1 Presentación del problema

La 1ª práctica de Robótica consistirá en la simulación y diseño de un vehículo autónomo usando el entorno de simulación *Webots* y el framework *ROS2*. El objetivo de la práctica será conseguir simular un vehículo con la capacidad de **seguir la carretera** y **reaccionar a distintas señales de tráfico**.
Para ello se proporciona al alumno un mundo de Webots llamado *city_traffic.wbt*. Dicho mundo contiene toda la información necesaria para poder diseñar el robot autónomo y **NO DEBE SER MODIFICADO**. Asimismo la lógica del control de los sensores y actuadores del vehículo deberá ser desarrollada con ROS2 y **no como controladores de Webots**.

2 Mundo simulado en Webots

El mundo proporcionado cuenta con un escenario de una carretera urbana, con marcas viales en mitad de la calzada, que el coche autónomo usará como referencia para seguir su recorrido. También existirán ciertas señales viales que deberán cambiar el comportamiento del vehículo.
En cuanto al robot del vehículo, este es del tipo *CitroenCZero*, un tipo de robot estándar de Webots. En la Figura 1 se puede observar el coche simulado, así como las proyecciones de sus cámaras.

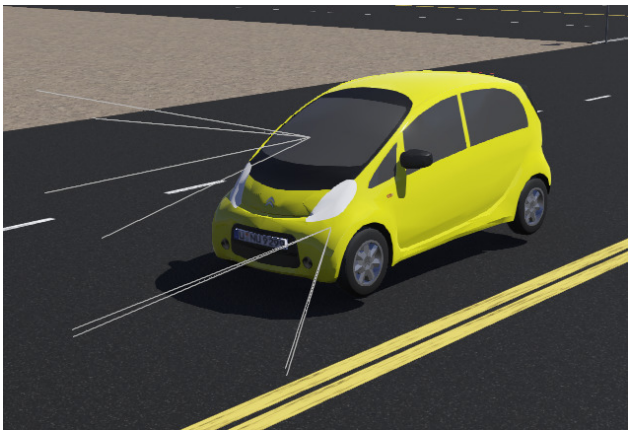

Figura 1: Coche simulado en Webots.

En concreto, el coche cuenta con **2 cámaras** RGB para recibir información de su entorno:

- *"car_camera"*: Es una cámara que apunta de manera frontal al coche, de modo que será la encargada de recibir información de las **señales de tráfico**.
- *"road_camera"*: Es una cámara que apunta a las **marcas viales** de la carretera. Su resolución es mucho más estrecha que la anterior, ya que su objetivo es únicamente detectar el centro de la carretera.

Las imágenes capturadas por ambas cámaras se pueden observar en la Figura 2.

(a) Imagen de ejemplo capturadas por la *"car_camera"*.

(b) Imagen de ejemplo capturadas por la *"road_camera"*.

Figura 2: Imágenes capturadas por las cámaras del vehículo simulado.

En cuanto a la estructura dentro de webots, se puede observar en la Figura 3 el árbol de componentes.

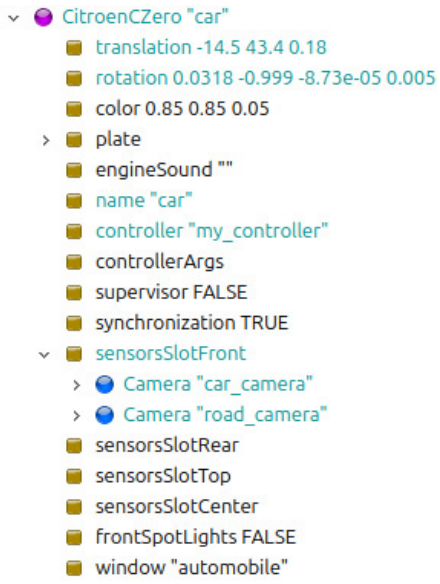

Figura 3: Configuración del coche en Webots.

