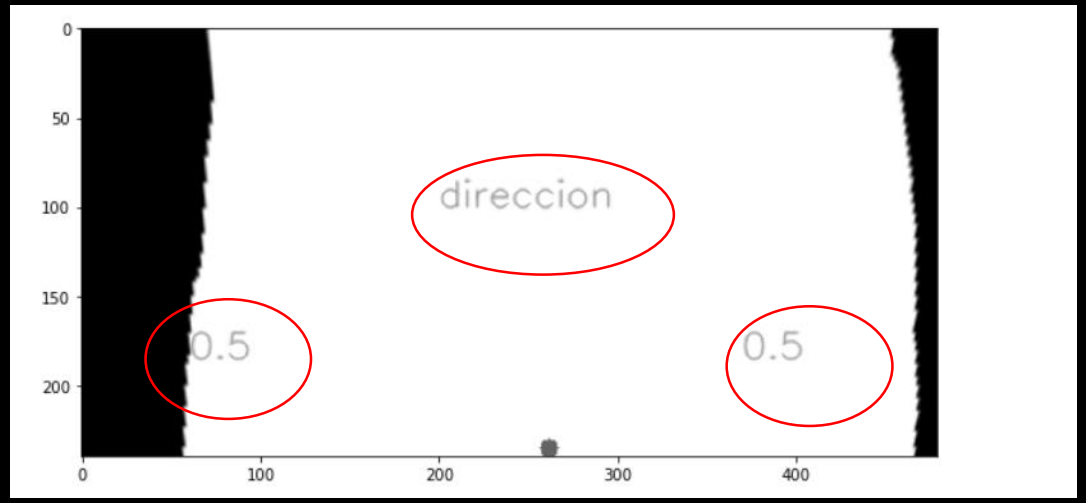
```
#Aplicamos todas las funciones
img = cv.imread('figuras/imagen_0.jpg')
img_bin = binarizacion(img)
img_interes=area_interes(img_bin)
mid_point= punto_medio(img_interes)

# textos
text1 = str(0.50)
text2 = str(0.50)
text3 = 'direccion'
# Tipo de fuente
font = cv.FONT_HERSHEY_SIMPLEX
# origen de cada texto
org1 = (60, 185)
org2 = (370, 185)
org3 = (200, 100)
# Tamaño
fontScale = 0.7
# Color de la fuente
color = (150, 150, 150)
# Grosor de la linea del texto
thickness = 1

# Usamos la función cv.putText() para agregar texto
cv.putText(img_interes, text1, org1, font, fontScale,
                color, thickness, cv.LINE_AA, False)
cv.putText(img_interes, text2, org2, font, fontScale,
                color, thickness, cv.LINE_AA, False)
cv.putText(img_interes, text3, org3, font, fontScale,
                color, thickness, cv.LINE_AA, False)

plt.figure(figsize=(10,7))
cv.circle(img_interes, (mid_point,235), 5, (100, 100,100 ), -1) ;
plt.imshow(img_interes,cmap='gray')
plt.show()
```

- Se definen las tres variables tipo str que contienen lo que se desea desplegar
- El tipo de fuente
- Las coordenadas de origen para desplegar el texto
- El tamaño de fuente
- El grosor de la fuente
- La función cv.putText agregara el texto en el área de interés.

Implementación del método para encontrar la dirección de giro

- Esta parte es para verificar que todo este configurado correctamente antes de colocarlo en el video

```
#Aplicamos todas las funciones
img = cv.imread('figuras/imagen_0.jpg')
img_bin = binarizacion(img)
img_interes=area_interes(img_bin)
mid_point= punto_medio(img_interes)

# textos
text1 = str(0.50)
text2 = str(0.50)
text3 = 'direccion'
# Tipo de fuente
font = cv.FONT_HERSHEY_SIMPLEX
# origen de cada texto
org1 = (60, 185)
org2 = (370, 185)
org3 = (200, 100)
# Tamaño
fontScale = 0.7
# Color de la fuente
color = (150, 150, 150)
# Grosor de la linea del texto
thickness = 1

# Usamos la función cv.putText() para agregar texto
cv.putText(img_interes, text1, org1, font, fontScale,
                color, thickness, cv.LINE_AA, False)
cv.putText(img_interes, text2, org2, font, fontScale,
                color, thickness, cv.LINE_AA, False)
cv.putText(img_interes, text3, org3, font, fontScale,
                color, thickness, cv.LINE_AA, False)

plt.figure(figsize=(10,7))
cv.circle(img_interes, (mid_point,235), 5, (100, 100,100 ), -1) ;
plt.imshow(img_interes,cmap='gray')
plt.show()
```



