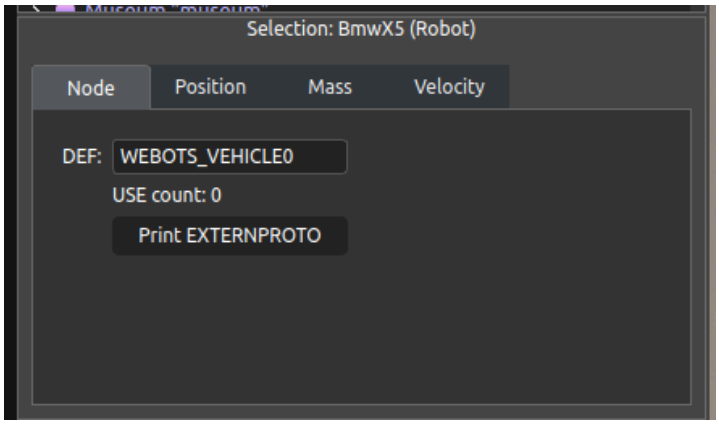
Proyecto Final

MR4010 - Navegación Autónoma

Las siguientes instrucciones indican los pasos necesarios para cumplir con la solución del proyecto final. Dicha solución involucra el uso de mundos de Webots proporcionados por el profesor y el uso de código muestra de Webots y código compartido por el profesor. En caso de alguna duda, hay que ponerse en contacto con el profesor tutor y/o el profesor titular.

