Actividad 3: Signal Identification

Curso: TE3002B.561 Implementación de Robótica Inteligente
Profesor: Diego López Bernal
Alumnos:

- Jennifer Lizeth Avendaño Sánchez - A01656951
- Juan Francisco García Rodríguez - A01660981

Fecha: 18 de mayo de 2024

1. Documentación del proceso

1. Se cargan las imágenes de referencia de las señales de tránsito en escala de grises utilizando `cv2.imread`.

```
    right_arrow_img = cv2.imread('turnright.jpg', cv2.IMREAD_GRAYSCALE)
    left_arrow_img = cv2.imread('turnleft.jpg', cv2.IMREAD_GRAYSCALE)
    give_way_img = cv2.imread('giveaway.jpg', cv2.IMREAD_GRAYSCALE)
    work_in_progress_img = cv2.imread('workinprogress.jpg', cv2.IMREAD_GRAYSCALE)
    forward_arrow_img = cv2.imread('straigth.jpg', cv2.IMREAD_GRAYSCALE)
    turn_around_arrow_img = cv2.imread('turnaround.jpg', cv2.IMREAD_GRAYSCALE)
    stop_img = cv2.imread('stop.jpg', cv2.IMREAD_GRAYSCALE)
```

2. Se crea un objeto SIFT utilizando `cv2.SIFT_create()`.
3. Para cada imagen de referencia, se detectan los *keypoints* y se calculan los descriptores.

```
    kp1, des1 = sift.detectAndCompute(right_arrow_img, None)
    kp2, des2 = sift.detectAndCompute(left_arrow_img, None)
    kp3, des3 = sift.detectAndCompute(give_way_img, None)
    kp4, des4 = sift.detectAndCompute(work_in_progress_img, None)
    kp5, des5 = sift.detectAndCompute(forward_arrow_img, None)
    kp6, des6 = sift.detectAndCompute(turn_around_arrow_img, None)
    kp7, des7 = sift.detectAndCompute(stop_img, None)
```

4. Se inicia la captura de video desde la cámara de la computadora utilizando `cv2.VideoCapture(0)`.
5. Se define el umbral mínimo de coincidencias para considerar que una señal ha sido detectada.
6. Se crea un buffer para almacenar los resultados de las últimas N detecciones; para este caso se determinó de 7 elementos.
7. Se inicializa un bucle, que comienza capturando y procesando los *frames* del video.
8. Se establece un contador para las coincidencias de cada señal.

```
    match_counts = {
    'turnright': 0,
    'turnleft': 0,
    'give_way': 0,
    'work_in_progress': 0,
    'straight': 0,
    'turnaround': 0,
    'stop': 0
    }
```

9. Se comparan los descriptores del *frame* actual con los descriptores de las imágenes de referencia utilizando el algoritmo de coincidencia *BFMatcher* y un umbral para el filtrado de coincidencias.
10. Se identifica la señal que tiene la mayor cantidad de coincidencias en el *frame* actual.

11. Se actualiza el buffer con el resultado de la señal detectada.
12. Se calcula la moda del buffer para determinar la señal más común en los últimos frames.
13. Si la cantidad de coincidencias de la mejor señal supera el umbral mínimo de coincidencias, se considera que la señal ha sido detectada y se muestra en la imagen la etiqueta de dicha señal.
14. Se muestra el *frame* procesado con la señal detectada.
15. Se cierra la captura de video y las ventanas si se presiona la tecla "x".
16. Se liberan los recursos de la cámara y se cierran todas las ventanas de OpenCV.