Implementación del método para encontrar la dirección de giro

- Definimos las sumas normalizadas, el valor de normalización es 240 (alto) x 480 (ancho) x 255 (pixel blanco)

```
#Funcion suma normalizada izquierda
def sum_izquierda(imagen, valor_punto_medio):
    return np.round(np.sum( imagen[:, :valor_punto_medio].sum(axis=0) )/(255*240*480),2)
```

```
#Funcion suma normalizada derecha
def sum_derecha(imagen, valor_punto_medio):
    return  np.round(np.sum( imagen[:, valor_punto_medio:].sum(axis=0) )/(255*240*480),2)
```

- En nuestra notebook una vez que definimos las sumas normalizadas y las evaluamos para asegurarnos que todo esta bien.

```
#Funcion suma normalizada izquierda
def sum_izquierda(imagen, valor_punto_medio):
    return np.round(np.sum( imagen[:, :valor_punto_medio].sum(axis=0) )/(255*240*480),2)
```

```
#Funcion suma normalizada derecha
def sum_derecha(imagen, valor_punto_medio):
    return  np.round(np.sum( imagen[:, valor_punto_medio:].sum(axis=0) )/(255*240*480),2)
```

```
sum_derecha(img_interes,mid_point)
```

```
0.42
```

```
sum_izquierda(img_interes,mid_point)
```

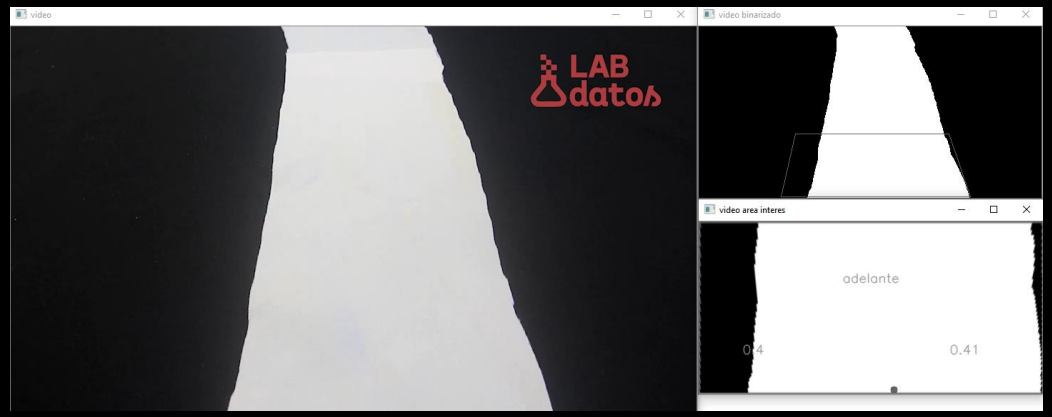
```
0.4
```

Implementación del método para encontrar la dirección de giro

```
#Implementación de la dirección de giro en el video
import time

video = cv.VideoCapture('videos/video_carretera_2.mp4')
while(video.isOpened()):
    ret, frame = video.read()
    if ret:
        cv.imshow("video", frame)
        img_bin=binarizacion(frame)
        cv.polylines(img_bin,[pts_poligono],True,(100,100,100))
        cv.imshow("video binarizado", img_bin)
        img_interes=area_interes(img_bin)
        mid_point = punto_medio(img_interes)
        valor_sum_izquierda=sum_izquierda(img_interes,mid_point)
        valor_sum_derecha=sum_derecha(img_interes,mid_point)
        cv.putText(img_interes, str(valor_sum_izquierda), org1, font, fontScale,
                color, thickness, cv.LINE_AA, False)
        cv.putText(img_interes, str(valor_sum_derecha), org2, font, fontScale,
                color, thickness, cv.LINE_AA, False)
        delta = valor_sum_izquierda - valor_sum_derecha
        if delta > 0.07 :
            movimiento = "izquierda"
        elif delta < -0.07 :
            movimiento = "derecha"
        else:
            movimiento = "adelante"
        cv.putText(img_interes, movimiento, org3, font, fontScale,
                color, thickness, cv.LINE_AA, False)
        cv.circle(img_interes, (mid_point, 235), 5, (100,100,100), -1) ;
        cv.imshow("video area interes", img_interes)
        time.sleep(0.02)
        if cv.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
video.release()
cv.destroyAllWindows()
```