3 Lógica de control del vehículo

El control del vehículo se dividirá en 2 acciones principales: seguimiento de la carretera y detección de señales. En esta sección se definen los algoritmos para realizar ambos procesos.

3.1 Seguimiento de la carretera

Para que el coche realice un seguimiento de la carretera cuenta con la ya mencionada *"road_camera"*. Esta tiene una proyección lateral que hace que cuando el vehículo se encuentra en el **centro del carril** las marcas viales están en el **centro de la imagen**. Por tanto el objetivo del robot es **mantener** en el **centro de la imagen las marcas viales**.
Este proceso se divide asimismo en dos subprocesos:

- La detección de la posición de las marcas viales dentro de la imagen capturada por la *"road_camera"* se realizará procesando la imagen de la siguiente manera (se recomienda el uso de numpy para el procesamiento de la imagen):

- Se captura la imagen de resolución 512x16x3 píxeles.
- Se hace una **media de valores** de cada columna, para los 3 canales. Obteniendo un vector unidimensional de 512 valores.
- El centro de la carretera corresponde con el **máximo valor** del vector de 512 valores.

La Figura 4 contiene el esquema lógico para encontrar las marcas viales en la imagen capturada.

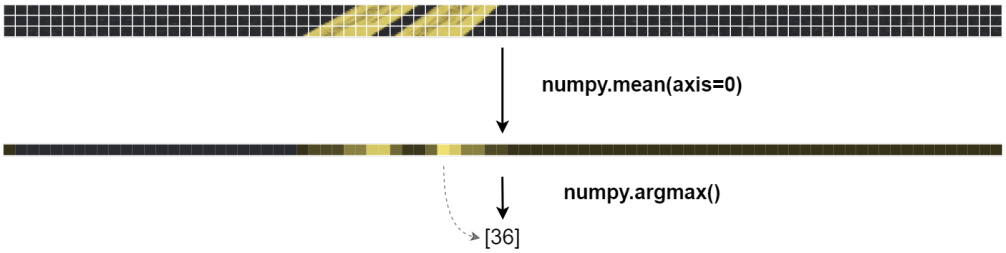

Figura 4: Algoritmo de búsqueda de las marcas viales.

- Por otro lado, una vez detectado el **centro de la carretera capturado**, y junto al **centro de la carretera ideal** (que es el centro de la imagen), se deberá controlar al coche con un controlador borroso o clásico para que mantenga el rumbo correcto, haciendo que el centro de la imagen sea lo más cercano posible a las marcas viales.

3.2 Reconocimiento de señales

El reconocimiento de las señales viales por parte del coche definirá el comportamiento del mismo, ya que dependiendo de la **última señal observada**, el coche debe actuar de una manera u otra. Para ello se utiliza la cámara *"car_camera"* y se realizará reconocimiento de objetos sobre las imágenes obtenidas.

- La detección de las señales se realizará a través del uso de la librería de python *opencv*. Para ello es necesario definir un pequeño *dataset* de imágenes de referencia, pues la detección se realizará por comparación. En concreto, se seguirá el siguiente proceso:
  - Se captura la imagen de resolución 512x256 píxeles.
  - Se obtiene el conjunto de imágenes de señales de referencia.
  - Se buscan coincidencias de las imágenes de referencia dentro de la imagen capturada. Para ello se usa la función *matchTemplate* de opencv (`https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html`).
  - A través del valor de la máxima confianza obtenido con *matchTemplate* se decide si alguna señal está presente en la imagen. Para ello se debe definir un valor de threshold de manera empírica.

  La Figura 5 contiene el esquema de funcionamiento del procedimiento de reconocimiento de señales.