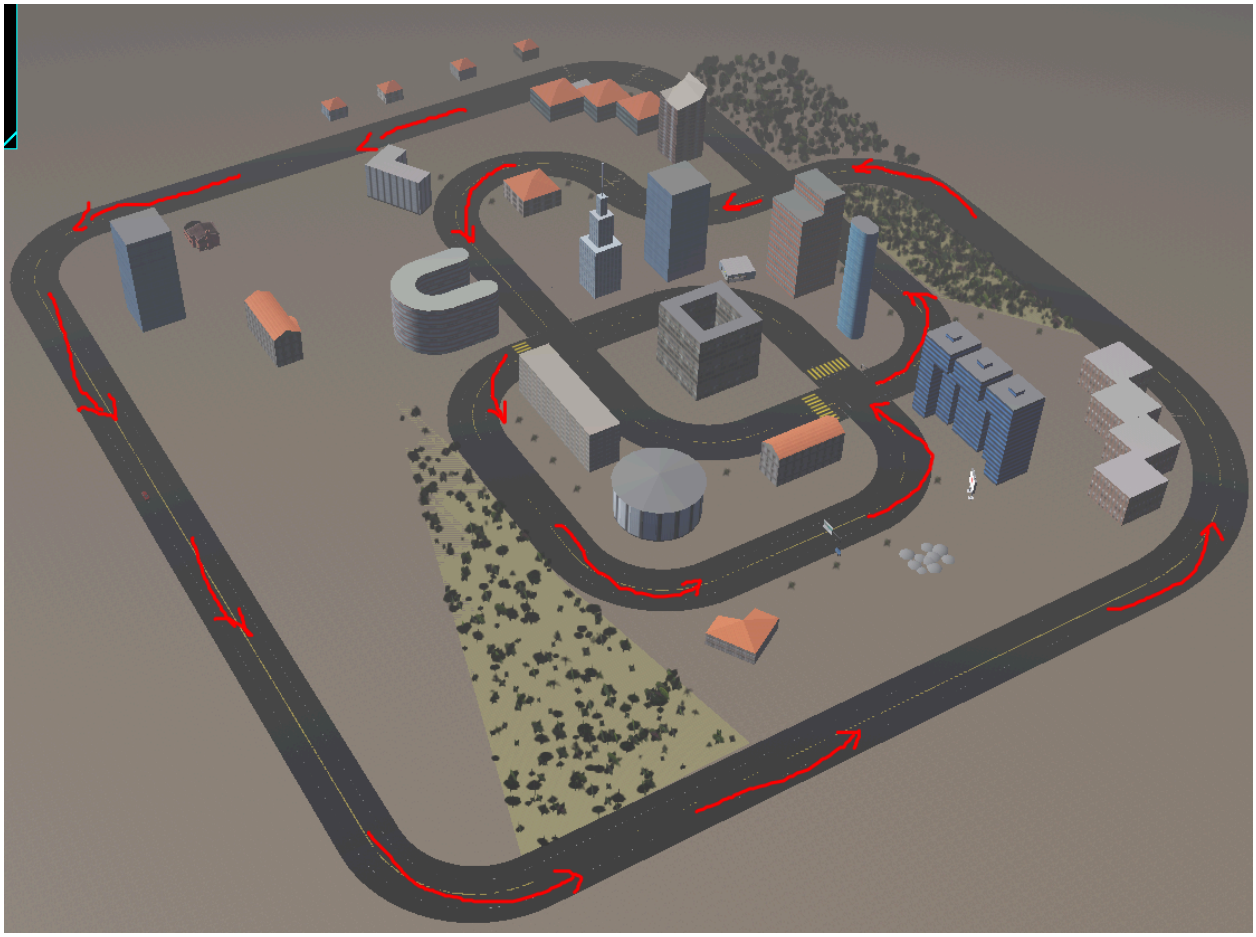
Behavioural Cloning

1. Se debe cargar el mundo ***city_traffic_2025_01.wbts***, el cual ya cuenta con un vehículo y su controlador indicado como externo. Si se desea, se puede agregar otro modelo de vehículo, eliminando el original, pero es necesario que este sea definido como DEF y que posea las mismas características que el vehículo original en el mundo, como muestra la imagen a continuación. Más información puede ser obtenida en la documentación de Webots: https://cyberbotics.com/doc/automobile/sumo-interface



2. El vehículo está esperando un controlador externo. Se recomienda hacer uso del mismo controlador desarrollado en la segunda actividad del curso, con las siguientes modificaciones:
    a. En lugar de que una tecla sea presionada para guardar una imagen capturada por la cámara, la captura de las imágenes deberán hacerse de manera automática y guardadas en una carpeta específica.
    b. Cada vez que una imagen sea guardada, es necesario que el nombre de la imagen junto con el ángulo de dirección sean registrados en un archivo CSV.
    c. En caso de que el periodo de captura de imágenes sea demasiado corto de tal manera que afecte la operación normal de la PC, se pueden introducir retardos o aumentar el ***time step*** del mundo.
3. El recorrido del vehículo en el mundo debe estar controlado por el teclado, manteniendo una velocidad constante y bien posicionado en el carril derecho (para no colisionar con

los vehículos generados por SUMO). El recorrido consiste en cubrir por completo la periferia del mundo, ir hacia el centro del mundo y regresar a la periferia, véase la imagen abajo. Se recomienda completar este recorrido en ambas direcciones y por al menos cinco repeticiones en cada sentido. El objetivo es tener un gran número de imágenes con las cuales entrenar una CNN. Es recomendable tener un dataset con al menos 10000 imágenes. Si el número de imágenes guardadas es considerado insuficiente, el número de repeticiones se puede incrementar o se puede hacer uso de data augmentation como se explica más adelante.



4. Se sugiere subir la carpeta de imágenes junto con el archivo CSV a una cuenta de GitHub y desde ahí importarlo a un Notebook en Google Colab, desde donde se va a entrenar una red neuronal convolucional (CNN) que va a intentar predecir un ángulo de conducción a partir de alguna imagen (i.e. problema de regresión). La CNN sugerida se puede obtener del artículo en formato PDF que se comparte con estas instrucciones.
5. Se sugiere hacer uso de ***data augmentation*** para mejorar la generalización del modelo. Esto es comentado en el artículo y se dieron algunos detalles durante una de las active classes.
6. Una vez que el modelo ha sido entrenado, este debe ser exportado y puesto a prueba sobre el mismo mundo a través de un controlador externo en Webots. Se espera que el recorrido pueda ser completado al menos una vez sin salirse del carril derecho.