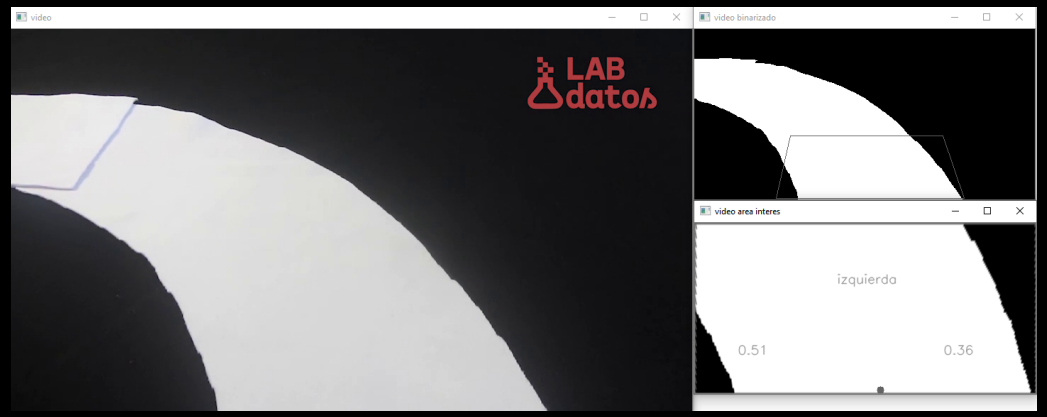
- Se agrega la librería time que servirá para llevar a cabo una pausa entre cada frame
- Valor de las sumas normalizadas
- Se despliegan los valores
- Se calcula el valor de *delta*
- Dependiendo del valor, la variable *delta* adquiere un valor string, ya sea “izquierda”, “derecha” o “adelante”
- Se despliega el texto de la variable *movimiento*
- time.sleep nos permite una pausa entre cada frame. Se puede comentar en caso de no ser requerido.

