

**TE3002B.561**

**Implementación de Robótica Inteligente:**

**Módulo 2 - Visión por Computadora**

# **Actividad 3 - Reconocimiento de Señales de Tránsito**

| Daniel Nava Mondragón | A01661649 | A01661649@tec.mx |
| --- | --- | --- |
| Karina Ruiz Tron | A01656073 | A01656073@tec.mx |

Fecha de entrega: 21 de mayo de 2023

Profesor:

Diego López Bernal

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Ciudad de México

1. Importación de librería: Se importó la librería OpenCV para el procesamiento de las imágenes.
2. Declaración de funciones:
   1. Se declara una función con el propósito de buscar un valor determinado en una lista, y si encuentra dicho valor numérico o uno mayor, devuelve el índice en donde se encuentra en la lista. Si no encuentra un valor que cumpla los requisitos devuelve un -1.

def encontrar\_indice\_mayor\_igual(lista, valor\_limite):

for indice, valor in enumerate(lista):

if valor >= valor\_limite:

return indice

return -1



* 1. Se declara una función que recibe la ruta de almacenamiento de las imágenes y devuelve la imagen después de realizarle los siguientes procesos
     1. Se lee la imagen y se guarda en una variable
     2. Se convierte la imágen a una escala de grises
     3. Se le aplica la función de ruido Gaussiano (Gaussian Blur).
     4. Posteriormente, se aplica el threshold **Otsu** para convertir las imágenes a imagenes binarias, con valores o de blanco o de negro. Esto ayudará a definir aquellos elementos que se busca resaltar para encontrar la señal correspondiente.

def cargar\_y\_procesar\_imagen(ruta):

img = cv2.imread(ruta)

img = cv2.cvtColor(img, cv2.COLOR\_RGB2GRAY)

img = cv2.GaussianBlur(img, (5, 5), 0)

\_, img = cv2.threshold(img, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

return img



1. Variables a utilizar:
   1. Se crea una lista de 7 de largo donde guarda en cada índice las imágenes de las señales procesadas por la función *cargar\_y\_procesar\_imagen*
   2. Se declaró una listas del mismo largo, la cual contiene los nombres de las señales
   3. Se declaró una tercera lista de la misma longitud con sus valores iniciales en cero
   4. Se declaró una variable llamada iteración con la finalidad de contar la cantidad de frames en los cuales va a procesar la identificación
2. Se inicializa la cámara de la computadora
3. **While True**
   1. Guarda cada frame del video en una variable y la procesa con los siguientes funciones de OpenCV:
      1. Se convierte el frame a una escala de grises
      2. Al igual que con las imágenes de las señales se generó un ruido gaussiano
      3. Finalmente se le aplicó también un threshold de tipo otsu para definir la imagen con solo valores de unos y ceros.
   2. Se crea un objeto tipo **SIFT**
      1. Se obtienen con la función **detectAndCompute** de dicho objeto sus keypoints y sus descriptors. Los keypoints son puntos clave de la imagen, mientras que los descriptores almacena información alrededor de los keypoints con la finalidad de realizar comparaciones

sift = cv2.SIFT\_create()

keypoints, descriptors = sift.detectAndCompute(img,None)



* 1. Se crea una variable denominada bf con la función **BFMatcher** con la finalidad de hacer las comparaciones de los keypoints posteriormente.

bf = cv2.BFMatcher(cv2.NORM\_L1, crossCheck=True)

* 1. Se itera toda la lista de las imágenes:
     1. Se obtienen los keypoints y descriptores de la imagen
     2. Se obtienen las coincidencias de la imagen de la cámara con la imagen de la lista en cuestión
     3. Se guarda en la lista de puntos la cantidad de coincidencias que se encontraron entre la referencia y la señal
  2. Una vez obtenida la lista con las coincidencias del frame con cada señal se obtiene el valor máximo de dicha lista. Reflejando así las coincidencias con cada una de las imágenes.
  3. Se extrae el índice donde se encuentra dicho valor
  4. Posteriormente se aumenta en uno la lista de conteo de acuerdo al índice correspondiente.
  5. Se aumenta el contador de iteraciones
  6. Se llama a la función de encontrar\_indice\_mayor\_igual, la cual recibe como entrada la lista de conteo y un valor previamente asignado en 20. Esta lista nos devuelve que índice contiene un valor mayor o igual a 20. Esto con la finalidad de que si identifica la misma imagen durante veinte frames seguidos es dicha imagen.
  7. Si supera un límite de iteraciones:
     1. Imprime que no hay señal
     2. La variable de iteraciones la vuelve cero
     3. La lista de conteo la vuelve también cero
  8. sí no:
     1. Si el índice es diferente de -1:
        1. Obtiene el valor de la lista de nombres con dicho índice y lo imprime
        2. La variable de iteración se vuelve cero
        3. La lista de conteo se vuelve cero
  9. Imprime los frames del fideo

1. Cierra todas las ventanas abiertas