7. Para más detalles de este método y/o más recomendaciones para el entrenamiento (data augmentation, manejo de recuperación, uso de GitHub, etc.), se sugiere consultar los capítulos de **Behavioural Cloning** en ambos libros de texto del curso (capítulo 10 del libro de Ranjan y capítulo 8 del libro de Venturi).
8. Opcional: Se puede generar un video de evidencia de que el método de Behavioural Cloning ha sido completado exitosamente. Se recomienda hacer uso de la herramienta de captura de video con Webots para tal propósito.

Generación de tráfico

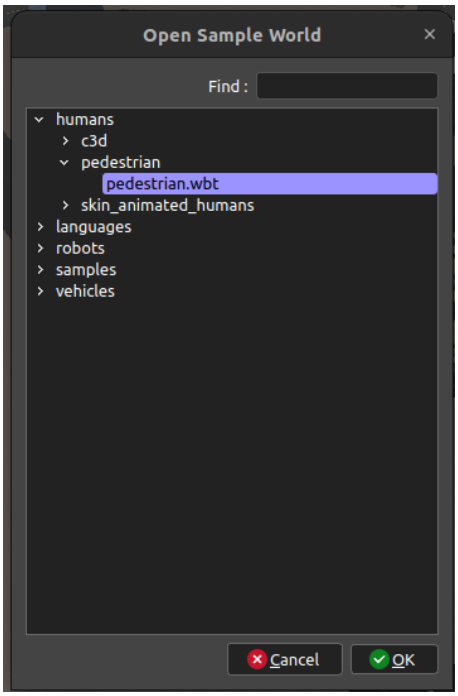
1. El modelo creado con el método de Behavioural Cloning debe ser sometido a prueba con tráfico y diez peatones. Este apartado cubre el aspecto de regeneración de tráfico usando SUMO, Webots y el mundo ***city_traffic_2025_02.wbt***, proporcionado como documento adjunto. Este mundo ya incluye una interfaz a SUMO (***SumoInterface***) con la generación de hasta 100 vehículos y un vehículo autónomo con controlador externo. El archivo del mundo deberá ser descargado junto con la carpeta del mismo nombre y con terminación ***net*** (***city_traffic_2025_02_net***). Esta carpeta es necesaria para que SUMO haga la generación de tráfico y controle los vehículos generados.
2. El vehículo autónomo incluido en el mundo puede ser sustituido por algún otro, pero este debe mantener el mismo DEF y el resto de las características que el original. Una vez incluido el modelo alterno, el vehículo original puede ser eliminado del mundo.
3. Se deben agregar los sensores necesarios al vehículo autónomo para lograr la detección de vehículos. Se pueden agregar cualesquiera de los sensores explicados en la active class y colocados en los slots del vehículo. La inicialización del sensor o sensores, su lectura y su toma de decisiones deberán ser codificados en el controlador externo.
4. Opcional: Se recomienda que el controlador incluya algún modelo ML o CNN desarrollado en alguna actividad o explicado en el curso para lograr la detección de vehículos con más certeza. El objetivo de este modelo es indicar si se ha detectado un vehículo en la imagen capturada por la cámara y, de esta manera, el sensor o sensores agregados al vehículo puedan indicar posición, velocidad, etc. También se puede hacer uso de un modelo CNN previamente entrenado y que este modelo sea importado directamente sin incurrir en el entrenamiento. Como documento adjunto a estas instrucciones, se ha incluido un ejemplo de cómo importar un modelo entrenado y usarlo para predecir.
5. El vehículo autónomo debe guardar cierta distancia (**distancia de umbral**) respecto al vehículo más próximo detectado modificando la velocidad. Si el vehículo detectado se encuentra a una distancia menor que la de umbral, el vehículo autónomo debería detenerse. El equipo deberá indicar en el video de evidencia el valor de esta distancia de umbral.
6. Debe tenerse en cuenta que, mientras más tiempo dure la simulación en Webots, un mayor número de vehículos controlados por SUMO serán agregados al mundo, con lo cual la simulación se hará más lenta y más recursos de la PC demandará Webots. Si la