Implementación del método para encontrar la dirección de giro

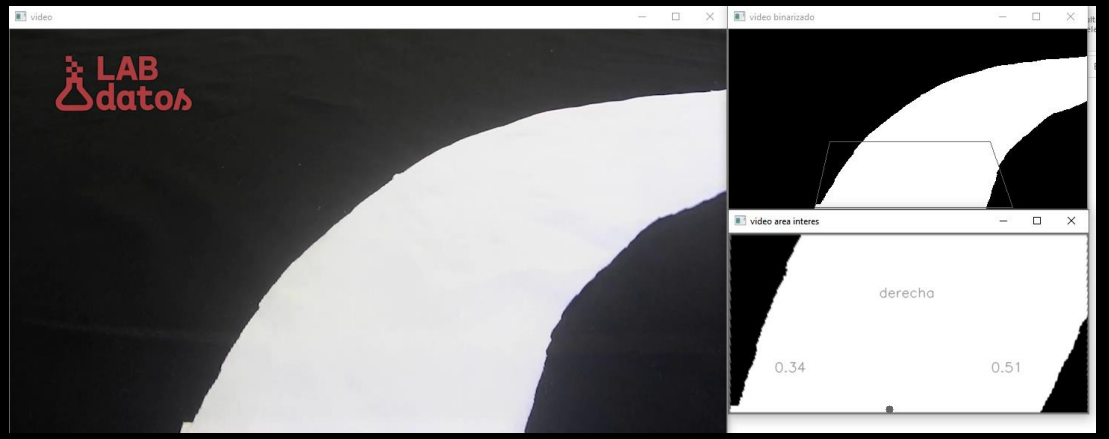


Sin curva

Con curva
y giro a la izquierda



Con curva
y giro a la derecha

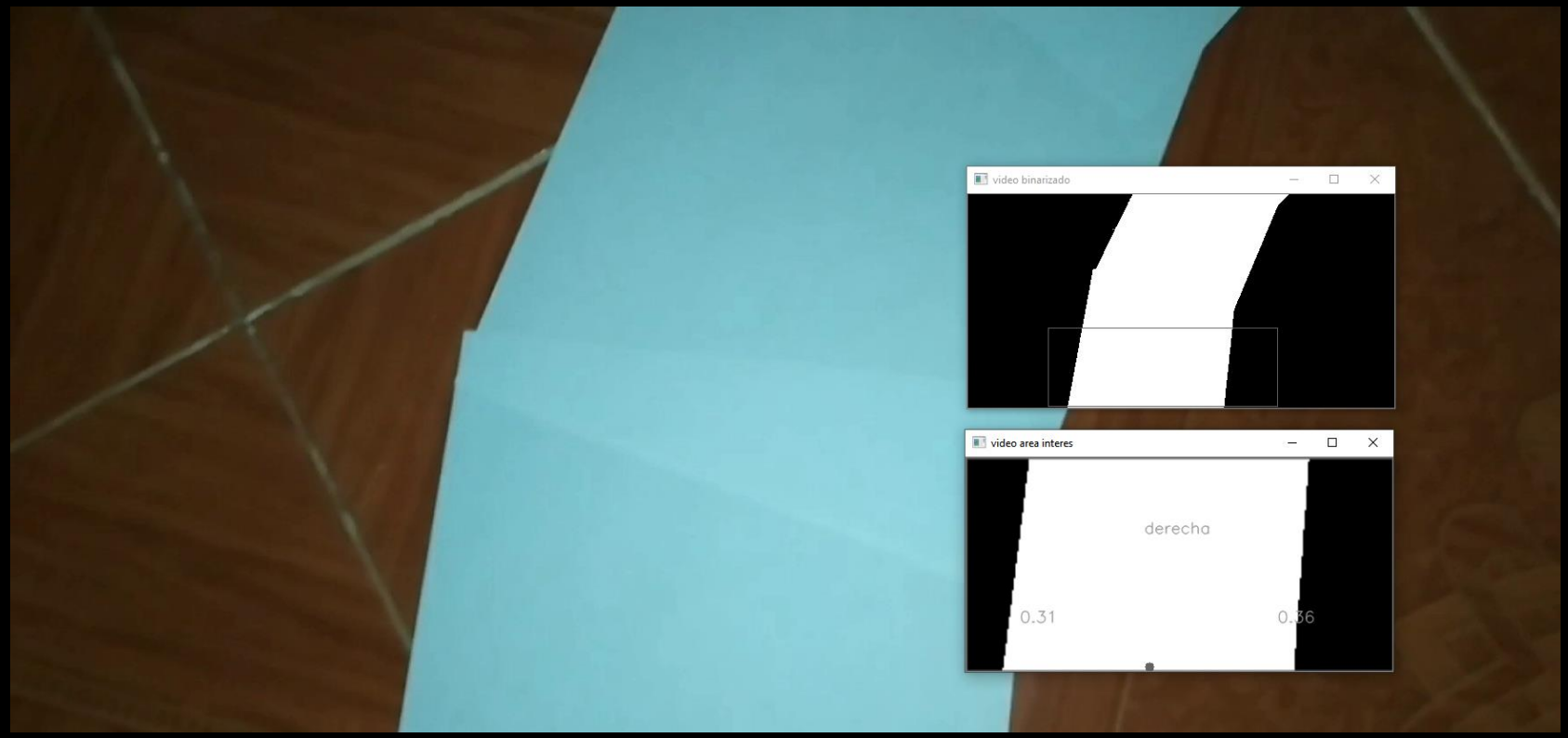


Implementación del método para encontrar la dirección de giro

- El valor del parámetro *delta* se ajusta dependiendo de los valores que aparecerán y depende de cada configuración de video.

Instrucciones

- Para esta primera parte guarde el notebook con el siguiente nombre: SDC_6A_Nombre_Apellido.ipynb