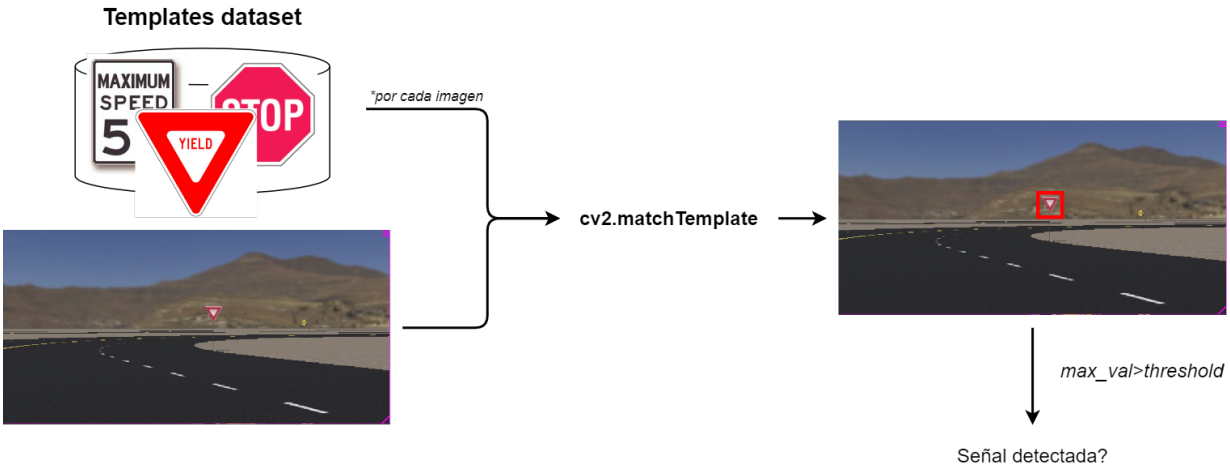

Figura 5: Algoritmo de búsqueda de las señales de tráfico.

Cabe recalcar que la función *matchTemplate* buscará coincidencias de la imagen de referencia (template) en la imagen completa. Para ello es especialmente sensible al **tamaño de la imagen de referencia**. Por tanto será deseable usar imágenes con distintos tamaños, para asegurar el reconocimiento de señales a distinto tamaño. Se proporciona un dataset de imágenes de referencia, pero el alumno es libre de aumentarlo como considere necesario.

- Dependiendo de la señal detectada el coche debe realizar una de las siguientes acciones:
  - **Señal de velocidad máxima**: El coche reducirá su velocidad a la velocidad máxima.
  - **Señal de ceda el paso** (yield): El coche reducirá su velocidad a mitad de la velocidad máxima.
  - **Señal de STOP**: El coche parará por completo durante 1 segundo y reanudará su marcha (no es necesario que pare justo a la altura justa del STOP). La velocidad máxima podrá definirla el alumno a su preferencia.

4 Evaluación

La evaluación de la práctica se realizará a través del código desarrollado siguiendo los parámetros previamente definidos. Debe ser subida al Moodle de la asignatura con fecha anterior a la fecha máxima de entrega. No se permitirá la copia entre prácticas y será motivo de suspenso.

4.1 Entregables para la evaluación

Para una correcta evaluación (y estandarización de las prácticas) se pide que el código desarrollado se realice en un paquete llamado *car_pkg*, dentro del cual se encuentren las carpetas *world*, *launch*, *car_pkg* así como el fichero setup.py[1]. Para ello, el sistema de ficheros sería el siguiente:

```
car_pkg/
├── car_pkg/
│   └── ficheros .py de los nodos de ROS2
├── launch/
│   └── launch.py
├── setup.py
└── world/
    └── city_traffic.wtb
```

[1] Es decir, la configuración estándar de desarrollo de un paquete en ROS2

Por tanto, cada entrega deberá contar con los archivos .py de los nodos de ROS2, así como el archivo launch.py y el setup.py.
Asimismo el alumno debe entregar una **memoria .pdf** con una explicación de la implementación realizada, incluyendo capturas de códigos del código fuente relevante, así como diagramas de comunicación y paso de mensajes entre los distintos nodos.