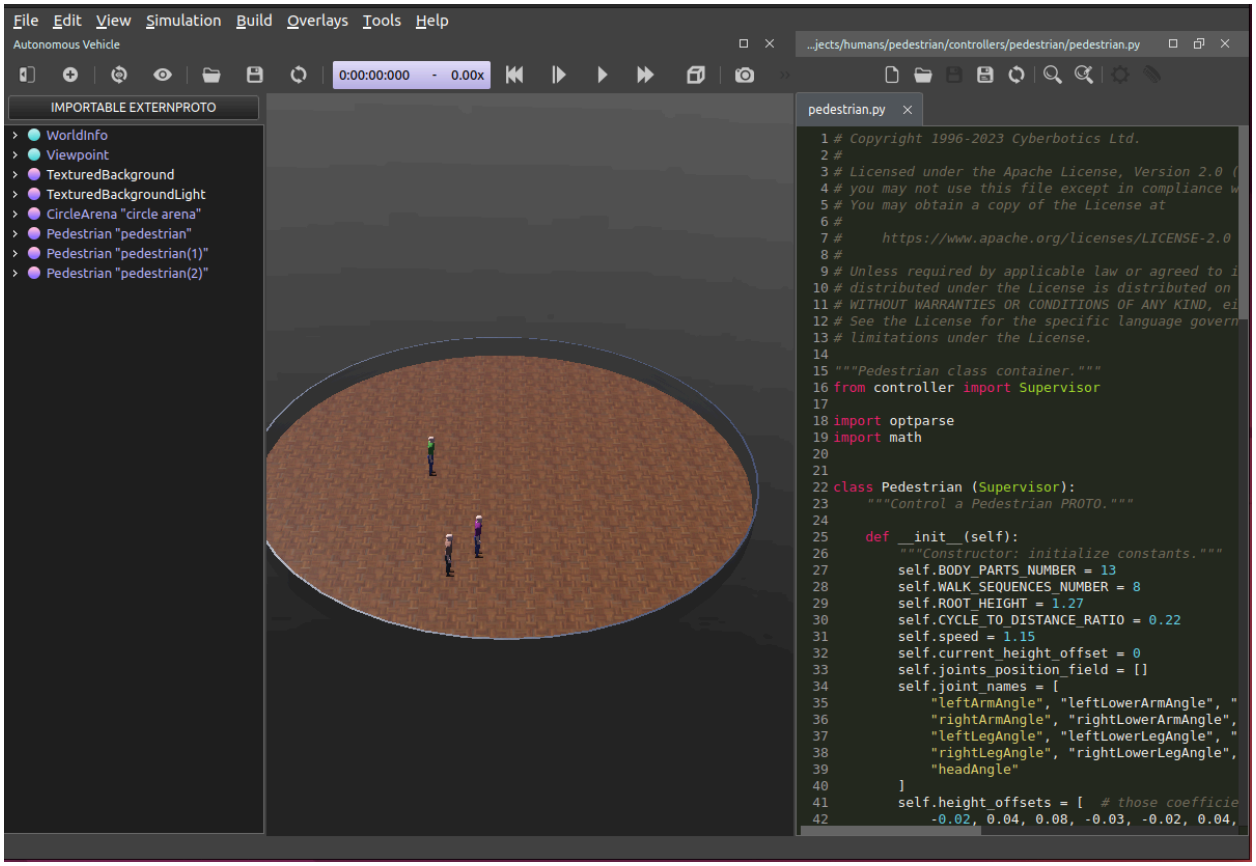
reducción de la ejecución de la simulación es muy drástica, se recomienda reducir el número de vehículos agregados por SUMO. Hay que asegurarse de que este número no sea menor a 30.