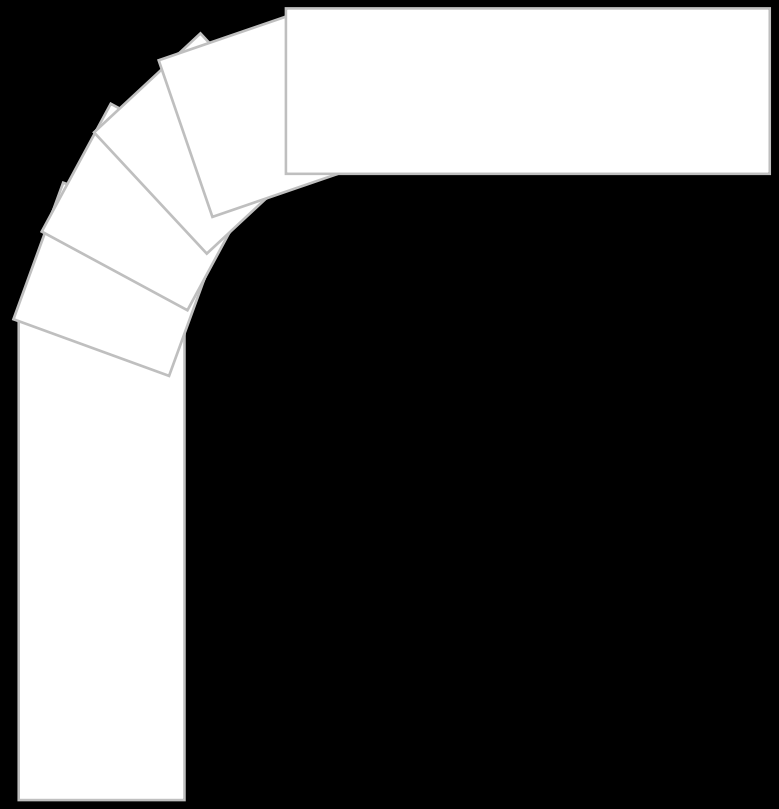


Ejercicio 2:
Mostrar la dirección de giro en su video

Realice la implementación en su propio video

Implementación del ejercicio 1 en su video

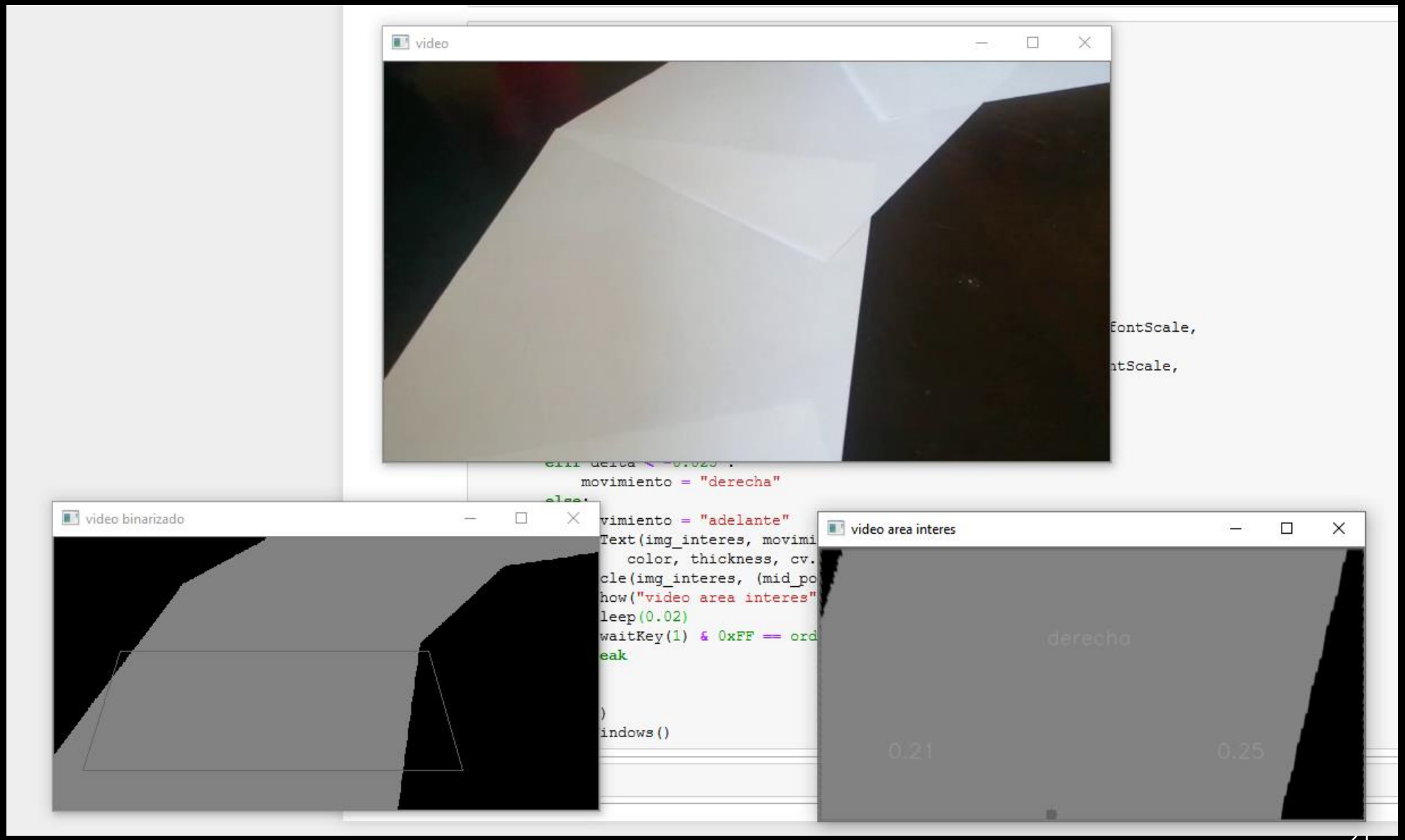
- Utilice el video que hizo en la actividad SDC-5
- Realice todos los pasos mostrados en la primera parte en un nuevo notebook de Jupyter.



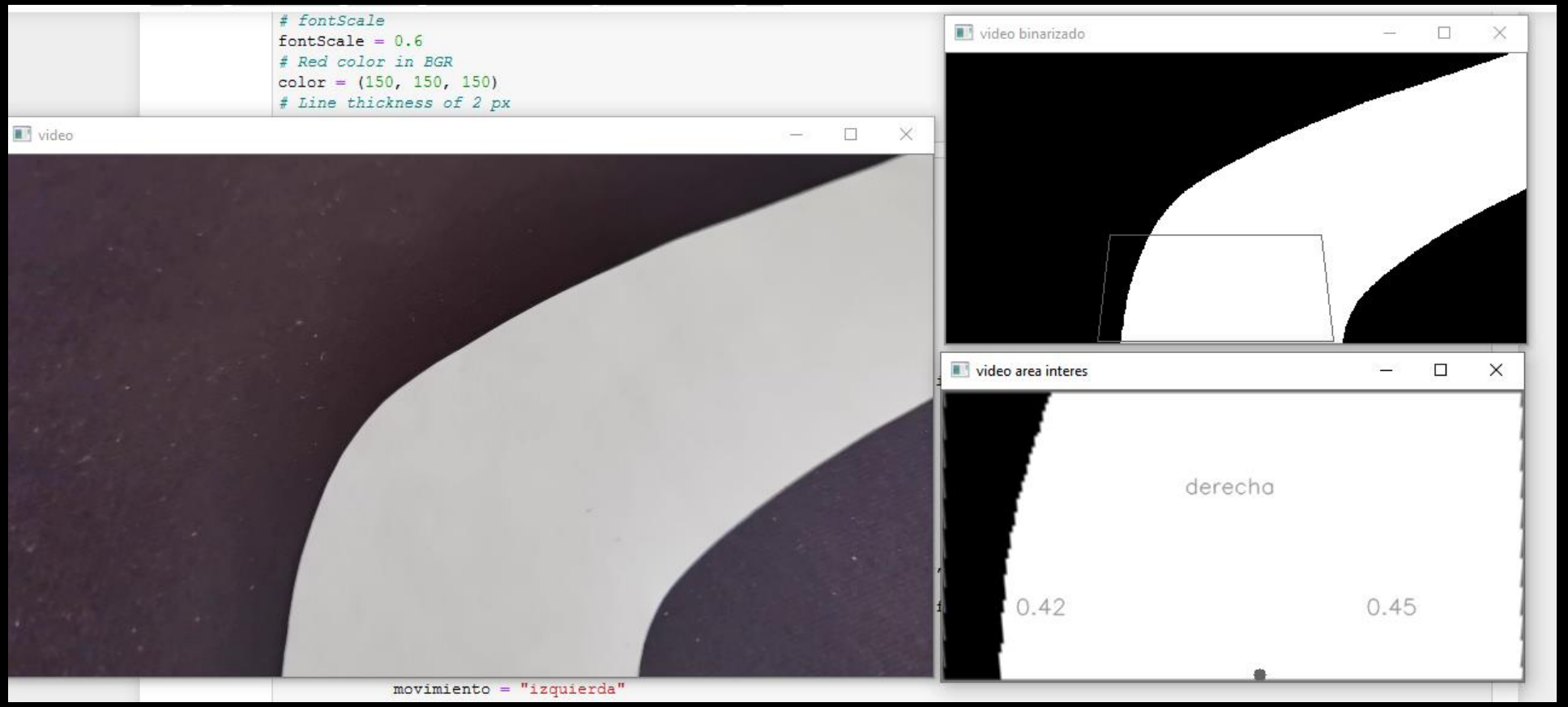
Instrucciones

- Para esta segunda parte guarde el notebook con el siguiente nombre: SDC_6B_Nombre_Apellido.ipynb
- Comprima toda su carpeta en un archivo .ZIP con el nombre SDC_6_Nombre_Apellido.ZIP (debe incluir los dos notebooks y su video).
- En caso de que sea muy grande el archivo ZIP, subirlo a su Google drive o alguna plataforma en la nube y compartirme el link.

Ejemplos


video
video binarizado
video area interes
movimiento = "derecha"
movimiento = "adelante"
fontScale,
derecho
0.21
0.25

Ejemplos


fontScale = 0.6
color = (150, 150, 150)
video
video binarizado
video area interes
derecha
0.42
0.45
movimiento = "izquierda"

Notebook

Cargar librerias

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import cv2 as cv
```

Función de binarización

In [2]:
```
#Definir la función de binarización
def binarizacion(imagen):
    img = cv.cvtColor(imagen, cv.COLOR_BGR2RGB)
    img_gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    img_gauss = cv.GaussianBlur(img_gray,(3,3),0)
    thr, img_thr= cv.threshold(img_gauss ,160 ,255,cv.THRESH_BINARY)
    alto=img.shape[0]