7. La forma de validar que el algoritmo desarrollado en este apartado funciona correctamente es mostrar en video que el vehículo autónomo detecta vehículos al frente (imprimiendo mensajes en consola, por ejemplo) y que este reduce la velocidad (mostrando el velocímetro en la simulación de Webots)

Peatones

1. Al mundo que se ha probado en el apartado anterior, se le deben agregar diez peatones, y se espera que, cuando el vehículo autónomo detecte alguno de ellos, se detenga por completo hasta que el peatón deje de obstruir la trayectoria del vehículo y/o no sea detectado por algún sensor.
2. Los diez peatones junto con su controlador deben ser importados del ejemplo muestra incluido en Webots y que fue demostrado en una active class. A manera de guía, se incluyen capturas de pantalla de la ubicación de este ejemplo muestra en Webots.





3. Si se explora este mundo, se podrá comprobar que los tres peatones son controlados por el mismo controlador ***pedestrian.py***. Al momento de incorporar estos diez peatones en el mundo de prueba comentado en el apartado anterior, se tienen dos opciones para la ejecución del controlador: mantener la ejecución dentro de Webots o indicar que el controlador es externo. Si el controlador se mantiene como de ejecución local, se debe pulsar el botón de “play” en Webots para poder ejecutarlo.
4. Se deben agregar los sensores necesarios al vehículo autónomo para lograr la detección de peatones. Se pueden agregar cualesquiera de los sensores explicados en la active class y colocados en los slots del vehículo. La inicialización del sensor o sensores, su lectura y su toma de decisiones deberán ser codificados en el controlador externo que controla al vehículo autónomo. Si se juzga adecuado, se pueden usar los mismos sensores usados para la detección de vehículos.
5. Opcional: Se recomienda que el controlador del vehículo autónomo incluya algún modelo ML o CNN desarrollado en alguna actividad o explicado en el curso para lograr la detección de peatones con más certeza. El objetivo de este modelo es indicar si se ha detectado un peatón en la imagen capturada por la cámara. También se puede hacer uso de un modelo CNN previamente entrenado y que este modelo sea importado directamente sin incurrir en el entrenamiento. Como documento adjunto a estas instrucciones, se ha incluido un ejemplo de cómo importar un modelo entrenado y usarlo para predecir.

6. La validación de la detección de peatones se mostrará junto con el resto de los requerimientos indicados en este documento, como se explica a continuación.

Validación del Vehículo Autónomo

1. La validación del vehículo autónomo deberá hacerse dentro de Webots usando el mundo **city_traffic_2025_02.wbt**, al cual se le deberá haber habilitado **SUMO** para la generación de tráfico y haber agregado diez peatones con un controlador incluido en la instalación de Webots.
2. El vehículo autónomo deberá incluir un número de sensores para la detección de vehículos y peatones. Estos sensores deberán haber sido agregados en los slots del vehículo.
3. En Webots, se deberán habilitar los diversos ***Optional Rendering*** para mostrar la presencia y operación de los sensores agregados, así como la visualización de la cámara y la medición de la velocidad del vehículo, en todo momento.
4. Opcional: La detección de peatones y vehículos puede ser mejorada incluyendo modelos entrenados desarrollados en el curso (SVM, DL o CNN) o importados desde Keras.
5. El vehículo autónomo deberá completar una vuelta alrededor de la periferia del mundo, entrando al centro del mundo y regresando a la periferia, mientras 1) detecta vehículos al frente y reduce su velocidad mientras mantiene una distancia prudente (distancia de umbral) y 2) detecta peatones y se detiene hasta que el peatón no sea detectado y/o no obstruya su trayectoria. Si la detección de peatones y/o vehículos no se logra en la primera vuelta, se pueden incluir una segunda o más vueltas en la simulación.