    ancho=img.shape[1]
    ratio=0.2
    img_r = cv.resize(img_thr,(480,240), interpolation=cv.INTER_NEAREST)
    return(img_r)
```

Función de área de interés

In [3]:
```
#Poligono de área de interés
pts_poligono = np.array([[135, 150], [350, 150], [380, 238], [115, 238]], np.int32)
pts_poligono = pts_poligono.reshape((-1,1,2))
```

In [4]:
```
#Funcion de área de interés
def area_interes(imagen):
    pts1 = np.float32([[135, 150], [350, 150], [115, 238], [380, 238]])
    pts2 = np.float32([[0, 0], [480, 0], [0, 240], [480, 240]])
    matrix = cv.getPerspectiveTransform(pts1, pts2)
    img_warp = cv.warpPerspective(imagen, matrix, (480, 240))
    return (img_warp)
```

Función de punto medio

In [5]:
```
#Función para encontrar el punto medio
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

Función de punto medio

In [5]:
```
#Función para encontrar el punto medio
def punto_medio(imagen):
    img_cercana= imagen[220:, :]
    suma_columnas = img_cercana.sum(axis=0)
    x_pos = np.arange(len(suma_columnas))
    mid_point=int( np.dot(x_pos,suma_columnas) / np.sum( suma_columnas ) )
    return mid_point
```

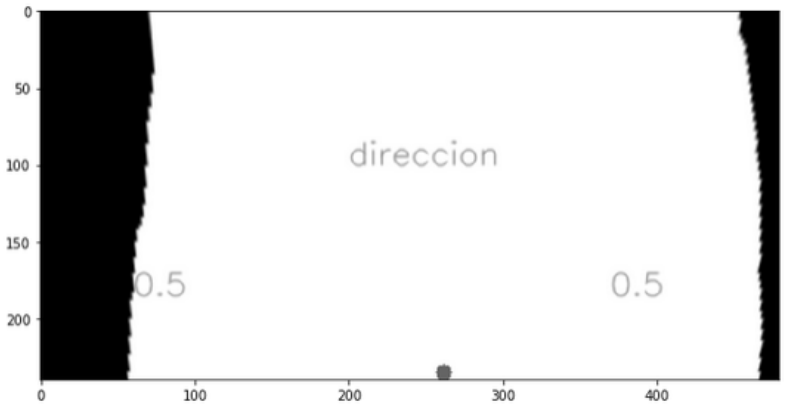
Textos de apoyo

In [6]:
```
#Aplicamos todas las funciones
img = cv.imread('figuras/imagen_0.jpg')
img_bin = binarizacion(img)
img_interes=area_interes(img_bin)
mid_point= punto_medio(img_interes)

# textos
text1 = str(0.50)
text2 = str(0.50)
text3 = 'direccion'
# Tipo de fuente
font = cv.FONT_HERSHEY_SIMPLEX
# origen de cada texto
org1 = (60, 185)
org2 = (370, 185)
org3 = (200, 100)
# Tamaño
fontScale = 0.7
# Color de la fuente
color = (150, 150, 150)
# Grosor de la linea del texto
thickness = 1

# Usamos la función cv.putText() para agregar texto
cv.putText(img_interes, text1, org1, font, fontScale,
                  color, thickness, cv.LINE_AA, False)
cv.putText(img_interes, text2, org2, font, fontScale,
                  color, thickness, cv.LINE_AA, False)
cv.putText(img_interes, text3, org3, font, fontScale,
                  color, thickness, cv.LINE_AA, False)

plt.figure(figsize=(10,7))
cv.circle(img_interes, (mid_point,235), 5, (100, 100,100 ), -1) ;
plt.imshow(img_interes,cmap='gray')
plt.show()
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
cv.putText(img_interes, text1, org1, font, fontScale,
                color, thickness, cv.LINE_AA, False)
cv.putText(img_interes, text2, org2, font, fontScale,
                color, thickness, cv.LINE_AA, False)
cv.putText(img_interes, text3, org3, font, fontScale,
                color, thickness, cv.LINE_AA, False)

plt.figure(figsize=(10,7))
cv.circle(img_interes, (mid_point,235), 5, (100, 100,100 ), -1) ;
plt.imshow(img_interes,cmap='gray')
plt.show()
```



In [7]:
```
#Funcion suma normalizada izquierda
def sum_izquierda(imagen, valor_punto_medio):
    return np.round(np.sum( imagen[:, :valor_punto_medio].sum(axis=0) )/(255*240*480),2)
```

In [8]:
```
#Funcion suma normalizada derecha
def sum_derecha(imagen, valor_punto_medio):
    return  np.round(np.sum( imagen[:, valor_punto_medio:].sum(axis=0) )/(255*240*480),2)
```

In [9]:
```
sum_derecha(img_interes,mid_point)
```

Out[9]: 0.42

In [10]:
```
sum_izquierda(img_interes,mid_point)
```

Out[10]: 0.41

Mostrando la dirección de giro

In [11]:
```
#Implementacion de la dirección de giro en el video
import time

video = cv.VideoCapture('videos/video_carretera_v2.mp4')
```

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

In [9]:
```
sum_derecha(img_interes,mid_point)
```

Out[9]: 0.42

In [10]:
```
sum_izquierda(img_interes,mid_point)
```

Out[10]: 0.41

Mostrando la dirección de giro

In [11]:
```
#Implementacion de la dirección de giro en el video
import time

video = cv.VideoCapture('videos/video_carretera_v2.mp4')
while(video.isOpened()):
    ret, frame = video.read()
    if ret:
        cv.imshow("video", frame)
        img_bin=binarizacion(frame)
        cv.polylines(img_bin,[pts_poligono],True,(100,100,100))
        cv.imshow("video binarizado", img_bin)
        img_interes=area_interes(img_bin)
        mid_point = punto_medio(img_interes)
        valor_sum_izquierda=sum_izquierda(img_interes,mid_point)
        valor_sum_derecha=sum_derecha(img_interes,mid_point)
        cv.putText(img_interes, str(valor_sum_izquierda), org1, font, fontScale,
                color, thickness, cv.LINE_AA, False)
        cv.putText(img_interes, str(valor_sum_derecha), org2, font, fontScale,
                color, thickness, cv.LINE_AA, False)
        delta = valor_sum_izquierda - valor_sum_derecha
        if delta > 0.07 :
            movimiento = "izquierda"
        elif delta < -0.07 :
            movimiento = "derecha"
        else:
            movimiento = "adelante"
        cv.putText(img_interes, movimiento, org3, font, fontScale,
                color, thickness, cv.LINE_AA, False)
        cv.circle(img_interes, (mid_point, 235), 5, (100,100,100), -1) ;
        cv.imshow("video area interes", img_interes)
        time.sleep(0.02)
        if cv.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
video.release()
cv.destroyAllWindows()
```

In [ ]: