UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

APRENDIZAJE DE CONDUCTAS DE NAVEGACIÓN AUTÓNOMA EN SIMULACIÓN UTILIZANDO FRAMEWORK D-COACH

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO
CIVIL ELÉCTRICO

FELIPE ANDRÉS SALFATE NEIRA

PROFESOR GUÍA:
JAVIER RUIZ DEL SOLAR SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
SEBASTIÁN ISAO PARRA TSUNEKAWA
RODRIGO MORENO VIEYRA

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR AL
TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: FELIPE ANDRÉS SALFATE NEIRA
FECHA: 07/05/2020
PROF. GUÍA: DR. JAVIER RUIZ DEL SOLAR

APRENDIZAJE DE CONDUCTAS DE NAVEGACIÓN AUTÓNOMA EN SIMULACIÓN UTILIZANDO FRAMEWORK D-COACH

El Laboratorio de Robótica de la Universidad de Chile desarrolló un innovador *framework* de entrenamiento para redes neuronales deep, denominado D-COACH. Este *framework* consiste en un profesor humano que supervisa el funcionamiento de una red en tiempo real, quien corrige su funcionamiento por medio de señales positivas y negativas, sobre su espacio de acción. Hasta el momento, los trabajos que han empleado D-COACH demuestran ser prometedores, sin embargo, solo se ha probado con problemas relativamente simples hasta ahora.

Por otro lado, en la última década se han realizado avances en la conducción autónoma de vehículos. Sin embargo, existe una característica que no ha sido muy considerada, siendo esta la conducción *human-like*o *como lo haría un humano*. Esta característica es relevante cuando se realizan maniobras en presencia de conductores humanos, quienes esperan que el otro conductor se comporte de una manera determinada. Una incorrecta coordinación entre los conductores puede provocar un accidente de tránsito.

En este trabajo de memoria se utilizó D-COACH para entrenar agentes capaces de realizar la maniobra de adelantamiento de un vehículo, dentro del simulador para vehículos autónomos CARLA. También se buscó conseguir una conducción *human-like*, bajo la hipótesis de que esta característica es adquirible usando D-COACH. Para determinar el desempeño de estos agentes se realizaron 3 escenarios experimentales. El primer escenario considera una maniobra de adelantamiento sin elementos externos, mientras que en el segundo escenario se añaden más vehículos en el camino. El tercer escenario es una continuación del segundo, pero realizando un entrenamiento más acabado. En el primer y tercer escenario experimental, se consiguió entrenar agentes que logran realizar todas las maniobras de prueba exitosamente.

En los resultados obtenidos, se observó que el comportamiento de los agentes es similar al de los humanos, diferenciándose en que existe cierto zigzagueo en su movimiento (probablemente corregible) y que tienden a hacer la maniobra más cerca del vehículo que están adelantado. Pese a no ser *human-like*, esta conducta se puede considerar como una mejora. D-COACH presentó resultados satisfactorios para este trabajo de memoria, siendo más complejo que trabajos anteriores, sin embargo se hacen las siguientes observaciones:

- Con una mayor complejización de los escenarios, y en consecuencia un mayor número de eventos posibles, el tiempo de entrenamiento se extendió considerablemente. Esto es para evitar *overfitting* y conseguir la generalización de la red. Esta prolongación en el entrenamiento puede producir la fatiga del profesor, disminuyendo su efectividad.
- La diferencia de información entre el robot y el humano puede llevar a correcciones confusas para el robot. En este caso el robot solo podía observar el tiempo presente, mientras que un humano es capaz de retener eventos del pasado.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto Fondecyt 1161500.

Tabla de Contenido

Índice de tablas

Índice de figuras

1. Introducción

El Laboratorio de Robótica de la Universidad de Chile ha desarrollado un innovador *framework* de entrenamiento de redes neuronales. Este *framework* se llama Deep COACH [1] (D-COACH), una adaptación de *COrrective Advice Communicated by Humans* [2] (COACH) para redes profundas o *deep*, también desarrollado por el mismo laboratorio, basado en los paradigmas de modelación y asesoramiento de operadores (*Shaping paradigm* y *Advice Operators paradigm*).

A diferencia de otros métodos como Aprendizaje Supervisado (SL), Aprendizaje No Supervisado (NSL) o Aprendizaje Reforzado (RL), en donde se hace uso de una base de datos o de una función de recompensa, D-COACH emplea un supervisor humano para entrenar la red. Este supervisor monitorea el funcionamiento de la red en tiempo real, el cual entrega una corrección o *feedback* cuando considera que su comportamiento no es el adecuado. El *feedback* entregado puede ser negativo o positivo, y se aplica sobre el espacio de acción de la red. Este método de entrenamiento permite traspasar el conocimiento de un humano a una máquina, siendo esto particularmente útil en aquellas tareas que son consideradas triviales para una persona, pero que son todo un desafió para un robot. Esto se puede extender a tareas que una persona pueda entender como realizarlas, pero que por sus limitaciones físicas no es capaz de llevarlas a cabo (requiere un tiempo de reacción muy reducido, necesita mucha precisión o concentración, etc...).

D-COACH a demostrado ser mejor que otros métodos como RL, tanto en desempeño como en tiempo de convergencia, sin requerir de un profesional para su entrenamiento [1] [3]. Sin embargo, solo ha sido probado en casos relativamente simples, como lo son los problemas de *Cart-Pole, Car Racing* y *Pusher/Reasher*. Es por esto que este trabajo de memoria tiene como objetivo *probar D-COACH en un problema más desafiante, esto es entrenar un agente para realizar una maniobra de conducción vehicular*, en donde la dimensionalidad del estado es mayor y más variada. La maniobra escogida corresponde a la maniobra de adelantamiento, en donde el agente tendrá que controlar un vehículo y hacer que adelante a otro vehículo frente a él. El simulador utilizado para la realización de la maniobra corresponde a CARLA [4], un simulador de ambiente y vehículos autónomos dedicado al desarrollo de este tipo de vehículos.

Con respecto a la conducción autónoma, poco a poco los vehículos autónomos se han ido integrando a las calles, los cuales se espera que en un futuro consigan un nivel de autonomía total. Si esto se consigue, ayudaría a quienes no pueden o no deseen manejar, y ofrecería más tiempo libre a sus usuarios durante el intervalo de movilización. Sin embargo, para poder estar en circulación los vehículos autónomos deben cumplir con estrictas normas de seguridad, ya que un accidente de tránsito puede significar costos económicos, materiales, administrativos y sociales, que incluso puede llegar a comprometer vidas humanas [5].

Hoy en día existen vehículos autónomos que son capaces de seguir las reglas de tránsito y cuentan con mecanismos para evitar accidentes. Sin embargo, hasta el momento no se ha considerado una característica que podría disminuir el número de incidentes causados por vehículos autónomos. Esto corresponde a la conducción *human-like* o *"como un humano"*. En una maniobra, es importante que exista coordinación entre las partes involucradas, ya que debido al corto tiempo de reacción, si uno de los conductores actúa de forma inesperada, esto puede desencadenar en un percance. Con esto en consideración, un segundo objetivo es que *los agentes a implementar se comporten de manera human-like*. Se espera que usando D-COACH este objetivo sea cumplido, bajo el supuesto de que si un humano modela el comportamiento

de un agente, este tenderá a actuar como él.

1.1. Objetivos

De lo anterior se menciona el objetivo general y se enumeran los siguientes objetivos específicos.

1.1.1. Objetivos Generales

- Probar la capacidad de D-COACH para resolver un problema complejo. Para eso se utilizará en el entrenamiento de un agente, el cual deberá controlar un vehículo autónomo y hacer que adelante a otro vehículo frente a él. El vehículo también debe ser capaz de conducir dentro de la pista. Además se busca conseguir una conducta *human-like* por parte del agente, bajo la hipótesis de que esto puede ser logrado con D-COACH.

1.1.2. Objetivos Específicos

- Adaptar el código de D-COACH para ser utilizado con CARLA, un simulador de ambiente y vehículos autónomos.
- Implementar los módulos requeridos para realizar los experimentos.
- Determinar que y cuantos sensores serán utilizados por el agente, y su localización dentro del vehículo.
- Determinar la mejor arquitectura de la red a utilizar por el agente.
- Diseñar el sistema de aprendizaje.
- Conseguir que el agente sea capaz de conducir dentro de la pista por su propia cuenta.
- Entrenar y evaluar los agentes para realizar la maniobra de adelantamiento.
- Obtener mediciones de personas conduciendo para realizar la evaluación *human-likeliness*.

1.1.3. Estructura de la Memoria

En la sección 2 se presenta el marco teórico, entregando los conceptos necesarios para entender el trabajo realizado. En la sección 3 se da cuenta del trabajo realizado, dividiéndose en: sección 3.1, que describe de forma general el sistema diseñado para llevar a cabo los objetivos, sección 3.2, donde se detallan las características del sistema y su diseño, y sección 3.3, donde se describen los escenarios experimentales llevados a cabo. En la sección 4 Se presentan los resultados obtenidos de los escenarios experimentales y el análisis de estos. En la sección 5 se exponen las conclusiones obtenidas de la Memoria, junto al trabajo futuro propuesto a realizar.

2. Marco Teórico

2.1. Redes Neuronales

2.1.1. Neuronas

Una neurona es una unidad simple de procesamiento [6], compuesta por entradas, pesos sinápticos y una función de activación. El esquema de una neurona se puede ver en la figura 1. Cada vez que una neurona recibe una entrada U, esta pondera cada uno de sus elementos u_i por un coeficiente, denominado peso sináptico w_i. Se tiene un peso sináptico por cada elemento. Luego de la ponderación, cada uno de estos productos es sumado, generando un valor escalar. A este valor se le puede añadir un *bias*, el que puede ser considerado como un peso sináptico que pondera una entrada fija de valor 1. Lo mencionado anteriormente se puede expresar como:

$$v = \sum_{i \in inputs \cup bias} u_i \cdot w_i \tag{1}$$

Luego de ponderar y sumar las entradas, se introduce este valor en una función de activación $\varphi(\cdot)$, cuyo resultado corresponde a la salida de la neurona. Existe una gran variedad de funciones de activación, cuyo rango de salida suele ser 0 y +1, aunque también existen funciones con un rango de salida entre -1 y +1.

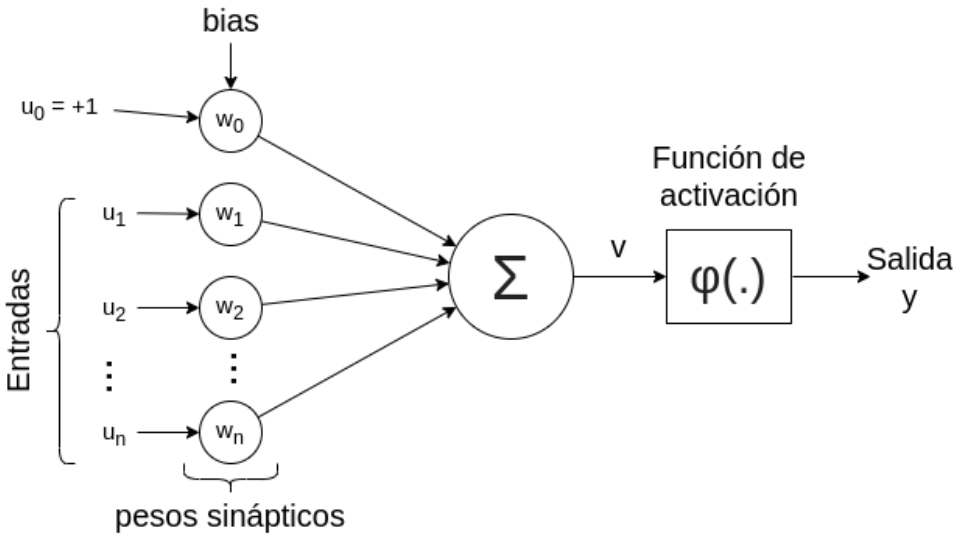


Figura 1: Representación de una neurona.

2.1.2. Red

Una red neuronal es una estructura compuesta por varias neuronas [6], cuyo funcionamiento se asemeja al de un cerebro. En una red las neuronas se encuentran ordenadas en conjuntos denominados capas, existiendo 3 tipos: capas de entrada, capas ocultas y capas de salida. En una red *Fully Connected*, cada

neurona dentro de una capa recibe la información de las neuronas de la capa anterior, esta información es procesada y entregada a las neuronas de la capa siguiente. En el caso de la capa de entrada, como su nombre lo indica, recibe la entrada de la red, mientras que la capa de salida entrega la respuesta final de esta. Una red *deep* es aquella que posee una o más capas ocultas.

2.1.3. Entrenamiento

Cuando se entrena una red neuronal, lo que se modifican son los pesos sinápticos de cada una de sus neuronas. Al modificar los pesos sinápticos, se modifica la importancia que le dan a cada una de sus entradas, resultando en un computo o procesamiento diferente para cada una de las neuronas.

El algoritmo de entrenamiento más utilizado es el de *Back-Propagation*, el cual utiliza el gradiente del error de la red para ajustar los pesos sinápticos. Esta corrección comienza en la capa de salida, en donde se usa dicho error, y luego continua en las capas anteriores, propagando el error dentro de la red, hasta llegar a la capa de entrada. El error generalmente se obtiene a partir de entradas cuya salida es conocida, por lo que se debe contar con un conjunto de pares entrada-salida o una función de costo para su entrenamiento.

2.2. Definición de un Problema

Dentro de un problema se desea que algo realice cierta acción. En el contexto de este trabajo de memoria, por ejemplo, se busca que un vehículo realice la maniobra de adelantamiento. Este elemento puede ser abstraído, reduciéndolo a un conjunto de variables y parámetros, los cuales pueden estar relacionados entre sí. Del mismo modo pueden existir elementos externos, con sus respectivas variables y parámetros, que pueden interactuar con el elemento principal. Siguiendo el mismo ejemplo, esto correspondería a otros vehículos y objetos presentes en la carretera, incluyendo el camino en sí mismo. El conjunto de todos esto elementos forman un sistema, en donde los elementos externos forman el entorno del elemento controlado. A las variables del sistema que son relevantes para el problema se les denomina estado, las cuales representan la evolución del sistema.

El control del elemento principal se realiza modificando sus variables. En la práctica, no todas estas variables son accesibles o modificables, por lo que a las variables que pueden ser modificadas y pueden alterar el comportamiento del elemento principal se les denomina variables de control. Por otra parte, las variables que se relacionan directamente con la conducta que se desea realizar se denominan variables objetivo, dando cuenta de si efectivamente se esta logrando el objetivo deseado. Entre las variables de control y las variables objetivos existe una serie de relaciones, que determinan como cambios en las primeras variables afectan a las segundas. En el caso del vehículo, las variables de control podrían ser sus pedales y dirección del volante, mientras que las variables objetivo podrían ser sus variables cinemáticas y posición relativa con respecto a los otros vehículos, que dan cuenta de forma directa si se está realizando la maniobra de adelantamiento.

Para llevar a cabo la tarea que se desea realizar, el elemento principal debe ser dirigido por un controlador. Este controlador entrega acciones, que corresponden a valores a fijar en las variables de control. De esto se desprende que el espacio de acción es un subconjunto del espacio de variables de control. Para poder generar dicha acción, el controlador podría necesitar una observación del estado, pudiendo ser

directa o indirecta (o ambas). Una medición directa podría ser la medición de la velocidad del vehículo por medio de un velocímetro, mientras que una medición indirecta podría ser la captura de una cámara, de la cual se pueden extraer variables como la posición relativa de los vehículos captados por la cámara. Por simplificaciones del lenguaje se le denominará a la observación del estado simplemente estado.

2.3. D-COACH

D-COACH [1] (*COrrective Advice Communicated by Humans for Deep networks*, o Asesoramiento COrrectivo Comunicado por Humanos para redes *Deep*) es un framework de entrenamiento para redes neuronales, que utiliza el *feedback* de un profesor humano para corregir su respuesta. D-COACH esta diseñado para que la persona que realiza el entrenamiento no sea necesariamente un usuario experto, asociado a la tarea que se esta intentando replicar. Este framework es una adaptación de COACH [2] para redes deep, que tiene el mismo funcionamiento, pero esta diseñado para entrenar modelos basados en políticas, como lo puede ser un modelo de funciones de aproximación.

El tipo de red a entrenar dependerá del número de variables que describan el estado. Para problemas con estados de baja dimensionalidad, se utilizan redes *feed-forward* o *fully-connected* (FNN) o redes recurrentes (RNN). Para problemas con estados de alta dimensionalidad, estas redes no son muy efectivas, debido a que el alto número de variables se traduce en un gran número de neuronas que deben ser entrenadas, siendo susceptible a *overfitting*. Esto significa que la red se termina sobre-ajustando al conjunto de entrenamiento, perdiendo su generalidad. En estos casos se utilizan redes convolucionales (CNN), cuyas entradas corresponde a imágenes, siendo procesadas por filtros. En estas redes los pesos sinápticos corresponde a los valores de los filtros, reduciéndose el número de variables a entrenar.

El *feedback* h entregado por el profesor humano es realizado en tiempo real, mientras la red esta en funcionamiento. Esta corrección es realizada en el dominio de la acción, en donde la persona indica la dirección en la que se debe corregir (de esto se desprende que h es un vector con la misma dimensionalidad que el espacio de acción). D-COACH asume que el usuario conoce el sentido de la rectificación, pero no su magnitud, por lo que h solo puede tomar los valores +1 y -1, siendo 0 cuando no se realiza una corrección sobre el eje respectivo. h es multiplicado por un valor e, denominado magnitud del error, generando el error de la red:

$$error = h \cdot e \tag{2}$$

Cada vez que el profesor realiza una corrección, este genera un par (s_t, $a_t + h_t \cdot e_t$), en donde s_t es la observación del estado en el tiempo t, $a_t = \pi_t(s_t)$ es la acción de la red en dicho estado en t y $h_t \cdot e_t$ corresponde al error de la respuesta en base al *feedback* entregado.

Como se mencionó, D-COACH asume que el usuario conoce la dirección de la corrección, pero no la magnitud. En el caso de COACH, del cual se basa D-COACH, la magnitud es determinada por medio de un modelo que caracteriza el *feedback* del humano, en donde su respuesta el multiplicada por el error. Este modelo es entrenado con el *feedback*, por lo que la magnitud del error dependerá de los valores entregados anteriormente a lo largo del tiempo. Sí para un determinado estado se reciben valores de $+1$ y -1 de forma alternada, la magnitud del error será cercana a 0, mientras que si para dicho estado

solo se reciben valores $+1$ o -1 de manera reiterada, este estará más próximo a $+1$ o -1, respectivamente.

En el caso de D-COACH, este estima la magnitud del error considerando correcciones pasadas, pero utilizando otro método, teniendo en cuenta que este trabaja con redes neuronales deep, en vez de RBF. D-COACH almacena los pares $(s_t, a_t + h_t \cdot e_t)$ en un *buffer* (algoritmo 1, línea 11), para luego entrenar la red con un *mini-batch* del *buffer*, correspondiendo a un subconjunto de elementos escogidos aleatoriamente. Nuevamente se están utilizando correcciones del pasado, para estimar la magnitud de la corrección. Como se observa en las lineas del 6 al 15 en el algoritmo 1, la red es entrenada en dos ocasiones: cuando se recibe un *feedback* y cada cierto número de iteraciones. En el primer caso, se entrena la red con el par generado en ese episodio y con un mini-batch, mientras que en el segundo caso, solo es entrenado con un mini-batch.

2.3.1. D-COACH con CNNs

Cuando el estado posee una dimensionalidad demasiado alta, se utiliza una red CNN, las cuales requieren una gran cantidad de información para ser entrenadas. Como en D-COACH esta información es generada por un humano, el proceso de su adquisición se convierte en una tarea lenta, lo que puede llevar a la fatiga del usuario, y en consecuencia reducir la efectividad el entrenamiento. Para resolver esto, se configura la parte convolucional de la red CNN como un AE (*autoencoder*).

Un AE busca reconstruir la información que recibe en su entrada, entregando dicha reconstrucción en su su salida. Para conseguir esto reduce la entrada a unas pocas variables, deshaciéndose de información redundante, a partir de las cuales comienza a recrear la entrada. El AE consta de dos partes, un codificador y un decodificador. El codificador esta compuesto por capas convolucionales, las cuales se encargan de reducir el número de variables de la entrada. El decodificar, por su parte, consta de capas convolucionales inversas, las que se encargan de la reconstrucción, aumentando el número de variables hasta llegar a la dimensionalidad original.

Al configurar la red CNN como un AE, se logra reducir el número de variables, utilizando el espacio latente de la red como entrada de la parte *fully connected*. El espacio latente corresponde a la capa ubicada entre el codificador y el decodificador, y corresponde a la capa con menor número de neuronas.

En D-COACH básico, la red es entrenada en una proceso de 3 etapas. En la primera etapa se obtiene un conjunto de muestras del estado. En la segunda etapa, el AE es entrenado con el conjunto de muestra. En la tercera etapa el AE se congela y solo se toma la parte del codificador, luego se procede a entrenar la parte *fully connected* con D-COACH, utilizando la salida del codificador como el nuevo estado. Los pasos 2 y 3 pueden verse en la figura 2. Uno de los problemas que presenta esta forma de entrenamiento es que no es robusto a cambios del entorno.

2.3.2. Enhanced D-COACH

Una alternativa al método presentado en la sección anterior es *Enhanced* D-COACH [3], el cuan entrena el AE y la red FNN de forma simultanea, eliminando la secuencialidad del entrenamiento. El AE es entrenado hasta que el error llegar a cierto umbral, el cual una vez alcanzado es congelado. Al limitar el entrenamiento del AE, se evita que el gradiente del agente se vea afectado cuando el error es mínimo.

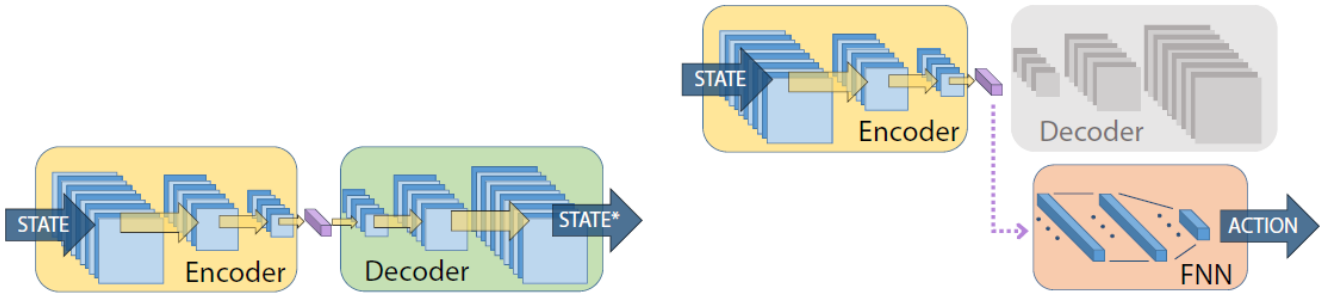


Figura 2: D-COACH en redes CNN. A la izquierda, el *autoencoder* en entrenamiento (segundo paso). A la derecha, uso del codificador (congelado) con FNN para reducir dimensionalidad del estado (tercer paso).

Enhanced D-COACH también ofrece una ventaja por sobre el método básico, ya que es capaz de adaptarse a su entorno si este cambia, debido a que cuando el error vuelve a superar el umbral, se descongela el AE y se vuelve a entrenar. Esto es útil cuando se pasa del simulador al mundo real. La representación de este sistema se puede ver en la figura 3.

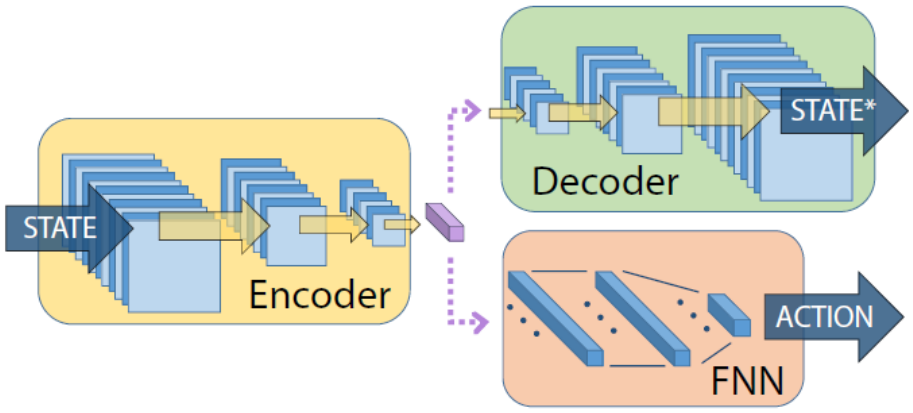


Figura 3: *Enhanced* D-COACH en redes CNN. El AE con la FNN son entrenadas simultáneamente.

En el algoritmo 1 se muestra el funcionamiento de *Enhanced* D-COACH, el cual como se mencionó es similar a D-COACH, con excepción de que se entrena el AE de forma simultanea con la red de acción. En las líneas 3 y 4 se muestra la observación del estado y la respuesta del agente ante esto. En la línea 5 se obtiene el *feedback* del humano y entre las líneas 6 al 13 se muestra el entrenamiento de la red cuando este *feedback* no es 0. Entre las líneas 14 y 20 se muestra el entrenamiento que se realiza cada b iteraciones, y la evaluación del AE para determinar si se sigue entrenando o no.

Algorithm 1: Enhanced D-COACH

Data: magnitud de error e, intervalo de actualización de buffer b, tamaño de muestreo de buffer N, tamaño de buffer K, parametros de encoder pre-entrenado (si aplica)

Init: $B \longleftarrow [\,]$ # Inicialización buffer
for $t = 1, 2, ...$ **do**
  Observar estado s_t
  Ejecutar acción $a_t \longleftarrow \pi(s_t)_t$
  Feedback humano h_t
  if h_t *no es* 0 **then**
    $error_t \longleftarrow h_t \cdot e$
    $y_(label(t)) \longleftarrow a_t + error_t$
    actualizar red usando con par (s_t, $y_(label(t))$)
    actualizar red usando con mini-batch de B
    agregar (s_t,$y_(label(t))$) a B
    if $length(B) > K$ **then**
      $B \longleftarrow B[2 : K + 1]$
  if *mod(t,b) es 0 y B no es* $\emptyset$ **then**
    Actualizar red con mini-batch de B
    if $AE_error > \epsilon$ **then**
      Actualizar AE con mini-batch de B
      Descongelar capas convolucionales
    else
      Congelar capas convolucionales

2.4. CARLA

CARLA [4] es un simulador creado para desarrollar, entrenar y validar sistemas de conducción autónoma. Este posee un código y protocolos *open-source*, además de *assets*[1] digitales diseñados cumplir el propósito de CARLA, que pueden ser usados de forma gratuita. Dentro de las características que posee CARLA están: mapas que simulan entornos urbanos, con sus respectivas señales de tránsito, vehículos controlables por medio de comandos o modificando directamente sus variables cinemáticas, sensores autónomos configurables, y una arquitectura para servidores multi-clientes. CARLA cuenta con una API para Python, con la cual se tiene un gran control sobre el simulador y los elementos presentes dentro de este.

El simulador está compuesto por 2 módulos: el simulador propiamente tal y la API de Python. El simulador se encarga de la lógica, físicas y *renderización* de los actores (sección 2.4.1), realizando el trabajo más pesado. La API corresponde a una librería de Python, con la cual se puede conectar e interactuar con la simulación. Por medio de esta API, se pueden generar y controlar los distintos elementos en el simulador, obtener información generada por estos elementos, y modificar diferentes configuraciones de

[1]En programación, un *asset* corresponden a recursos utilizados dentro de un proyecto que no corresponden a código o programación. Ejemplos de esto pueden ser imágenes, modelos 3-D, archivos de audio, etc.

la simulación, estando diseñada para ser una API con capacidades versátiles.

Dentro de las características que posee CARLA, esta el poder ser configurado para que cada paso del simulador tenga una duración o *time-step* fijo. Con esto, se logra que la simulación tenga un comportamiento más estable, aunque esto signifique un *delay* con respecto al tiempo real. Por tiempo real se entiende como el tiempo transcurrido en el mundo real, fuera de la simulación. Este *delay* ocurre cuando la iteración no se alcanza a computar en el *time-step* definido (tiempo real). En contraposición al modo *time-step* fijo, existe el modo de *time-step* variable, en donde la duración de cada iteración corresponde al tiempo de procesamiento requerido para computar dicha iteración. Con esto se logra que la simulación corra a la misma velocidad que el tiempo real.

Otra configuración que pose CARLA es el modo síncrono, en donde la simulación permanece en reposo hasta que el cliente envía una señal, controlando de forma manual la progresión de los pasos. En caso de no estar en modo síncrono, lo que corresponde al modo asíncrono, el simulador avanzará lo más rápido que pueda. Esto no es recomendado cuando se quiere que el tiempo del simulador sea el mismo que el tiempo real. Aunque el simulador este en modo síncrono, si no se alcanzan a computar los pasos a tiempo, este no correrá en tiempo real. Utilizando el modo síncrono junto al modo de *time-step* fijo se logra una simulación más estable y controlada.

2.4.1. Actores

Se denomina actor a una entidad dentro del simulador que es capaz de moverse y cumplir un rol. Un actor puede ser un vehículo, un peatón o un sensor, entre otras posibilidades. Estas entidades poseen un estado cinemático, compuesto por la posición con respecto al sistema de referencia global, rotación en sus 3 ejes, velocidad lineal, velocidad angular y aceleración lineal. A la posición y rotación con respecto a los ejes se le denomina pose. Esta información puede ser obtenida o modificada en todo momento.

2.4.1.1 Vehículos

En CARLA existen 27 modelos diferentes de vehículos, entre los que se encuentran motos y bicicletas. Cada uno cuenta con sus respectivas propiedades geométricas y dinámicas. Además de sus propiedades cinemáticas, un vehículo puede entregar información del tránsito, recibiendo estos datos directamente del simulador. Debido a esto, se les puede considerar como observaciones realizadas por sesores virtuales. Algunos de estos datos corresponden a la velocidad máxima de la última señal de tránsito observada, el estado del último semáforo visto y si se está frente a un semáforo.

Un vehículo puede ser controlado por medio de comandos contenidos dentro de un objeto de la API tipo *VehicleControl*[2]. Cuando el objeto que representa al vehículo dentro del programa recibe un *VehicleControl*, el simulador ejecuta los comandos contenidos dentro de este en la siguiente iteración. Cada comando posee un valor por defecto, el cual corresponde al comando a ejecutar en caso de que no se especifique su valor. Para variables numéricas el valor por defecto es 0, mientras que para variables booleanas es *false*. Los campos de este objeto son:

[2] `https://carla.readthedocs.io/en/latest/python_api/#carla.Vehicle`

- ***throttle*:** Indica la posición del pedal de aceleración, estando esta valor dentro del rango $[0, 1]$. La posición del pedal determina la cantidad de aire y combustible que llega al motor, controlando su potencia. En la señal *throttle*, 0 representa el pedal sin presionar, mientras que 1 representa el pedal presionado a fondo.

- ***steer*:** Representa la posición del manubrio, tomando valores dentro del rango $[-1, 1]$. La posición del manubrio determina la rotación de las ruedas delanteras, que se traduce en el ángulo de rotación del vehículo. El ángulo de rotación máximo del vehículo depende de cada modelo. En la señal *steer*, -1 representa el manubrio girado completamente hacia la izquierda, mientras que 1 representa el manubrio girado completamente a la derecha. 0 representa el manubrio en posición de reposo, correspondiente a un ángulo de rotación 0. Como se puede observar, este comando está normalizado, considerando todo el rango de rotación del vehículo.

- ***brake*:** Al igual que con *throttle*, representa la posición del freno, tomando valores dentro del rango $[0, 1]$, siendo 0 el pedal sin presionar y 1 el pedal presionado a fondo. La posición del pedal controla el sistema hidráulico que detiene las ruedas del vehículo, por medio de fricción. Cuando se aplican grandes fuerzas de desaceleración, las ruedas puede deslizar.

- ***hand brake*:** Valor booleano que activa o desactiva el freno de mano. El rango de este valor corresponde a $\{false, true\}$.

- ***reverse*:** Valor booleano que activa o desactiva el modo reversa. El rango de este valor corresponde a $\{false, true\}$.

- ***manual gear shift*:** Valor booleano activa o desactiva el modo de cambio automático. El rango de este valor corresponde a $\{false, true\}$.

- ***gear*:** Valor entero que indica el cambio del vehículo. Este cambio es único y no considera la reversa, siendo esto determinado por el comando *reverse*. El rango de este valor corresponde a $\{0, 1, 2, 3, 4\}$, siendo el valor del cambio -1.

2.4.1.2 Sensores

A continuación se presentan algunos de los tipos sensores[3] disponibles en CARLA:

- **Cámara RGB:** Corresponde a una cámara tradicional con los canales de color rojo, verde y azul. Dentro de los atributos que pueden ser ajustados están el tamaño de la imagen, la frecuencia de refresco y el FOV (*Field Of View* o campo de visión) horizontal. En la figura 4 se presenta una observación de este sensor.

- **Cámara de profundidad:** El valor de cada píxel corresponde a la distancia entre ese punto de la escena y la cámara. Los atributos que pueden ser ajustados están el tamaño de la imagen, la frecuencia de refresco y el FOV horizontal. Las imágenes generadas esta codificada en RGB, pudiéndose obtener la matriz de profundidad por medio de la siguiente ecuación:

$$distancia[m] = \frac{1000}{256^3 - 1}(R + 256G + 256^2 B) \tag{3}$$

[3] `https://carla.readthedocs.io/en/latest/ref_sensors/`

Figura 4: Imagen de una cámara RGB.

Se tiene que R representa el valor en el canal rojo, G representa el valor en el canal verde y B representa el valor en el canal azul. Estos valores están dentro del rango $[0, 255]$, por lo que la distancia máxima observada es de 1000 [m]. Este valor es fijo.

En la figura 5 se muestra una imagen de un sensor de profundidad, observándose la versión en RGB y como matriz de profundidad.

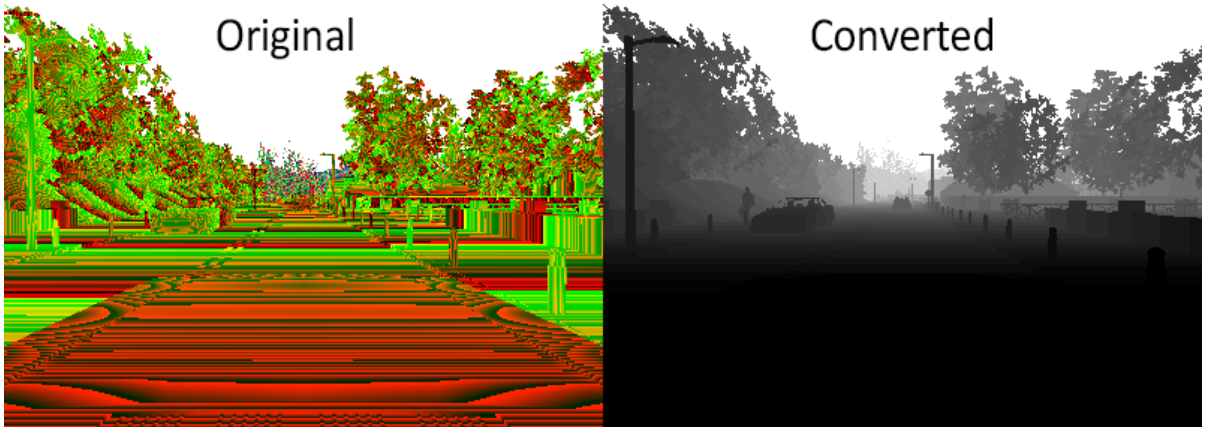


Figura 5: Captura de una cámara de profundidad. A la izquierda la imagen codificada en RGB, a la derecha transformada a valores de profundidad.

- **Cámara de segmentación semántica:** En las imágenes de este tipo, los valores de cada píxel corresponden al tipo de elemento al que pertenece. Estos elementos pueden ser construcciones, edificios o señaléticas, entre otros, siendo un total de 13 *labels*. Este sensor corresponde a una segmentación perfecta, obteniendo los *labels* desde el simulador. Los atributos que pueden ser ajustados son tamaño de la imagen, frecuencia de refresco y el FOV horizontal. Los *labels* van codificados

en el canal R.

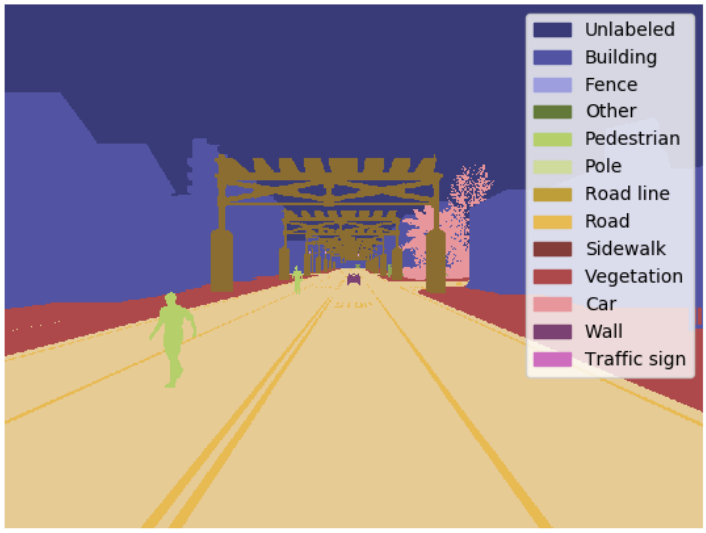


Figura 6: Captura de una cámara de segmentación semántica, con sus respectivos *labels*

En la tabla 1 se observan los *labels* disponibles en la segmentación semántica, junto con su codificación en el canal R. En la figura 6 se observa una imagen de segmentación semántica con sus respectivos *labels*.

- **LIDAR 3D:** Este sensor simula un LIDAR 3D utilizando *Ray-Casting*. *Ray-Casting* consiste en disparar múltiples rayos, o en este caso láseres, desde un mismo punto de origen en múltiples direcciones, y medir la distancia recorrida por cada uno de ellos, con lo cual se forma una imagen 3D del entorno. A cada una de estas mediciones se les llama puntos. Los puntos se calculan distribuyendo el número de canales definidos a lo largo del eje vertical, luego se calcula el ángulo de rotación por *frame* de acuerdo a la velocidad angular y se obtienen los puntos que se deberían generar en ese intervalo, de acuerdo al número de puntos por segundo.

  En este sensor se puede definir el número de canales, el rango del sensor, los puntos por segundo, velocidad rotatoria, el FOV superior e inferior, y los segundos entre captura. En la figura 7 se observa una representación 3D de las observaciones realizadas por un sensor.

- **Sensor de invasión de línea:** Este sensor detecta cuando el vehículo cruza alguna demarcación de una pista, equivalente a salir de la pista. Esta información es generada por un sensor virtual, obteniendo este dato directamente del simulador.

- **Otros:** Otros sensores disponibles son sensor de colisión, obstáculo (si un elemento se encuentra dentro de cierto rango) y GNSS. e invasión de línea.

Label	Valor en canal R
Unlabeled	0
Building	1
Fence	2
Other	3
Pedestrian	4
Pole	5
Road line	6
Road	7
Sidewalk	8
Vegetation	9
Car	10
Wall	11
Traffic Sign	12

Tabla 1: Clases presentes en la cámara de segmentación semántica con sus respectivos valores en el canal R.

2.4.2. Mapas

La versión utilizada de CARLA (0.9.5) cuenta con 7 mapas, los cuales representan una variedad de escenarios. Estos mapas pueden representar entornos rurales, urbanos o carreteras, con diferentes niveles de complejidad. Estos mapas cuentan con sus respectivas señales de tránsito y demarcaciones. Dependiendo del mapa se pueden encontrar diferentes tipos de pistas, número de carriles, intersecciones, elementos, etc.

Un mapa puede ser creado por medio de un programa llamado *RoadRunner* [7], el cual es un programa de pago. De acuerdo a la documentaciones disponible, un mapa puede ser modificado, sin hacer uso de *RoadRunner*, pero dicha sección no está disponible en la documentación de CARLA a la fecha.

2.5. Conducción *human-like*

Generalmente los vehículos autónomos se basan en métricas como una conducción eficiente en el tiempo o una conducción más segura, sin embargo, se han registrado múltiples choquen de conductores humanos con vehículos autónomos [8] [9] [10]. Esto se debe en parte a que aunque el vehículo esta realizando una conducción eficiente y segura, las personas pueden no reaccionar de buena manera esto. Al tipo de conducción realizada por un humano se le conoce como conducción *human-like* [11], el cual pese a que puede ser una conducción sub-óptima, puede resultar más segura cuando se enfrenta al factor humano.

Cuantificar que tan humana es la conducción de un agente no es una tarea fácil. Sin embargo, por medio de *Adversarial Machine Learning*, se han desarrollado herramientas para poder determinar si una red tiene un comportamiento *human-like*. Para esto se requiere utilizar una Red Generativa Antagonista [11] (GAN), entrenada para determinar si el comportamiento de un conductor corresponde a una máquina o un humano.

El trabajo a realizar se sustenta en la hipótesis de que D-COACH es capaz de generar una conducción

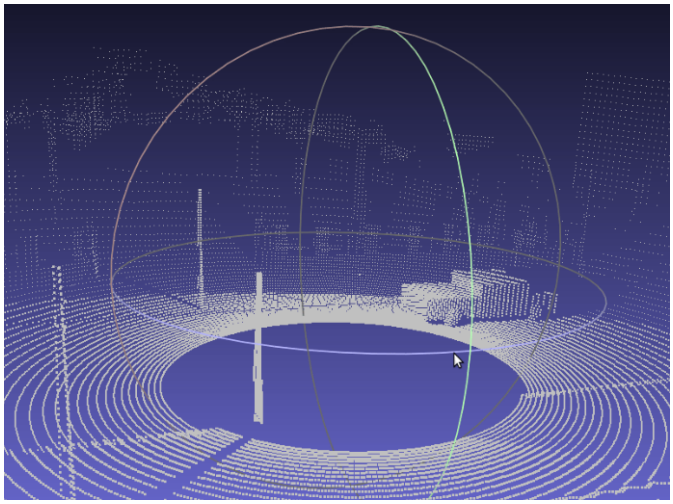


Figura 7: Representación 3D de observaciones de LIDAR.

human-like, con el supuesto de que si un humano corrige el actuar de un robot, este comenzará a adquirir características de la conducción humana.

3. Trabajo Desarrollado

Durante este trabajo se entrenaron distintos agentes mediante D-COACH, para controlar un vehículo dentro de una simulación. Este vehículo busca realizar la maniobra de adelantamiento de otro vehículo. Para el entrenamiento, se requiere de una persona que supervise a los agentes, cumpliendo el rol de profesor. La maniobra de adelantamiento se define como la acción de sobrepasar a otro vehículo o vehículos que se encuentren frente a uno. Esto debe ser realizado por la pista izquierda y asegurándose que ésta se encuentre despejada.

Para poder evaluar el desempeño de los agentes, se diseñaron y llevaron a cabo una serie de escenarios experimentales, los que se verán en la sección 3.3. Además, se les hizo realizar los mismos escenarios experimentales a un grupo de personas, con lo que se pudieron contrastar los resultados, y poder estimar cuan *human-like* fue el comportamiento de los agentes.

El sistema fue implementado utilizando *Python* 3.7.4, utilizado el simulador de vehículos autónomos CARLA 0.9.5. Para la implementación de las agentes se utilizó *Tensorflow* 2.0.0-beta0, mientras que para la interfaz se utilizó *Pygame* 1.9.6. Las herramientas mencionadas anteriormente se puede ver en la tabla 2.

Software	Versión	Uso
Python	3.7.4	Lenguaje de programación utilizado para implementación del sistema.
CARLA	0.9.5	Simulador de ambiente y vehículos autónomos utilizado para emular maniobra de adelantamiento.
Tesorflow	2.0.0-beta0	Librería utilizada para implementación de redes neuronales.
Pygame	1.9.6	Librería utilizada para implementación de interfaz del sistema.

Tabla 2: Software utilizado para la implementación del sistema.

3.1. Diseño del Sistema

El sistema implementado consta de 3 módulos: Entorno, Agente y GUI. Entorno corresponde a la simulación donde se llevan a cabo los experimentos, el mapa, donde se lleva a cabo la simulación, y todos los elementos presentes dentro de él, como lo son vehículos, sensores y señaléticas. A los elementos dentro del Entorno que cumplen algún rol se les denomina actores, usando la misma nomenclatura de CARLA. El Agente es el que se encarga de controlar al actor principal o *Hero* (siguiendo con la nomenclatura de CARLA), correspondiente al vehículo que debe realizar la maniobra de adelanto. D-COACH se encuentra dentro de Agente. La GUI corresponde al módulo por donde el profesor puede interactuar con

el sistema en tiempo real, por medio del cual puede observar y corregir el comportamiento del Agente, según se describe en D-COACH.

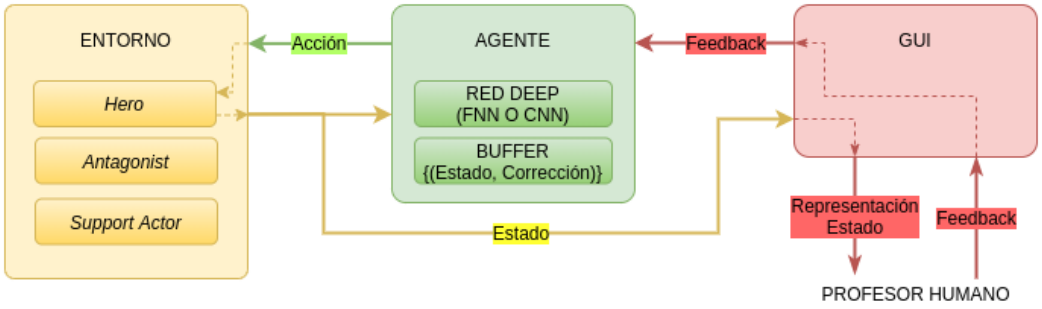


Figura 8: Diagrama de bloques del sistema.

En la figura 8 se observa un diagrama de los módulos y como se relacionan entre ellos, junto a los submódulos presentes en cada módulo. En Entorno se encuentran *Hero*, *Antagonist* y *Support Actor*, actores que cumplen un rol dentro de la maniobra de adelantamiento. En Agente se encuentra una red *deep*, necesario para generar la acción y un *buffer*, que almacena acciones y correcciones de tiempo pasados, necesarias para entrenar la red. En GUI se encuentra *feedback*, correspondiendo a la corrección realizada por el profesor humano. En cada iteración, Entorno entrega una observación del estado actual proveniente de *Hero*, el cual es transmitido a Agente y GUI. En base a este estado, Agente genera la acción que debe realizar *Hero*, y es enviada de vuelta a Entorno. Por su parte, cuando GUI recibe el estado, este presenta al profesor las imágenes RGB e información adicional contenida en el estado, con lo que el humano puede evaluar el comportamiento de Agente y entrega un *feedback*, sí este lo estima necesario. En caso de que se realice una corrección, Agente recibe este *feedback* y entrena su red.

El sistema esta diseñado bajo una lógica de episodios e iteraciones. Al inicio de cada episodio, *Hero* aparece detrás de *Antagonist*, el cual debe intentar adelantarlo. El episodio finaliza una vez *Hero* sobrepasa a *Antagonist* y vuelve a su pista original, o en caso de que no se pueda completar la acción. En un principio, solo se consideran a *Hero* y *Antagonist* como los únicos actores presentes dentro del episodio, sin embargo, también se puede considerar el caso en el que un tercer vehículo se encuentra presente al momento de realizar la maniobra, pudiendo afectar como esta se desarrolla. A este tercer vehículo o actor se le denomina *Support Actor*.

Cuando se realiza la evaluación de Agente y no recibe ningún tipo de corrección, los módulos de *feedback* y *buffer* son desactivados. Esto se traduce en que la red no recibe ningún tipo de entrenamiento durante la evaluación. Cuando se llevan a cabo la medición de una persona conduciendo, se desactiva el módulo de Agente, recibiendo la acción a realizar por *Hero* desde GUI. En la figura 9 se muestra un diagrama con ambas configuraciones. En la imagen superior se muestra el sistema cuando se está evaluando Agente, mientras que en la imagen inferior se muestra el sistema cuando es controlado directamente por un humano.

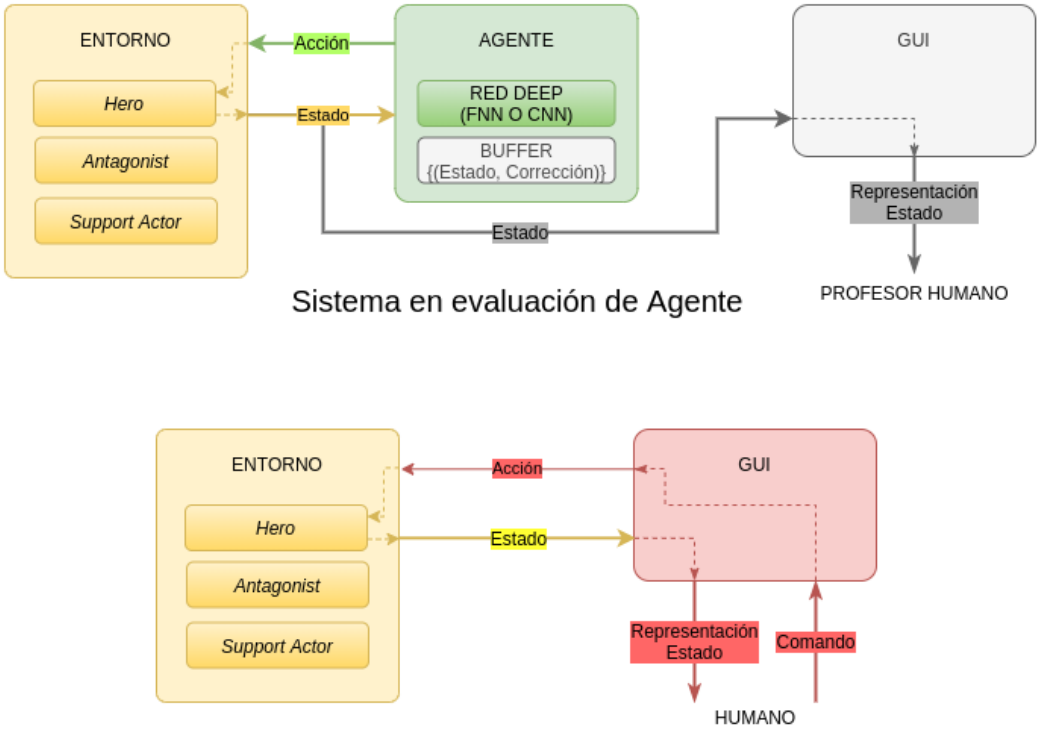


Figura 9: Diagrama de bloques del sistema cuando se está evaluando Agente (imagen superior) y cuando *Hero* es controlado por un humano (imagen inferior). En evaluación de Agente es opcional si se desea mostrar el estado al profesor humano.

3.1.1. Entorno

Dentro del Entorno es donde ocurre la simulación, conteniendo el mapa y los distintos elementos presentes en este. Esto incluya las pistas, señales de transito, elementos del paisaje y vehículos. Utilizando la nomenclatura de CARLA, a los vehículos que cumplen un rol dentro de la maniobra de adelanto se les denomina actores. Estos actores son:

- ***Hero*:** Vehículo que debe realizar la maniobra de adelantamiento. Puede ser controlado por Agente o por un humano, pero no puede haber más de un *Hero*. Este actor cuenta con una serie de sensores, por medio de los cuales percibe el entorno. Estas observaciones le sirven tanto al Agente, como a la persona que se encuentre utilizando el sistema, ya sea que este actuando como profesor o como conductor.

- ***Antagonist*:** Vehículo a adelantar. Al igual que *Hero* corresponde a un actor único, considerando que solo se busca realizar el adelantamiento de un solo vehículo. En principio, solo se requiere que viaje dentro de la pista, pero también esta la posibilidad de complejizar su comportamiento. Por ejemplo, su velocidad podría variar de forma aleatoria.

- ***Support Actor*:** Cualquier otro vehículo que participe en la maniobra, que no corresponda a *Hero* o *Antagonist*. Debido a esto, puede haber más de un *Support Actor*. Ese actor puede ser simplemente

un vehículo que circule por la pista, bloqueando la realización de la maniobra por un período de tiempo. Sin embargo, también puede ser un elemento más activo, pudiendo ser un vehículo que busque realizar la maniobra de adelantamiento al mismo tiempo que *Hero*, o ser el causante de un accidente de tránsito.

3.1.1.1 *Hero*

Además del vehículo, también se asocian a *Hero* los sensores equipados en él. A partir de las observaciones realizadas por los sensores, más información adicional del vehículo, se genera una observación del estado, lo cual es utilizado por Agente y GUI.

Para moverse, *Hero* debe recibir una acción, correspondiente a una serie de comandos a realizar. Esta acción es enviada por Agente, o por GUI, en caso de que una persona lo este controlando. Se considera que *Hero* corresponde a un vehículo automático, además de que no se contempla el uso del freno de mano o ir en reversa. Tomando esto en cuenta, los comandos que recibe *Hero* corresponden a 3, *throttle*, *brake* y *steer*, siendo los pedales del acelerador, frenos y volante, respectivamente.

3.1.2. Agente

Cuando *Hero* no es manejado por un humano, este es controlado por el módulo Agente, por lo que Agente debe ser capaz de hacer que *Hero* realice la maniobra de adelantamiento. En cada iteración recibe una observación del estado del vehículo, con lo cual genera una acción, siendo esta las señales de *throttle*, *steer* y *brake*, la que es entregada de vuelta a *Hero*. También, Agente recibe de GUI el *feedback* realizado por el profesor, el cual es una corrección positiva o negativa. Con este *feedback*, el Agente crea la corrección del sistema, ponderando y sumándolo a la última acción realizada.

Como se está utilizando D-COACH, el cual está diseñado para entrenar redes *deep*, Agente debe poseer una red de este tipo para generar su respuesta. Del mismo modo, la red utiliza el estado entregado por *Hero* como entrada. Debido a esto, el tipo de red a utilizar dependerá de la dimensionalidad del estado, utilizando una red *fully connected* para un estado de baja dimensionalidad, o una red CNN, en caso de ser demasiadas variables.

La red podría recibir como entrada directamente el estado generado por *Hero*, sin embargo esto podría no ser lo más eficiente. El estado podría ser una variable de alta dimensionalidad, pero con una gran cantidad de información redundante. Del mismo modo, la representación con la cual se recibe el estado, dependiendo directamente del tipo de sensores utilizado, podría no mostrar los cambios del entorno de forma conveniente, o no ser muy compatible con redes neuronales. Es por esto que en vez de utilizar el estado directamente, se puede realizar un pre-procesamiento, por el cual se puede reducir el número de variables y hacerlo más manejable por la red. Esto incluso puede cambiar el tipo de red a utilizar.

Al igual que con el estado, la salida de la red no necesariamente tiene que ser la respuesta de Agente. La red podría entregar una representación distinta de las señales de *throttle*, *brake* y *steer*, o entregar otro tipo de variables, de las cuales se determinen estas señales. La respuesta final se obtendría por medio de un post-procesamiento, correspondiendo a un tipo de función o transformación. Este post-procesamiento, además, puede servir para darle una característica a la respuesta, sin necesitar entrenar la red para esto.

Esto puede ser un post-procesamiento que suavice la salida, o le de un comportamiento no lineal.

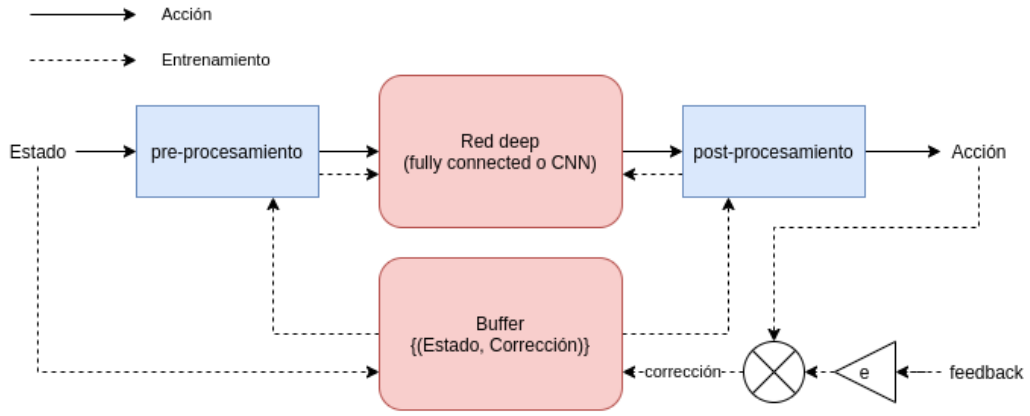


Figura 10: Diagrama del funcionamiento del Agente. Las líneas continuas representan la generación de la acción, mientras que las lineas punteadas representan el proceso de entrenamiento.

En D-COACH, cada vez que el profesor realiza una corrección, o cada cierto número de iteraciones, la red es entrenada con pares estado-corrección. Esto se realiza con la última corrección realizada por el profesor, pero también con correcciones realizadas en el pasado. Para poder almacenar estos pares, se requiere de un *buffer*, el cual retenga las últimas K correcciones realizadas, junto con los estados asociados a estas correcciones. Se debe considerar que antes de entrenar la red con esta información, se debe realizar el pre-procesamiento sobre los estados, y el post-procesamiento inverso sobre las correcciones, o realizar este proceso antes de almacenarlos en el *buffer*. Para ser más eficiente, también se puede almacenar el estado pre-procesado, junto con la salida directa de la red. Se hace notar que como la corrección se realiza sobre el espacio de acción, distinto al espacio de salida de la red, se requiere que el post-procesamientos tenga una transformación inversa única.

Tanto el entrenamiento de la red deep, como el pre y post procesamiento realizado sobre la entrada y salida de la red se puede ver en la figura 10.

3.1.3. GUI

Por medio de GUI, el profesor puede interactuar con el sistema. Este debe ser capaz de observar el funcionamiento de Agente en tiempo real, al igual que debe poder entregar *feedback*, cuando Agente no se comporta de la forma que estime conveniente. Para poder ver el funcionamiento de Agente, GUI muestra en pantalla lo que ven las cámaras de *Hero*, junto con información adicional del estado. Por su parte, el profesor entrega *feedback* por medio del teclado (o cualquier otro método o dispositivo que se utilice), con lo cual se obtiene una corrección, que es utilizado para moldear la respuesta del Agente.

El *feedback* corresponde a un valor positivo o negativo, en el espacio de la acción, siendo este las señales de *throttle*, *steer* y *brake*. Cuando el profesor considera que una de estas señales debe ser menor o mayor, entrega -1 o +1, respectivamente. Si estima que no se deben hacer correcciones, entrega 0. Se

puede entregar *feedback* en cada una de los ejes de forma simultanea, con lo que el *feedback* termina siendo un arreglo de 3 valores (-1, +1 o 0), cada uno asociado a una señal. Teniendo esto en consideración, el número total de *feedbacks* posibles es igual a $3^3 = 27$. Debido a esto, los *feedbacks* de los que dispone el profesor son un subconjunto de todos los *feedbacks* posibles, debiéndose escoger aquellos que sean más convenientes. Finalmente, a cada uno de estos *feedbacks* escogidos se le asocia un botón del teclado.

La información de la que debe disponer el profesor debe ser similar a la que tiene Agente, de lo contrario, la corrección realizada por el profesor podría ser malinterpretada. Sin embargo, esta no tiene que ser la misma, ya que datos que podrían ser útiles para Agente, podrían no serlo para el profesor. Del mismo modo, el profesor podría contar con información adicional, de la cual Agente no disponga, que le ayude a prepararse de mejor manera a lo que viene. No obstante, esto no debe generar discrepancia con lo percibido por Agente, al momento de entregar un *feedback*. Considerando que la vista es una parte significativa en la conducción, es importante mostrar lo que ven las cámaras del auto. Idealmente deben ser imágenes RGB, considerando que esto es a lo que está acostumbrado a ver una persona. Si se dispone de este tipo de imágenes, puede no ser necesario tener las imágenes de otro tipo de sensores, como por ejemplo segmentación semántica.

3.2. Detalles de Implementación del Sistema

En esta sección, se entrara más en detalle con la implementación final, y el trabajo realizado con respecto a esto.

Con respecto al sistema, este está configurado con un *time-step* de 0.05 segundos, equivalente a 20 [Hz], resultando en 20 iteraciones por segundo. En general, se intenta conseguir que el tiempo del simulador sea el mismo que el tiempo real, corriendo a la misma velocidad. Esto se debe a que una persona no está acostumbrada a responder en cámara lenta, pudiendo traducirse en el entrenamiento de una conducta que no es *human-like*. También se debe considerar que *frame rate* (número de cuadros por segundo) observado para el humano debe ser adecuado, debido a que un bajo *frame rate* puede ser molesto, pudiendo afectar la efectividad del entrenamiento. Esto puede ser especialmente incomodo si el entrenamiento se extiende por un prolongado periodo de tiempo. En este caso se considera el *frame rate* y el *time-step* iguales, siendo de 20Hz.

3.2.1. Entorno

3.2.1.1 Sensores de *Hero*

Los sensores que se implementaron para *Hero* fueron cámaras RGB, profundidad y segmentación semántica. Además de las observaciones realizadas por estos sensores, el estado entregado por *Hero* también se compone de la información cinemática del vehículo, correspondiente a su velocidad lineal, velocidad máxima, velocidad angular y aceleración lineal. En la tabla 3 se muestra un resumen de las variables que componen el estado. En la simulación, las variables cinemáticas pueden tomar cualquier valor positivo, sin embargo se consideró como rango máximo el valor para el cual son normalizados.

Variable	Tipo de Dato	Unidad de medida	Rango de valores
Imagen RGB	Matriz entera de 3 canales	–	$\{0, \ldots, 255\}^{mxnx3}$
Imagen de profundidad	Matriz entera de 3 canales	–	$\{0, \ldots, 255\}^{mxnx3}$
Imagen de seg. sem.	Matriz entera	–	$\{0, \ldots, 255\}^{mxn}$
Velocidad lineal	*float*	m/s	$[0, 35]$
Velocidad máxima	*float*	km/h	$[0, 120]$
Velocidad angular	*float*	m/s	$[0, 30]$
Aceleración lineal	*float*	m/s²	$[0, 10]$

Tabla 3: Variables sin procesar que componen el estado.

Se tiene que el sensor de segmentación semántica de CARLA corresponde a una segmentación perfecta. Para añadir más realismo a este sensor, se le añade un filtro medio de kernel 7x7, con una probabilidad de 10 %.

3.2.1.2 Representación del Estado

En la sección anterior se vieron los sensores utilizados por *Hero* en su formato original. Sin embargo, estas representaciones pueden no ser las óptimas para Agente, en particular las imágenes captadas por las cámaras. Es por esto que se consideraron diferentes representaciones para las imágenes. En el caso de las variables cinemáticas, al ser escalares, estas simplemente fueron normalizadas.

Para las imágenes en RGB, se evaluaron los formatos de RGB, HSV o escala de grises. HSV corresponde a *hue*, *saturation* y *value*, siendo el matiz, la saturación y valor del color respectivamente. Para las imágenes de profundidad, se considero la imagen sin decodificar (RGB), como matriz de profundidad o como matriz de profundidad en escala logarítmica. Para las imágenes en segmentación semántica, se consideró como matriz de valores o en *one-hot-key*.

Para el re-escalamiento de segmentación semántica se implementaron 3 métodos, denominados *Max*, *Most* y *Mean*. Sea $B(x, y)$ una matriz binaria, en donde $B(x, y) = 1$ significa que un objeto de una determinada clase existe en (x, y) y 0 si no existe. Al reducir la resolución en una razón m para el ancho y n para el alto, la imagen se divide en $m \times n$ regiones, en donde cada región representará un píxel en la nueva imagen. Para una región $B_{i,j}(x, y)$, siendo $B'(i, j)$ la matriz re-escalada, el método *Max* se define como:

$$B'(i,j) = Max\{(B_{i,j}(x,y))\} = \begin{cases} 1, \text{if } \exists x_0, y_0,\ B_{i,j}(x_0, y_0) = 1 \\ 0, \text{otherwise} \end{cases} \tag{4}$$

Se hace notar que es equivalente a tomar el valor máximo de la matriz. Considerando h y w como el alto y ancho de la región $B_{i,j}(x, y)$, el método *Most* se define como:

$$B'(i,j) = Most\{(B_{i,j}(x,y))\} = \begin{cases} 1, \text{if } \sum_{x,y} B_{i,j}(x,y) \geq \frac{h \cdot w}{2} \\ 0, \text{otherwise} \end{cases} \tag{5}$$

Finalmente, el método *Mean* se define como:

$$B'(i,j) = \frac{\sum_{x,y} B_{i,j}(x,y)}{w \cdot h} \tag{6}$$

En este caso se tiene que la imagen de segmentación semántica deja de ser una matriz binaria. Sin embargo, los elementos de la matriz siguen teniendo valores entre 0 y 1, por lo que puede seguir siendo utilizado por una red. Cabe mencionar que estos métodos son utilizados cuando la imagen se encuentra en su representación *one-hot-key*.

En la figura 11 se observan los diferentes métodos de re-escalamiento de segmentación semántica implementados sobre una imagen. Se observa que con los métodos *Most* y *Max* se pierden detalles de la imagen, mientras que el método *Mean* preserva de mejor manera las formas y contornos.

3.2.1.3 Configuración de las Cámaras

En un comienzo, solo se tenía una cámara frontal, sin embargo, se observó que para realizar la maniobra de adelantamiento, se requería saber cuando se había pasado al vehículo por completo. Debido esto,

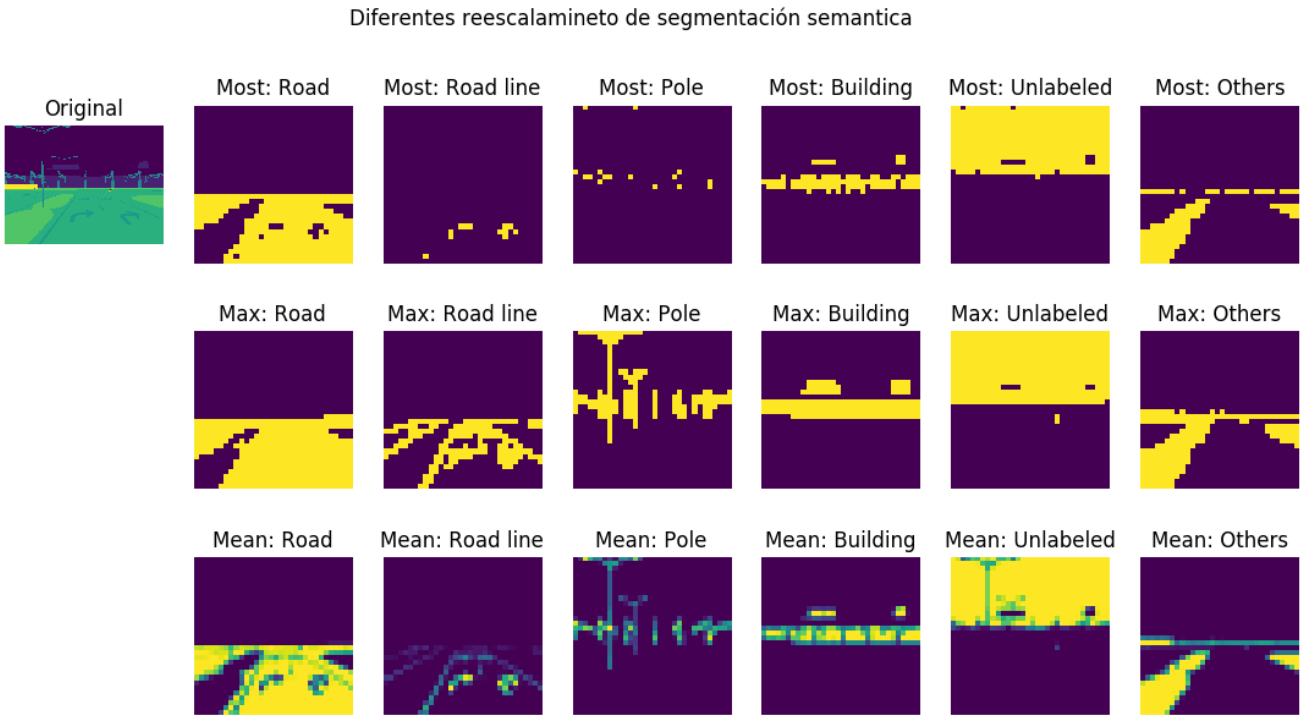


Figura 11: Diferentes métodos de re-escalamiento de segmentación semántica. Cada columna representa un canal, mientras que cada fila representa un método. Arriba a la izquierda se observa la imagen original, como una matriz de etiquetas.

se decidió agregar una cámara trasera, que imita la función del espejo retrovisor central. No obstante, al utilizar esta configuración, se notó que mientras se adelantaba al vehículo existía un punto ciego. Esto producía que por un intervalo de tiempo, Agente no podía distinguir si estaba realizando la maniobra de adelantamiento, o si se encontraba solo en la pista, generando problemas. Para solucionar esto, se definió una segunda configuración, en la que se dispone de dos cámaras laterales apuntando hacia atrás, las cuales imitan la función de los espejos retrovisores laterales. Estas cámaras están dispuestas de tal manera que permiten ver a los vehículos que se encuentran detrás. Esta configuración posibilita monitores en todo momento al vehículo adelantado, aunque la visión hacia atrás no es tan clara. Es por esto que se usaron ambas configuraciones, las cuales se pueden ver en la figura 3.2.1.3, además de la configuración inicial con una sola cámara.

La resolución utilizada para la cámara frontal es de 800x600, considerando que la conducción se realiza principalmente en base a lo que observa esta cámara. Con respecto a las cámaras trasera y laterales, como su única función es identificar vehículos que se hallan adelantado o que se encuentren detrás, se usó una resolución menor. Por esta misma razón, se optó por una imagen más alargada hacia los lados. Para la cámara trasera se escogió una resolución de 200x75, mientras que para las cámaras laterales se escogió 100x75. En el caso de las cámaras laterales, estas son entregadas como una sola imagen, siendo unidas una junto a la otra, formando una imagen de 200x75 (misma resolución que la cámara trasera). Esto se hizo para no tener que cambiar de arquitectura cada vez que se utiliza una configuración diferente.

Cabe mencionar que cada cámara esta compuesta un sensor RGB, un sensor de profundidad y un sen-

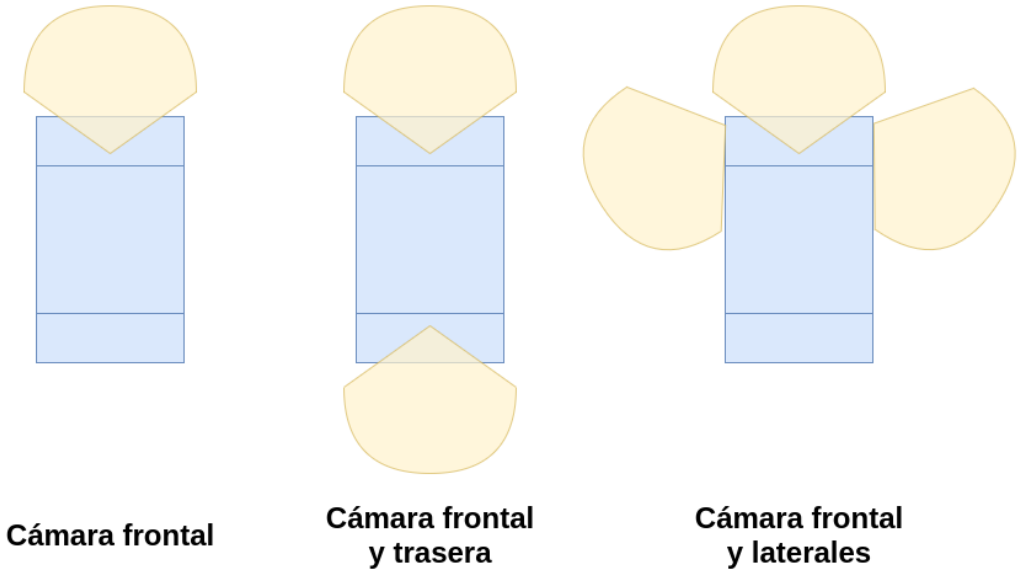


Figura 12: Configuración cámaras en *Hero*. A la izquierda, la configuración inicial, con una cámara frontal. Al centro, la configuración con una cámara frontal una trasera. A la derecha, la configuración con una cámara frontal y dos cámaras laterales.

sor de segmentación semántica.

3.2.1.4 Agente PID

Para que *Hero* pueda adelantar a *Antagonist*, en principio solo se requiere que *Antagonist* avance en línea recta, doblando cuando solo sea necesario. Para poder llevar a cabo esto, se utilizó una versión modificada de *Roaming Agent*[4], presente en la API de CARLA, para controlar a *Antagonist*. Este hace que el vehículo se mueva libremente por el mapa, a una velocidad determinada, deteniéndose cuando hay un vehículo en frente. Para poder generar la respuesta, el agente utiliza dos controladores PID, uno para determinar la acción longitudinal (freno y acelerador) y otro para determinar la acción lateral (volante). La información utilizada por el agente corresponde a datos obtenidos directamente del simulador, siendo estos información de la pista y variables cinemáticas de los otros vehículos. El agente fue modificado para que busque ir por la pista más paralelo a la pista actual (en otras palabras, ir en línea recta), además de mejorar la detección de vehículos de frente y la respectiva detención cuando esto ocurre. Este agente también es utilizado para controlar a *Support Actor* en algunos casos.

3.2.2. Agente

Como se mencionó, Agente debe ser capaz de generar una respuesta, correspondiente a 3 señales: *throttle*, *steer* y *brake*. Esto se debe realizar en base al estado, estando compuesto por observaciones realizadas por *Hero*, junto con información cinemática del vehículo. Lo anterior se puede ver en la siguiente

[4] `https://github.com/carla-simulator/carla/blob/master/PythonAPI/carla/agents/navigation/roaming_agent.py`

ecuación:

$$(\textit{throttle}, \textit{steer}, \textit{brake}) = \Pi(U, K) \tag{7}$$

En donde Π representa a Agente, U representa las observaciones de *Hero* y K el estado cinemático.

Como se esta utilizando D-COACH para entrenar su comportamiento, el cual funciona solo con redes *deep*, esta respuesta se debe obtener a partir de este tipo de redes. Un esquema general del funcionamiento del Agente se presentó en la figura 10. A continuación se muestra el diseño implementado para Agente y como se llegó a esto.

3.2.2.1 Modelos implementados

Dentro de los sensores utilizados, se tiene que por un lado, las imágenes RGB y de profundidad entregan información sin procesar, siendo una representación directa del entorno. Por otro lado, las imágenes de segmentación semántica entregan información ya procesada, siendo una abstracción del entorno. Debido a la diferencia en el tipo de información entregada por estos sensores, se consideró implementar 2 modelos diferentes, entendiéndose como 2 sistemas distintos. Al primer modelo se le denominó RGB-*Depth*, el cual utiliza las imágenes RGB y de profundidad como entrada. Al segundo modelo se le denominó Segmentación Semántica, el cual utiliza dicho sensor como entrada. Ambos modelos se presentan a continuación, de los cuales finalmente se utilizó el de segmentación semántica.

3.2.2.2 Modelo 1: RGB-*Depth*

Para este modelo se tuvo que determinar que representación se iba a utilizar para las imágenes RGB y de profundidad. Para RGB, las representaciones consideradas fueron RGB (representación original), HSV y escala de grises. En la figura 13 se muestra una observación del entorno en la imagen 1, junto con los canales RGB en las imágenes 2 a 4, y HSV en las imagenes 5 a 7. El canal valor de HSV, correspondiente a la imagen 7, es equivalente a la representación en escala de grises.

De lo que se puede apreciar, los canales de RGB no presentan gran variación entre ellos, haciéndose notar que la mayoría de los elementos en la escena son de tonalidad gris o de colores apagados (baja saturación). Por otro lado, en efecto, se pueden ver diferencias entre los canales de HSV. Para los canales de valor y saturación, se observa que los elementos son destacados de forma individual, en donde valor muestra más detalles en los elementos. En *hue*, se destacan elementos como la vegetación, el cielo y edificios, aunque muestra ruido en las zonas de color gris. Considerando esto, y como se aprecian diferencias entre saturación y valor, este último siendo equivalente a escala de grises, se decidió utilizar HSV.

En la figura 14 se observan diferentes representaciones de las imágenes de profundidad, siendo estas, de izquierda a derecha, codificación RGB (imagen original), matriz de profundidad y matriz de profundidad en escala logarítmica. Descartando la representación codificada, se considero utilizar una de las representaciones en matriz de profundidad. Se puede apreciar que la matriz de profundidad en escala logarítmica, presenta un mayor contraste entre los elementos más cercanos y los elementos más lejanos. Es por esto que se decidió utilizar esta representación. Luego de convertir las imágenes de RGB y profundidad a HSV y matriz de profundidad logarítmica, ambas imágenes son normalizadas, tomando valores entre 1 y 0.

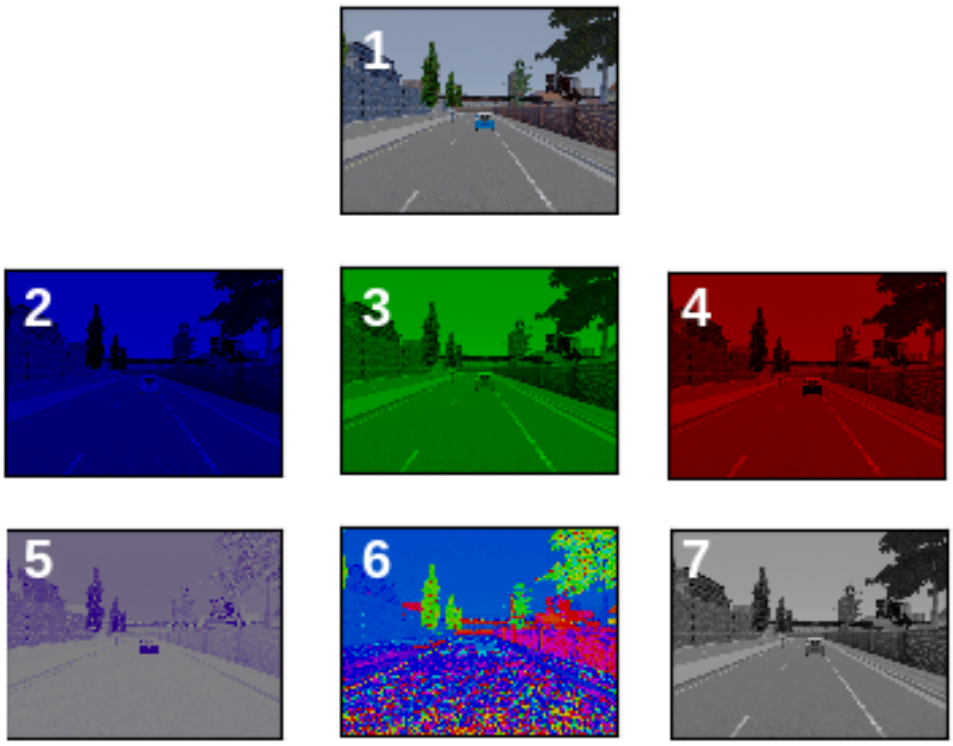


Figura 13: Imagen RGB y diferentes representaciones. En la imagen 1 se muestra la imagen original. En las imágenes 2 a 4 se muestran los canales azul, verde y rojo de la representación RGB, respectivamente. En las imágenes 5 a 7 se muestran los canales de saturación, *hue* y valor de la representación HSV, respectivamente.

Al tratarse de imágenes sin procesar, se utilizó un AE (sección 2.3.1) para la reducción de dimensionalidad del estado, conectando su espacio de latencia con una red *fully connected*. Ante esto se probaron dos enfoques, uno fue utilizar un único AE, que recibe los 4 canales como entrada, y el otro fue utilizar un AE por cada canal, siendo 4 AE por canal, cuyos espacios de latencia son posteriormente concatenados y utilizados en la red *fully connected*. Para escoger entre una de estas dos opciones, se realizaron una serie de experimentos, en donde se evaluó la similitud entra la imagen de entrada y de salida. La arquitectura de las redes utilizadas consistió en una permutación de los hiper-parámetros que se muestran en la tabla 4. Para el entrenamiento de las redes, se realizaron 250 episodios, en donde se obtuvieron 20 muestras por episodios, utilizando un total de 5000 observaciones. De estas observaciones se utilizó un 60 % para entrenamiento, 20 % para validación y 20 % para prueba.

De los resultados obtenidos, se encontró que utilizar un AE por canal es mejor que utilizar un único AE para todos los canales. Esto tiene sentido ya que cada AE se entrena para las características propias de cada canal, en vez de tener que adaptarse para funcionar bien con los 4 canales a la vez. Además de esto se realizaron las siguientes observaciones con respecto a cada uno de los hiper-parámetros de la red:

- **Resolución:** Se obtuvieron buenos resultados con imágenes de dimensiones entre 64x64 y 128x128 píxeles.

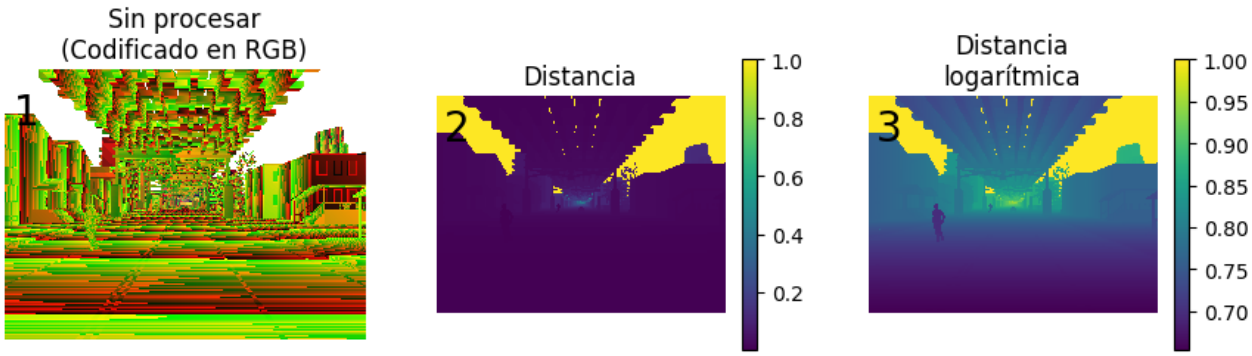


Figura 14: Imagen de profundidad y sus diferentes representaciones. En la imagen 1 se observa la imagen original codificada en RGB. En la imagen 2 se observa la imagen como una matriz de profundidad. En la imagen 3 se observa la imagen como una matriz de profundidad en escala logarítmica.

Hiper-parámetros	Valores utilizados
Resolución imagen	32x32, 64x64 y 128x128
Número de capas	2, 3 y 4
Número de filtros por capa	32-16-8-4 y 16-8-4-2
Tamaño del *kernel*	3x3, 5x5 y 7x7

Tabla 4: Parámetros utilizados en la evaluación de AEs. La reducción de número de capas se realizó a partir de las primeras capas (en el caso de 32-16-8-4, con 3 capas solo se tiene 16-8-4).

- **Número de filtros:** A mayor número de filtros, mejores resultados, aunque con el costo de tener un mayor espacio de latencia y mayor número de computaciones.
- **Número de capas:** A menor número de capas mejores resultados, pero al igual que con el número de filtros presenta un mayor espacio de latencia, aunque con un menor costo computacional.
- **Tamaño de los filtros:** No se observaron mejores resultados más allá de 3x3 y 5x5, teniéndose que filtros de tamaño 7x7 o superiores presentaron resultados similares a los de 9x9.

3.2.2.3 Modelo 2: Segmentación Semántica

Considerando que las imágenes de segmentación semántica corresponden a información pre-procesada, se escogió un método alternativo para reducir su dimensionalidad. En un primer paso, las imágenes son transformadas a su representación *one-hot-key*. De esta representación, se escogen los canales que se consideraron relevantes para realizar la maniobra de adelanto, descartando los otros canales. Los canales escogidos fueron *Road*, *Road line*, *Car* y *Other*. Luego, se redujo su resolución a 32x32, utilizando uno de los métodos descritos en la sección 3.2.1.2. Sobre esta imagen de 32x32x4, se aplicó histogramas acumulativos sobre cada uno de sus canales. Además se obtuvo la desviación estándar, valor y posición máximo de cada uno de los histogramas. Esto da como resultado un total de $(32 + 3) \cdot 2 \cdot 4 = 280$ variables, en contraste con las 4096 variables que tiene una imagen de 32x32x4. $32 + 3$ corresponde al histograma en un eje junto con las 3 estadísticas, luego se multiplica por el número de ejes (2, x e y), y finalmente se multiplica por el número de canales (4). Al tener un número reducido de variables, se puede utilizar una

red *fully connected* para aprender la maniobra de conducción y generar la acción de *Hero*.

Con respecto al método de reescalamiento, se realizaron pruebas preliminares con *Mean* y *Max*. Como se observa en las figuras 15 y 16, en las respuestas entregadas por Agente, se puede ver que *Mean* entrega una respuesta con una curvatura más suave que *Max*, teniendo una menor cantidad de componentes de baja frecuencia no deseadas y de menor magnitud. El hecho de que *Max* tenga respuestas de baja frecuencia de una magnitud considerable, se traduce en una exigencia innecesaria a los componentes del vehículo y una conducción menos confortable para el usuario. Del mismo modo, este no es una conducción típica de un humano, lo que la hace menos *human-like*. Debido a esto se escogió utilizar el método de re-escalamiento *Mean*.

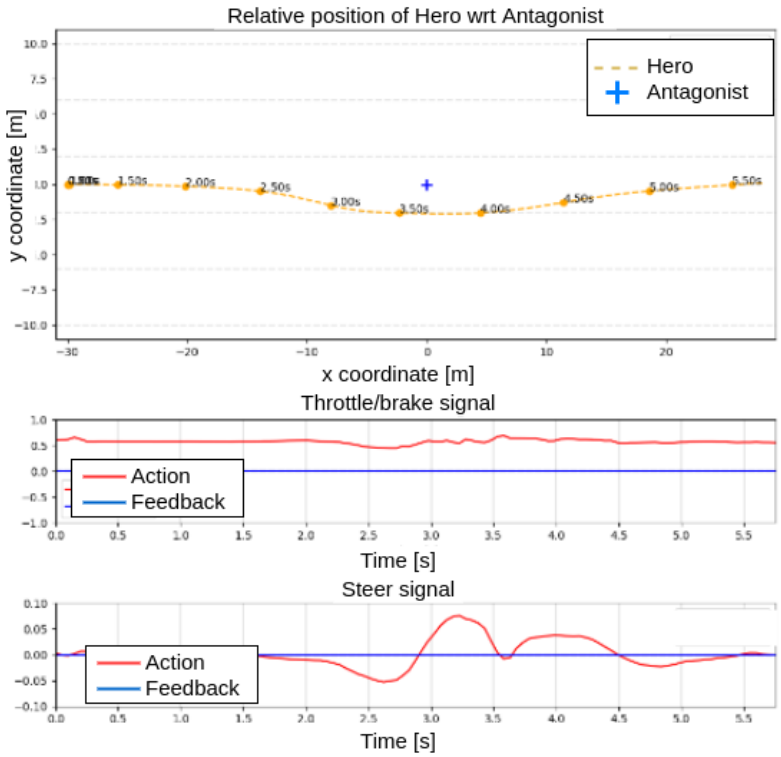


Figura 15: Trayectoria y respuesta de Agentes con segmentación semántica, utilizando método de re-escalamiento *Mean*.

RGB-*Depth* presentó peores resultados que el modelo de Segmentación Semántica, lo cual se cree es debido a que este utiliza imágenes sin procesar. Sin embargo, esto no es necesariamente un problema mayor, considerando que se tiene la ventaja de que utiliza información obtenida directamente de los sensores. El sensor de segmentación semántica, por su parte, requiere de otro elemento de procesamiento para su generación.

La principal complicación presentada en el modelo 1 fue debido a que era muy pesado computacionalmente, resultando en una simulación lenta. Esto presenta un problema para el usuario, ya que por un lado, usar este sistema en esas condiciones se vuelve bastante incómodo, haciéndolo difícil de utilizar, incluso por un breve período de tiempo. Por otro lado, una persona no esta acostumbrada a responder

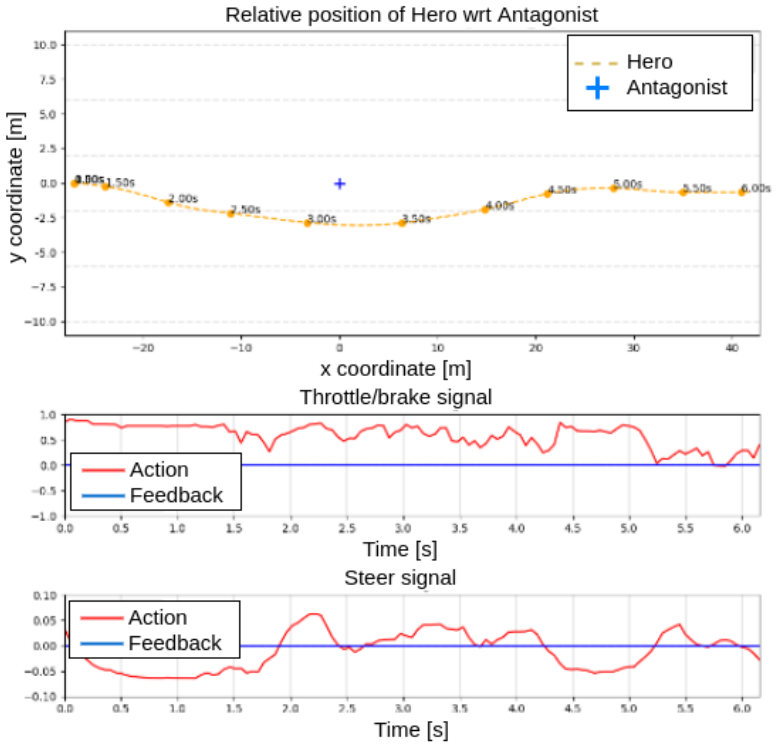


Figura 16: Trayectoria y respuesta de Agentes con segmentación semántica, utilizando métodos de re-escalamiento *Max*.

a una velocidad de ejecución diferente al tiempo, pudiendo entrenar al Agente con un comportamiento que no es *human-like*. Un ejemplo sería un usuario que entrena a un Agente para realizar las acciones de doblar y acelerar, con un tiempo de simulación más lento que el tiempo real. Durante el entrenamiento observa que se comporta correctamente, pero una vez terminado el entrenamiento, haciendo funcionar a Agente en tiempo real, se observa que este acelera demasiado rápido y realiza curvas muy pronunciadas, no siendo *human-like*.

3.2.2.4 Señales *Throttle* y *Brake*

Al evaluar los modelos, hubieron problemas al tener las señales *throttle* y *brake* de forma independiente, requiriendo entrenar constantemente al Agente para entregar 0 en la señal de *brake*, cada vez que se se requiere que el vehículo este en movimiento. Consultando trabajos relacionados a control longitudinal de vehículos autónomos [12] [13] [14] [15], se observó que estas señales son tratadas como una sola, o equivalentemente, solo una de estas señales puede ser distinta a cero en todo momento. Esto se debe a que no es recomendado presionar ambos pedales al mismo tiempo, considerando que produce el calentamiento, desgaste y perdida de energía innecesaria del vehículo, además de que querer acelerar y frenar al mismo tiempo no tiene sentido. En la figura 17 se muestra un ejemplo de esto, observándose que al aplicar ambas señales (figura (c)) se generan dos fuerzas opuestas, considerando que el movimiento del vehículo es generado por el giro de las ruedas. Es por esto que ambos pedales están al mismo lado, teniendo que ser operados con el pie derecho, incluso vehículos más modernos no permiten esta operación. Del mismo modo, al no ser una acción recomendada, esta no es realizada por conductores humanos, lo que la hace

no *human-like*.

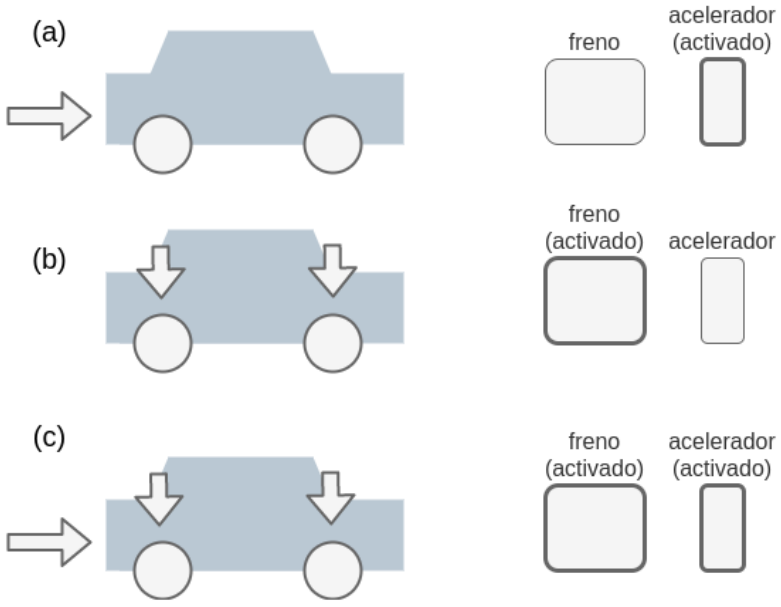


Figura 17: Diferentes ejemplos del uso de los pedales del acelerador (*throttle*) y freno (*brake*). En la figura (a) al acelerar se genera movimiento del vehículo, mientras que en la figura (b) al presionar el freno se genera una fuerza sobre las ruedas que detienen el movimiento. En la figura (c) se muestra que al presionar ambos pedales se generan dos fuerzas opuestas, que producen el desgaste del vehículo.

Teniendo esto en consideración, se decidió combinar ambas señales en la señal *throttle/brake*, la cual toma valores entre -1 y 1. Cuando *throttle/brake* toma valores positivos, se entrega una señal de *throttle* igual a la de *throttle/brake*, mientras que si *throttle/brake* toma valores negativos, se entrega una señal de *brake* igual al módulo de *throttle/brake*. Cuando una de las señales toma el valor de *throttle/brake*, la otra es igual a 0. Lo descrito anteriormente se puede ver en las siguientes ecuaciones:

$$\textit{throttle} = \begin{cases} \textit{throttle/brake}, \textit{throttle/brake} \geq 0 \\ 0, therwise \end{cases} \tag{8a}$$

$$\textit{brake} = \begin{cases} -\textit{throttle/brake}, \textit{throttle/brake} \leq 0 \\ 0, therwise \end{cases} \tag{8b}$$

3.2.2.5 Red *Fully Connected*

Una vez realizada la reducción de dimensionalidad del estado, ya sea por medio de un AE como en el modelo 1, o por otros métodos no relacionados con redes *deep* como el modelo 2, este nuevo estado es utilizado como datos de entrada para a una red *fully connected*. Esta red es entrenada con D-COACH para entregar la acción a realizar por *Hero*. Como se mencionó en la sección anterior, las señales de *throttle* y

brake fueron combinadas en la señal *throttle/brake*, por lo que la red *fully connected* entrega a su salida 2 valores entre -1 y 1, correspondientes a las señales de *throttle/brake* y *brake*.

La red *fully connected* diseñada corresponde a una red con 4 capas, 1 capa de entrada, 2 ocultas y 1 de salida. . Para determinar el número de neuronas de la red *fully connected*, considerando que se tiene un espacio de entrada mucho mayor que el de salida, se utilizó la heurística de escoger como número de neuronas ocultas, 2/3 del número de neuronas de entrada. Si se considera que se tienen n entradas, esto resulta en una capa de entrada de n neuronas, seguido por una oculta de 2/3n, luego de una de 4/9n, para finalmente tener una capa de salida con 2 neuronas. A esta red se le denominará red de acción. En la figura 18 se puede ver un la arquitectura de esta red.

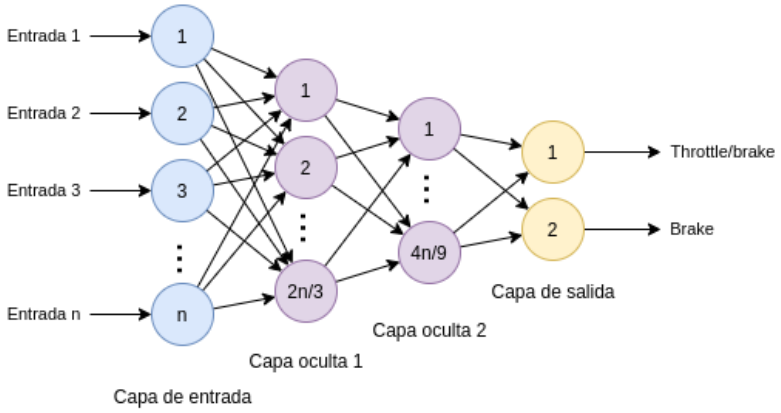


Figura 18: Red *fully-connected* encargada de generar la acción. Cuenta con una capa de entrada, dos capas ocultas y una capa de salida. Sus salidas corresponden a las señales *throttle/brake* y *steer*.

Se puede apreciar que pese a realizarse una reducción de dimensionalidad previa sobre las imágenes de las cámaras, el número de variables resultantes sigue siendo mayor a las variables cinemáticas. Para igualar el tamaño de ambas entradas, se utilizó otra red *fully connected*, la cual recibe como entrada las variables cinemáticas y entrega una representación de mayor tamaño en su salida. De no realizarse este aumento de dimensionalidad, se requeriría de un gran número de entrenamientos para conseguir que repercutan en la red *fully connected*, junto con tener pesos demasiado elevados. A esta red se le denominó red de escalamiento.

La red de escalamiento diseñada corresponde a una red de 3 capas, una capa de entrada, una capa oculta y una capa de salida. Debido a que solo se quería aumentar el número de neuronas, no siendo en principio una tarea complicada, se escogió únicamente una capa oculta. Considerando k como el número de variables proveniente de los sensores, se observa que en la capa de entrada se tienen 4 neuronas, correspondiente a las variables cinemáticas, en la capa oculta se tienen $k/4$ neuronas y en la capa de salida $k/2$ neuronas. Para la proporción entre el número de variables de las observaciones y las variables cinemáticas, probando diferentes valores se encontró que $k/2$ entregaba buenos resultados. Como el tamaño de la entrada es mucho menor que el de la salida, la heurística utilizada para el número de neuronas en la capa oculta fue un valor entre el número de neuronas en la capa de entrada y la capa de salida, siendo este $k/4$. Un diagrama de esta arquitectura se puede ver en la figura 19.

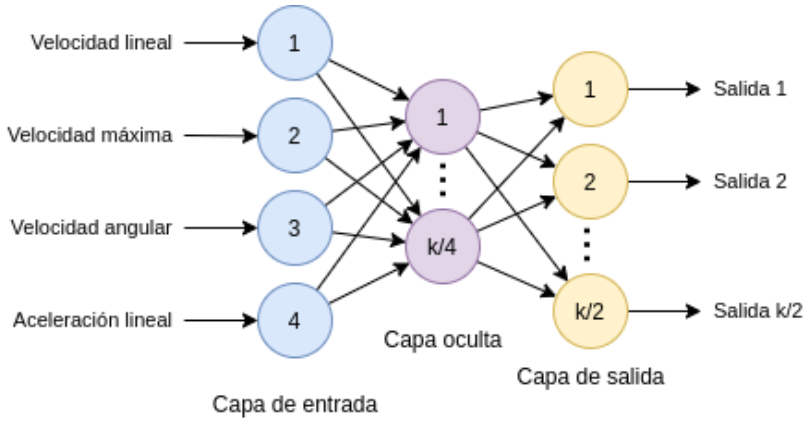


Figura 19: Red *fully-connected* de escalamiento. Cuenta con una capa de entrada, una capa oculta y una capa de salida. Sus entradas corresponden al estado cinemático de *Hero*.

3.2.2.6 *Multiplexer*

Además de realizar la maniobra de adelanto, también se requiere que el vehículo sea capaz de conducir dentro de la pista antes y después de realizar la maniobra. Para esto, se utilizó un segundo Agente y un *muxer* (*multiplexer*). El segundo Agente se encarga de hacer que el vehículo conduzca dentro de la pista, mientras que el *muxer* se encarga de alternar la respuesta entre ambos Agentes.

El Agente utilizado para la conducción corresponde a Agente PID, descrito en la sección 3.2.1.4, considerando que entrenar un Agente para realizar esto no está dentro de los objetivos. Con respecto al *muxer*, el cambio de señal se realiza de forma gradual, tardando 1 segundo en completarse. Se escogió este valor ya que para valores más pequeños es como si se hiciera un cambio instantáneo, mientras que para valores más altos, existe una interferencia en la acción de Agente principal, por parte de Agente PID. Durante esta transición, el *muxer* entrega una combinación lineal entre las respuestas de ambos Agentes. Considerando que la simulación tiene un *time-step* de 0.05 segundos, el cambio se realiza en 20 iteraciones. Un diagrama con el funcionamiento del *muxer* con los dos Agentes se puede ver en la figura 20. También se incluyen los otros componentes de Agente principal.

Para efectos del entrenamiento y experimentos, el cambio de señales se hace según la distancia a la que *Hero* este de *Antagonist*. El rango de activación del Agente en entrenamiento (el que realiza el adelantamiento) es menor al rango de desactivación. Esto es para evitar cambios reiterados e innecesarios en torno al rango de activación, los cuales se pueden producir por una reducción en la velocidad por parte del Agente en entrenamiento.

3.2.2.7 Pre-Entrenamiento

Antes de realizar el entrenamiento con D-COACH, la red es pre-entrenada para tener una base a partir de la cual comenzar a hacer las correcciones. Para eso se consideraron 2 tipos de pre-entrenamiento:

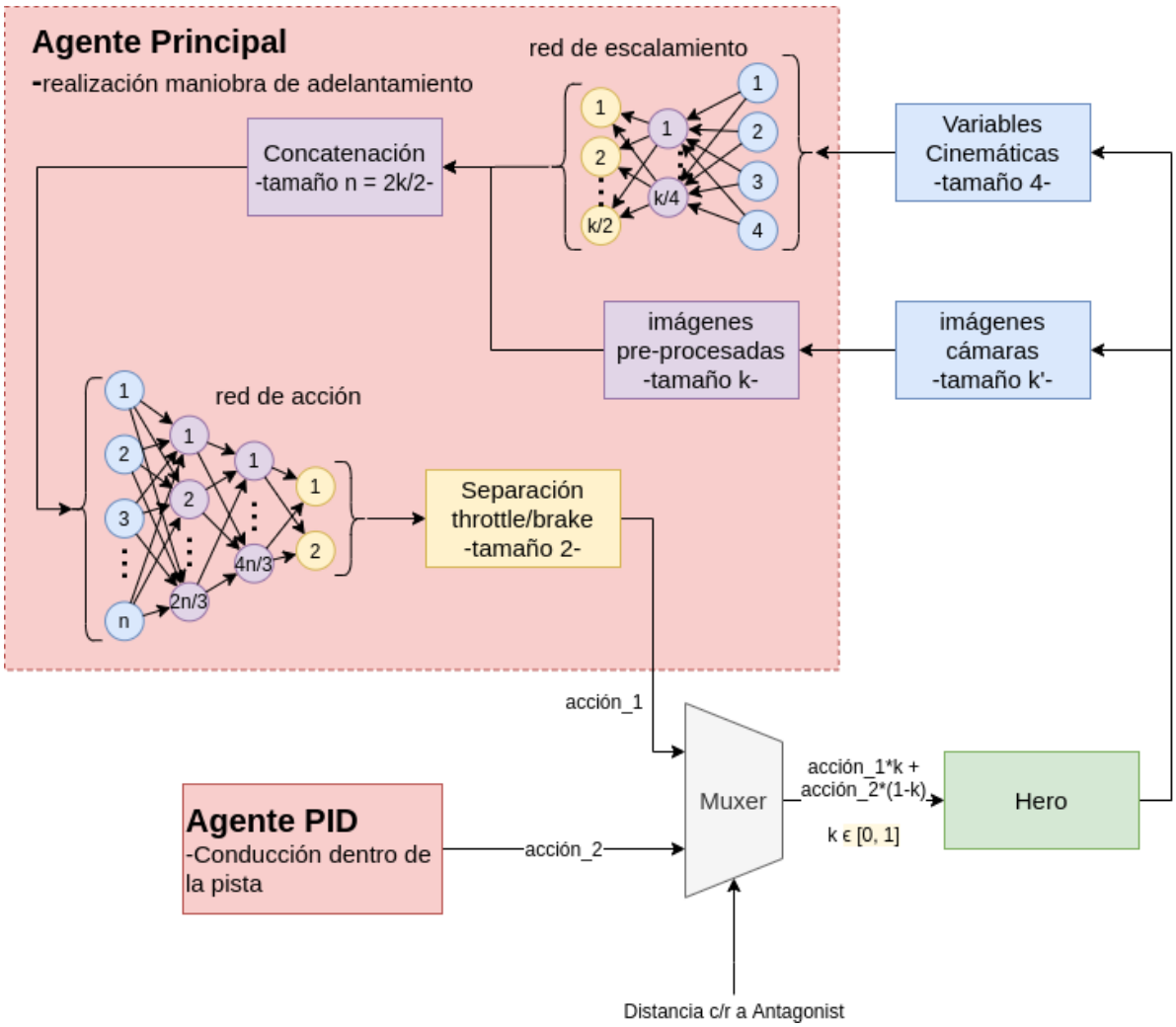


Figura 20: Diagrama de bloques del funcionamiento completo del Agente. Se aprecian las redes *fully connected* utilizadas y como se maneja la información. También se puede ver el *muxer* utilizado y el Agente PID, encargado de hacer conducir el vehículo dentro de la pista, cuando este no se encuentra realizando la maniobra de adelanto.

- **Pre-entrenado con agente PID:** En este caso la red es entrenada para comportarse como el Agente PID presentado en la sección 3.2.1.4. Este entrenamiento consta de dos etapas. En la primera etapa, se obtiene una base de datos con observaciones del vehículo, junto con las acciones realizadas por el Agente durante en ese momento. Luego de esto, se utiliza dicha base para entrenar la red. En la segunda etapa, se hace funcionar el Agente y la red de forma simultanea. En este caso *Hero* es controlado por la red, mientras que la respuesta generada por el Agente PID es utilizada para seguir entrenando la red. Este corresponde a un entrenamiento normal, utilizando pares de entrada-salida. En la primera etapa la red aprende a comportarse como el Agente, mientras que en la segunda etapa se corrigen errores acumulativos que podría tener la red.

  Los entrenamientos son realizados por episodios. Al inicio de cada episodio se generan entre 0 y 50 vehículos en torno a *Hero*, controlados con piloto automático, dentro de un radio de 30 metros.

Como solo se esta imitando el comportamiento del Agente PID, el cual solo conduce dentro de la pista, *Antagonist* no es generado. En la primera etapa se realizan 250 episodios de 1000 iteraciones cada uno, generando un total de 250.000 pares estado-acción. En la segunda etapa se realizaron 500 episodios de 1000 iteraciones cada uno, generando un total de 500.000 observaciones.

- **Pre-entrenamiento con respuesta cero:** En este caso se entrena la red para que no realice ninguna acción, o que es equivalente a que sea igual a 0. Para esto se ocupa la base de datos generada en la primera etapa del pre-entrenamiento con agente PID, pero con las acciones iguales a 0. Como se esta entrenando la red para que el vehículo se quede quieto, no se requiere realizar una segunda etapa, considerando que en este caso se requeriría de un Agente que entregue una respuesta 0. Esto sería exactamente igual a realizar la primera etapa.

Se observa que este segundo método tiene la ventaja de no requerir un agente a imitar, necesitando únicamente imágenes del entorno. Esto es útil, ya que si se quiere cambiar el comportamiento que se desea realizar, no se requiere hacer un pre-entrenamiento diferente, ni volver a implementar un Agente que imite esta conducta por otros medios. También resulta práctica en caso de que se utilice un estado diferente o el tipo de entorno cambie, ya que solo se necesita una nueva base de datos para su entrenamiento. Del mismo modo se reduce el tiempo de entrenamiento al no requerir de la segunda etapa, la cual debe ser realizada de forma *online*.

3.2.3. GUI

La GUI implementada consta de las imágenes de las cámaras RGB de *Hero*, teniendo la cámara frontal como imagen central, y la imagen generada por la(s) cámara(s) retrovisor(as) en la parte superior central. Esto es similar a una vista FPV (*First Person View*) del conductor, con el espejo retrovisor en la parte superior, pero en este caso la vista está centrada en el vehículo en vez del conductor, estando en el central en vez del lado izquierdo. También se muestra un velocímetro en la parte inferior izquierda y la acción entregada por el Agente en la parte inferior central. Además de esto, se muestra información adicional en la parte izquierda, principalmente para monitorear el correcto funcionamiento de los módulos. Un *screenshot* de la GUI se puede ver el la figura 21.

3.2.3.1 *Feedback*

Originalmente el *feedback* eran similares al del trabajo original, entregando *feedback* de forma simultanea en las señales de *throttle/brake* y *steer*, disminuyendo la señal del *throttle/brake* cada vez que se realizaba una corrección en *steer*. Estas señales fueron desacopladas debido a que en la práctica, lo que requiere mayor corrección fue la señal de *steer*. Esto producía que cuando se hacían correcciones sobre *steer*, se reducía la velocidad del vehículo. Esto es particularmente perjudicial, ya que al estar junto a otro vehículo en movimiento, cuando se reduce la velocidad, se pierde el posicionamiento del vehículo.

Finalmente los *feedbacks* definidos corresponden a los que se ven en la tabla 5, en donde el valor de la primera posición corresponde a la señal de *throttle/brake* y el de la segunda posición a la señal de *steer*.

Figura 21: GUI implementada.

Estos *feedbacks* corresponden a disminución y aumento de cada un de las señales de forma desacoplada, asignadas a las teclas de dirección.

Feedback $\{throttle/brake, steer\}$	Teclado
$\{1, 0\}$	Arriba $\uparrow$
$\{0, 1\}$	Derecha $\rightarrow$
$\{0, -1\}$	Izquierda $\leftarrow$
$\{-1, 0\}$	Abajo $\downarrow$

Tabla 5: Lista de *feedbacks*.

3.3. Experimentos

Una vez establecida la estructura de los agentes, se procedió a evaluar el desempeño de D-COACH. Para esto se entrenaron una serie de agentes con D-COACH para realizar la maniobra de adelantamiento. una vez entrenados, se les hizo realizar una serie de episodios, sin ningún tipo de intervención humana. Además de esto, se tomaron datos de personas conduciendo, para poder realizar la comparación con D-COACH. Los voluntarios fueron trabajadores del laboratorio de Robótica de Campo de la Universidad de Chile. En la figura 22 se observa una de las sesiones de toma de datos realizado.

Figura 22: Toma de datos conducción humana.

En todos los experimentos se consideró que el vehículo a adelantar viaja en torno a los 20 [km/h], además la velocidad máxima establecida fue de 30 [km/h]. Esto se determinó considerando que a mayores velocidades, el agente se vuelve más inestable, esto debido a roces con el camino y a que a mayor velocidad, mayor es la desviación de la trayectoria por variaciones de *steer*. Esto se debe a que la simulación corre a 20 [Hz], con lo que la señal de control esta por debajo de esta frecuencia, con lo que se afecta la velocidad máxima controlable. Tampoco se tomaron en cuenta señales de transito o semáforos, considerando una conducción en línea recta, doblando solo cuando era requerido. Pese a que los agentes también fueron entrenados para adelantar en presencia de una curva, para las mediciones se hicieron en base a adelantamientos en línea recta. Esto fue debido a que la obtención de datos se complicaba en estos casos y a que esto requiere más entrenamientos para obtener resultados satisfactorios.

Para las mediciones de conducción humana, se utilizó un volante Logitech G29, que se observa en la figura 23, el cual fue utilizado por los participantes para manejar a *Hero*. Los conductores manejaron bajo las mismas condiciones que los agentes, por lo que solo podían frenar, acelerar y girar el volante, sin tener que pasar cambios o poder ir en reversa. La GUI que utilizaron fue la misma que fue implementada para los entrenamientos (a excepción de los controles), disponiendo de las imágenes captadas por los sensores RGB, habiendo una cámara frontal y otra retrovisor, además de un velocímetro. Factores que pudieron afectar la conducción de las personas son el hecho de no disponer de cámaras retrovisores laterales, falta de sonido y que la cámara frontal se encontraba al centro del vehículo, en vez de a la izquierda (posición del piloto).

En total se realizaron 3 escenarios experimentales. El primero consistió en un caso simple, en el que solo están involucrados *Antagonist* y *Hero* en la maniobra de adelantamiento. En el segundo escenario,

Figura 23: Volante Logitech G29.

se complejizo la maniobra, añadiendo un nuevo actor, correspondiente a *Support Actor*, el cual también circula por la pista, afectando como se realiza la maniobra. En estos dos primeros escenarios se buscó que los agentes fueran capaces de realizar la maniobra, sin embargo, pequeñas variaciones en su conducta resultaban en incidentes en algunos de estos episodios. Considerando esto, se realizó un tercer escenario experimental, en el cual se tomaron los mejores agentes del escenario 2 y se volvieron a entrenar de forma más exhaustiva. Los entrenamientos se realizaron hasta conseguir que el uno de los agentes consiguiera cumplir con todos los episodios de forma exitosa.

Para estos experimentos se probaron 4 configuraciones del sistema, en donde el tipo de pre-entrenamiento del agente (sección 3.2.2.7) y la disposición de las cámaras de *Hero* cambian (sección 3.2.1.3). Estas configuraciones se presentan a continuación:

- ***Pre-trained*:** En esta configuración el Agente utilizado esta pre-entrenado con un agente PID y *Hero* cuenta con una cámara frontal y una cámara retrovisor.
- ***Init zero*:** En esta configuración el agente utilizado esta pre-entrenado con un respuesta cero, y *Hero* cuenta con una cámara frontal y una cámara retrovisor.
- ***Pre-trained* con cámaras laterales:** En esta configuración el agente utilizado esta pre-entrenado con un agente PID, y *Hero* cuenta con una cámara frontal y dos cámaras retrovisores laterales.
- ***Init zero* con cámaras laterales:** En esta configuración el agente utilizado esta pre-entrenado con un respuesta cero, y *Hero* cuenta con una cámara frontal y dos cámaras retrovisores laterales.

3.3.1. Experimento 1: Caso Simple

Este caso solo considera el adelantamiento del auto, sin ningún otro vehículo externo presente en la pista. Para este caso se utilizó el mapa *Town06*, debido a que cuenta con múltiples pistas de un solo sentido, que se extienden en línea recta por cientos de metros. Los experimentos comienzan dentro de estas pistas, a una distancia de 100 metros con respecto a las intersecciones. Una vista aérea del mapa se puede ver en la figura 24.

Figura 24: Mapa *Town06*.

En este caso se consideraron las 4 configuraciones descritas anteriormente, *pre-trained*, *init zero* y sus respectivas versiones con cámaras laterales. Estos fueron entrenados hasta que pudiesen realizar la maniobra de adelantamiento. Para cada agente, esto ocurrió en torno a los 30 episodios, no observándose mejoras considerables después de esto. Para su evaluación se consideraron 20 episodios por agente, correspondientes a adelantamientos en línea recta. Para este caso se estableció que *Antagonist* aparece 30 metros en frente de *Hero*, además el rango de activación y desactivación del *Multiplexer* de *Hero* fue de 25 y 40 metros, respectivamente.

Para la toma de datos de conducción humana, se les hizo realizar este escenario experimental a un grupo de 7 personas, en donde el séptimo participante (Persona 1) corresponde a quien realizó los entrenamientos con D-COACH. Los participantes tuvieron que llevar a cabo 36 episodios ordenados aleatoriamente, siendo el mismo conjunto de episodios para cada persona. Los análisis se hicieron sobre adelantamientos ocurridos en línea recta, considerando que el análisis se complicaba cuando había una curva involucrada. Debido a esto, el número de episodios utilizados fue menor que los realizados. Este número varía dependiendo de la persona, teniendo en cuenta los distintos estilos de conducción, resultando en un número diferente de adelantamientos en línea recta. El número de episodios considerados por persona se puede ver en la tabla 6, en donde el menor número de episodios considerados para una persona fue 11, y el mayor fue 24.

3.3.2. Experimento 2: Complejización del Problema

En este experimento se consideraron 3 casos, en donde en 2 de estos existe un tercer vehículo involucrado, correspondiente a *Support Actor*. Estos casos fueron:

- **Caso simple:** Similar al experimento 1, solo considera el vehículo controlado por el agente y el vehículo a adelantar.
- **Vehículo desde atrás:** En este caso un tercer vehículo se sitúa detrás de *Hero*, el cual comienza

Conductor	Episodios Totales	Episodios considerados
Persona 1	36	18
Persona 2	36	11
Persona 3	36	24
Persona 4	36	22
Persona 5	36	21
Persona 6	36	16
Persona 7	36	23

Tabla 6: Número de episodios considerados por persona para el experimento 1.

realizando la maniobra de adelantamiento. Para poder realizar la maniobra, este vehículo también fue entrenado con D-COACH. El vehículo aparece 10 metros detrás de *Hero*.

- **Vehículo por delante:** En este caso el tercer vehículo viene viajando en dirección contraria por la pista izquierda. Este vehículo aparece 100 metros más adelante de *Hero*. El vehículo esta controlado por el piloto automático, por lo que se rige por las normas del tránsito. Esto incluye respetar la velocidad máxima real de la pista o detenerse frente a semáforos en rojo.

Figura 25: Mapa *Town01*.

Para este experimento se utilizó el mapa *Town01*, que se puede ver en la figure 25. Este mapa esta compuesto por calzadas de doble sentido y 2 carriles, por lo que se puede realizar el tercer caso. Sin embargo, las pistas son más angostas y las secciones rectan son más cortas, además de que existe un mayor número de intersecciones. Otra diferencia con el mapa *Town06* es que éste cuenta con más elementos en torno a la calzada, por lo que existe un mayor riesgo de chocar en caso de salirse de ella.

Al tenerse pistas más cortas, ahora *Antagonist* aparece a 17 metros delante de *Hero*. Del mismo modo los valores de activación y desactivación de *muxer*, tanto para *Hero* como para *Support Actor* (en el 2do

caso), fueron de 15 y 30 metros, respectivamente.

En este escenario experimental se utilizaron 6 configuraciones, siendo las 4 configuraciones principales mencionadas anteriormente (*init zero*, *pre-trained* y sus versiones con cámaras laterales) y 2 configuraciones nuevas. Una de estas configuraciones es *old pre-trained*, en donde el agente es pre-entrenado con un agente PID y posee una cámara frontal y trasera, pero se distingue por tener sus cámaras a una altura de 2.4 [m]. Esto corresponde a *pre-trained* pero con sus cámaras a una mayor altura. Esto corresponde a la posición inicial de las cámaras, pero que posteriormente se cambió a 1.5 [m]. Se uso esta configuración para contrastar con la nueva posición. La otra configuración es *init zero other*, en donde una persona sin experiencia previa utilizando D-COACH entrena al agente. *Init zero other* utiliza la misma configuración que *init zero*. En la tabla 7 se muestran las 6 configuraciones descritas.

Configuración	Descripción
Init zero	Agente pre-entrenado con respuesta cero, con una cámara frontal y una trasera.
Init zero con cámaras laterales	Agente pre-entrenado con respuesta cero, con una cámara frontal y dos cámaras laterales.
Pre-trained	Agente pre-entrenado con agente PID, con una cámara frontal y una trasera.
Pre-trained con cámaras laterales	Agente pre-entrenado con agente PID, con una cámara frontal y dos cámaras laterales.
Old pre-trained	Misma configuración que *pre-trained* pero con sus cámaras a 2.4 [m] de alto.
Init zero other	Agente entrenado por un usuario sin experiencia previa con D-COACH.

Tabla 7: Configuraciones utilizadas en escenario experimental 2.

Nuevamente, los agentes fueron entrenados hasta que se consideró que entregaban buenos resultados, siendo esto cuando se observaba cierta regularidad en la realización completa de las maniobras. Esto correspondió a entrenamientos en torno a los 50 episodios. Como en este experimentos se tenían casos más complejos, ocurría que para situaciones puntuales, los agentes fallaban por poco, doblando poco o más de la cuenta. Para corregir esto, después de la primera ronda de entrenamientos, se volvía a realizar una segunda ronda, en esta oportunidad buscando corregir estos casos específicos. Estos ajustes se llevaron a cabo por un periodo entre 30 minutos y una hora, sin embargo se pudieron haber prolongado por un mayor período de tiempo.

Para la evaluación se realizaron 30 episodios, siendo 10 por cada tipo. Cada agente y participante

tuvieron que realizar el mismo conjunto de 30 episodios, los cuales fueron ordenados aleatoriamente. En este experimento se obtuvieron las mediciones de 4 personas, quienes realizaron los episodios en orden aleatorio, enumerados de la misma forma que en el experimento 1 (Persona 1, Persona 2, etc…). Dentro de los conductores humanos, Persona 1 fue quien entreno la mayoría de los agentes, mientras que Persona 4 entrenó a *init zero other*. Para los conductores humanos, si la maniobra se extendía por un prolongado periodo de tiempo y no se sentías capaces de finalizar la maniobra, esta terminaba, siendo considerada como un fracaso.

3.3.3. Experimento 3: Entrenamiento Extendido

En este experimento se escogieron los agentes con mejor desempeño y se siguieron entrenando, hasta que uno de ellos obtuvo los resultados esperados. Estos agentes fueron *init zero* e *init zero* con cámaras laterales. Como se inició el entrenamiento con los mismos agentes, estos ya eran capaces de realizar la maniobra con una alta taza de éxito. La ronda de entrenamientos fue similar a la segunda ronda del experimento anterior, buscando solo corregir fallas puntuales en los agentes. Esta ronda duró en torno a 2 horas por agente. Considerando que esto una extensión del segundo escenario experimental, se tienen los mismos casos y condiciones en este experimento.

4. Análisis y Resultados

4.1. Experimento 1: Caso Simple

4.1.1. Trayectoria Promedio

Para poder analizar la conducción de los agentes y humanos, y realizar la comparación entre ambos tipos de conducción, se obtuvo la trayectoria descrita por *Hero* relativa a *Antagonist*, consiguiendo una descripción clara de como se llevó a cabo la maniobra. Si se utilizaba las trayectorias absolutas de *Hero* y *Antagonist*, se tenían dos curvas que se interponían por gran parte del episodio. Al usar la trayectoria relativa a *Antagonist*, la trayectoria se deformaba en las curvas, es por esto que solo se consideraron adelantamientos realizados en línea recta.

El análisis se llevó a cabo sobre el promedio de las trayectorias relativas, realizando una interpolación espacial sobre ellas antes de promediar. La interpolación se hizo debido a que, de lo contrario, el promedio sería en función del tiempo, siendo susceptible a deformaciones causadas por diferencias de velocidad. Un caso extremo de esto sería si, para un tiempo determinado, el vehículo se encontraba detrás de *Antagonist* en un episodio y y delante de él en otro, teniéndose que en promedio *Hero* se encontraba sobre *Antagonist*, aunque esto nunca ocurrió en ningún episodio. Considerando el eje x como paralelo a la pista, y el eje y perpendicular al eje x, la interpolación se realizó a lo largo del eje x. Con esto se obtiene una aproximación de la posición de *Hero* en el eje y para cada posición de x, manteniendo la forma general de la trayectoria al realizar la maniobra de adelantamiento. Cabe mencionar que al realizar esta interpolación se pierde la noción de cuando el auto retrocede o se mantiene en la misma posición con respecto a *Antagonist*, aunque de todas formas los retrocesos (relativos a *Antagonist*) observados suelen ser pequeño. Del mismo modo, si se cambia de pista a diferentes distancias, se observará una mayor varianza en dichas zonas.

En la figura 26 se muestran las trayectorias promedio generada por los agentes, mientras que la figura 27 se muestran las trayectorias promedio generada por los conductores humanos. Además de la trayectoria promedio, se observan la desviación estándar para cada punto en el eje x, siendo las áreas grises en torno a las curvas. Mientras más clara es el área mayor es la desviación estándar. También se observa una cruz azul en el origen que representa a *Antagonist*, y líneas punteadas que representan el borde de las pistas. En los gráficos de los agentes puede observar el rango de activación del *multiplexer* (circunferencia roja), junto con su rango de desactivación (circunferencia naranja).

Los agentes muestras una trayectoria similar al de los humanos, aunque con una curvatura más irregular. También se aprecia que los agentes tienden a moverse más apegado al vehículo adelantado, aunque no es exclusivo de ellos. Con respecto a la variabilidad, los conductores humanos tienden a tener una mayor variación en los tramos en donde se realiza el cambio de pista, mientras que los agentes muestran una trayectoria más regular. El agente *init zero* muestra una mayor variación a lo largo de toda su trayectoria, mientras que el agente *pre-trained* con cámaras laterales muestra variaciones al final de esta.

Considerando que la Persona 7 fue quien entrenó los agentes, se observa que la similitud entre la trayectoria de este conductor y de los agentes, no es mayor a la de los agentes con los otros conductores. Incluso se puede ver que la trayectoria de la Persona 7 es más parecida a la de las otras personas que a la de los agentes. Esto muestra que aunque con D-COACH una persona puede generar un comportamiento

determinado en una red, esto no significa necesariamente que vaya a comportarse exactamente como él. Esto no necesariamente significa algo malo, teniéndose que por medio de D-COACH, una persona es capaz de enseñarle a una red a realizar una acción que ni él puede llegar a realizar.

4.1.2. Mediciones y Rankings

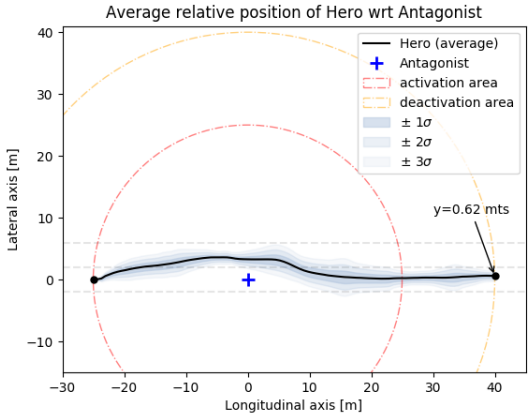
A continuación se muestran una serie de mediciones realizadas sobre las trayectorias generadas en cada episodio por cada uno de los agentes y conductores humanos. Para cada agente se tomo el promedio, desviación estándar y valores mínimos y máximos de los valores obtenidos para cada trayectoria. Estos valores son presentados en tablas, ordenadas de menor a mayor sobre el promedio.

La tabla 8 muestra la velocidad a la que viajaron cada conductor. Se muestra que los humanos fueron mejores que los agentes para respetar la velocidad máxima establecida, siendo esta 30 km/h. Esto también se aprecia en las velocidades máximas registradas. Esto se puede deber a que por un lado, el entrenamiento se centró más en la realización de la maniobra de forma correcta, dejando el tema de la velocidad como algo secundario. Por otro lado, considerando que el vehículo estaba en modo automático, el cambio se hacía al superar los 30 km/h, acelerando hasta los 40 km/h, luego se tenía el hecho de requerir una velocidad alta para poder hacer la maniobra rápidamente, y que existe cierta variabilidad en la respuesta de los agentes. Una solución a esto podría ser añadir el cambio del vehículo a la entrada de la red. Las velocidades máximas registradas corresponden al cambio de pista, el cual al ser un proceso rápido, es más difícil entregar múltiples *feedbacks*. Para el caso de *init zero* con cámaras laterales, se hace notar que se obtiene un valor particularmente alto debido que en este caso particular, una vez adelantado el vehículo, se hizo acelerar para terminar el episodio más rápido.

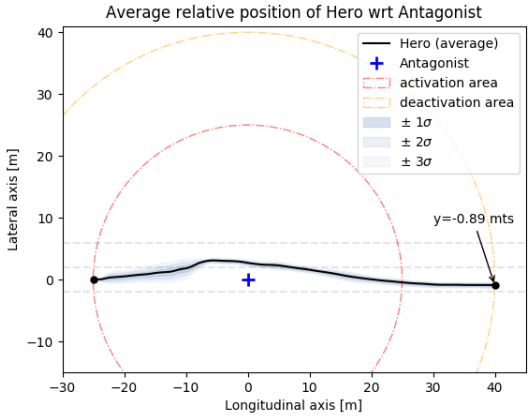
Conductor	Promedio [km/h]	STD [km/h] (σ)	Min [km/h]	Max [km/h]
Persona 2	28.79	3.58	0.00	41.77
Persona 1	29.97	5.45	0.00	53.78
Persona 3	31.75	8.62	0.00	70.66
Persona 6	33.36	5.06	0.00	57.30
Persona 4	33.97	7.02	0.00	71.32
Init zero	37.71	5.27	0.00	61.48
Persona 5	37.95	10.20	0.00	79.24
Pre-trained	41.15	8.38	0.00	70.65
Pre-trained side cams	44.82	7.78	0.00	67.82
Persona 7	45.90	15.73	0.00	95.76
Init zero side cams	58.03	15.24	0.00	103.14

Tabla 8: Ranking de velocidad [km/h].

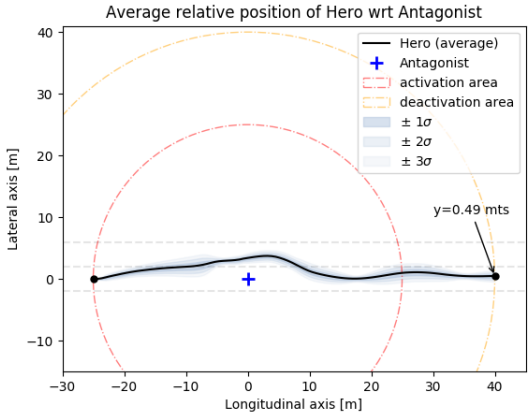
En las tablas 9 y 10 se muestra la posición en la que se deja y retoma la pista inicial. Esta posición corresponde a la coordenada x de *Hero* relativo a *Antagonist*. Se debe considerar que ambos vehículos tienen una longitud de 4 [m] y un ancho de 1.75 [m], teniéndose que la posición de los vehículos corresponden al centro de masa, ubicado en el centro del vehículo. El eje x es paralelo a la pista, siendo x=0 el centro de *Antagonist*, con valores negativos detrás de y positivos delante de este. En la figura28 se puede



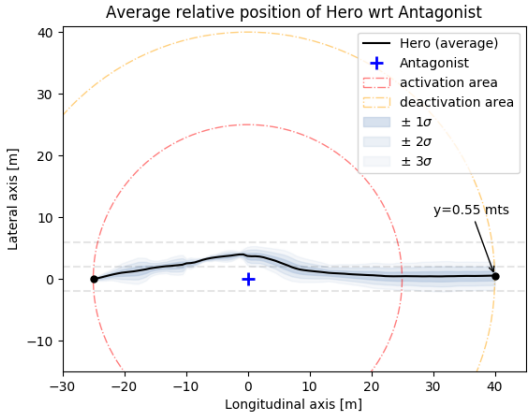
(a) *Init zero*



(b) *Pre-trained*

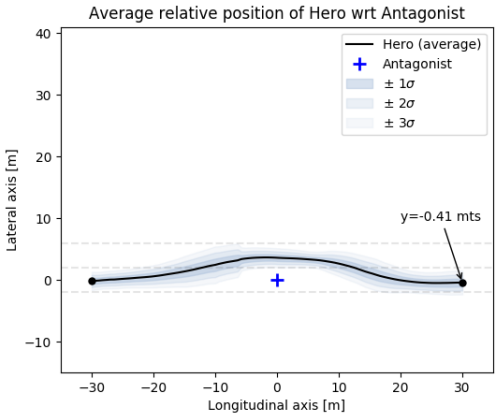


(c) *Init zero* con cámaras laterales

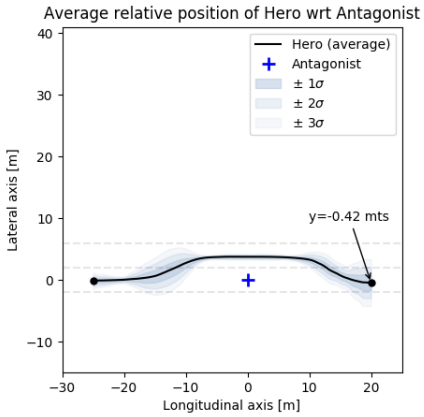


(d) *Pre-trained* con cámaras laterales

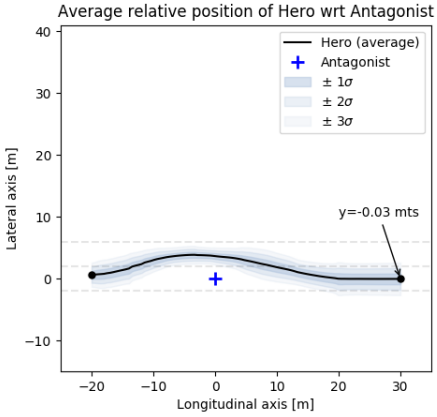
Figura 26: Trayectoria promedio de conducción por agentes, relativo a *Antagonist*.



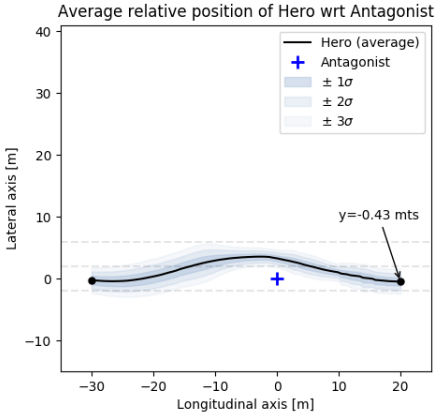
(a) Persona 1



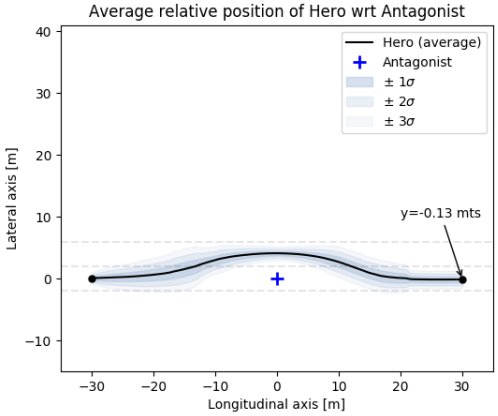
(b) Persona 2



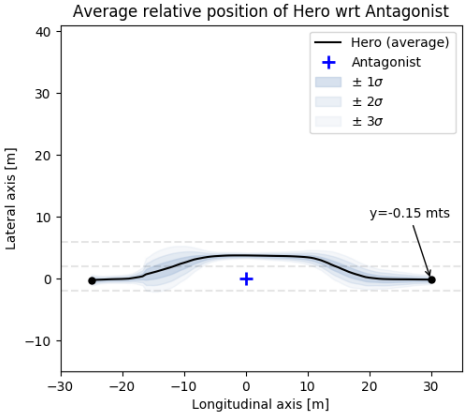
(c) Persona 3



(d) Persona 4

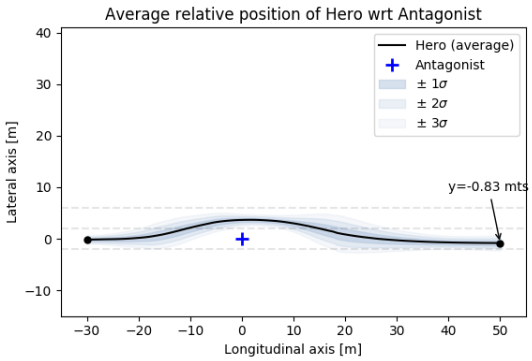


(e) Persona 5



(f) Persona 6

Figura 27: Trayectoria promedio de conducción humana, relativa a *Antagonist*.



(g) Persona 7

Figura 27: Trayectoria promedio de conducción humana, relativa a *Antagonist* (cont.).

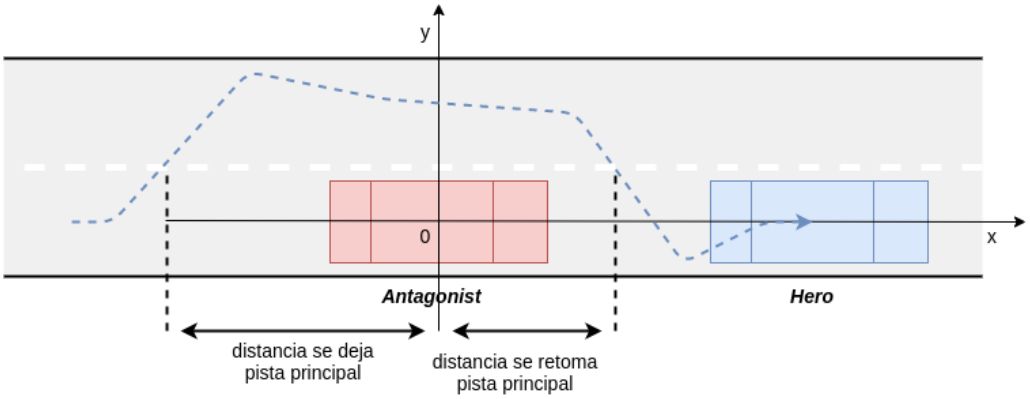


Figura 28: Medición distancia relativa a la que se deja y se retoma la pista inicial.

ver un diagrama de dichas mediciones.

Considerando que la dimensión de los vehículos, todos los valores se encuentran a distancias razonables. Se observa una mayor diferencia entre la distancia a la que se retoma la pista, siendo esta de hasta 10 metros, mientras que la diferencia a la distancia que se deja la pista es de hasta 5 metros.

Se observa que los agentes tienden a dejar la pista inicial a una menor distancia del vehículo que adelantan, aunque existe traslape con los conductores humanos. Sin embargo, se pude ver que los conductores humanos que dejan después la pista, también tienden a retomarla después y viceversa. Un ejemplo de esto es Persona 1, que pese a dejar la pista más cerca a *Antagonist* (puesto 2 en tabla 9), la retoma más lejos (puesto 11 en tabla 10). El caso contrario se observa con Persona 4. Los valores más bajos de los agentes también se pueden explicar a que estos tienden a tener una trayectoria más triangular, mientras que los conductores humanos tienden a tener una trayectoria más trapezoidal. Esto se puede deber a la dificultad que tienen los agentes de controlar la velocidad.

Conductor	Promedio [m]	STD [m] (σ)	Min [m]	Max [m]
Pre-trained	-10.15	2.37	-18.43	-8.44
Persona 7	-10.79	3.51	-21.06	-5.19
Init zero side cams	-10.92	3.74	-16.22	-4.97
Persona 2	-12.03	2.37	-17.34	-9.01
Persona 6	-12.05	2.75	-18.10	-7.73
Persona 1	-12.14	3.49	-18.28	-6.19
Persona 3	-13.61	2.98	-20.10	-9.49
Pre-trained side cams	-14.11	2.72	-19.47	-10.56
Persona 4	-14.33	3.61	-22.25	-7.77
Persona 5	-14.41	4.03	-23.40	-8.50
Init zero	-15.89	3.81	-21.11	-11.07

Tabla 9: Ranking de distancia longitudinal relativa a la que se deja la pista inicial [m].

Conductor	Promedio [m]	STD [m] (σ)	Min [m]	Max [m]
Persona 4	5.81	2.71	1.04	14.14
Pre-trained	6.88	1.50	3.17	8.89
Pre-trained side cams	7.32	2.70	5.17	12.09
Init zero	8.06	1.74	3.24	11.22
Init zero side cams	8.37	0.88	6.93	10.68
Persona 3	10.70	5.95	3.80	29.42
Persona 5	11.98	3.10	5.68	21.85
Persona 1	12.40	2.46	8.32	18.13
Persona 2	12.74	1.19	11.12	14.95
Persona 6	14.75	2.07	11.89	21.07
Persona 7	15.81	4.06	8.19	24.14

Tabla 10: Ranking de distancia longitudinal relativa a la que se vuelve a retomar la pista inicial [m].

En las tablas 11 y 12 se observa la distancia recorrida por los vehículos fuera de la pista inicial. En el caso de la tabla 11, esta representa la distancia absoluta, mientras que la tabla 12 representa la distancia relativa a *Antagonist*. Estos valores están aproximados, siendo calculados como la diferencia de las coordenadas paralelas a la pista, en el momento en el que se deja y retoma la pista. Siendo x la coordenada paralela absoluta y x' la coordenada paralela relativa, estos valores se calcularon como:

$$length_{out} = x_{in} - x_{out} \tag{9}$$

$$rel\_length_{out} = x'_{in} - x'_{out} \tag{10}$$

En el caso de la distancia absoluta, no se observan diferencias entre los distintos tipos de conductores. Dentro del rango de valores obtenidos, tanto humanos como agentes se encuentran a lo largo de este rango, aunque los agentes tienden a posicionarse en los extremos. La mayor diferencia es de 40 metros, siendo esta entre 2 agentes (ambos siendo los dos casos de *init zero*). Por otro lado, se puede ver que para la distancia relativa, los agentes tienden a recorrer una menor distancia que los humanos, habiendo una diferencia de hasta casi 10 metros. Nuevamente se confirma que los agentes recorrer una menor

distancia fuera de la pista, al menos con respecto al vehículo adelantado. Tampoco se observan diferencias considerables entre cada tipo de agente.

Conductor	Promedio [m]	STD [m] (σ)	Min [m]	Max [m]
Init zero side cams	51.73	14.37	29.58	80.90
Persona 7	55.27	12.65	37.16	83.01
Persona 4	60.04	26.25	44.37	89.93
Pre-trained	65.85	10.21	47.03	92.34
Persona 5	68.17	14.09	46.68	97.62
Persona 3	70.15	13.65	47.80	93.63
Persona 1	71.95	13.24	56.57	105.30
Persona 2	79.70	12.79	62.84	106.58
Pre-trained side cams	81.36	11.51	56.28	96.41
Persona 6	89.46	18.02	62.56	140.03
Init zero	93.16	20.67	69.86	129.62

Tabla 11: Ranking de distancia longitudinal absoluta recorrida fuera de la pista [m].

Conductor	Promedio [m]	STD [m] (σ)	Min [m]	Max [m]
Pre-trained	17.03	2.74	12.02	25.81
Init zero side cams	19.29	4.26	12.37	25.75
Persona 4	20.15	4.00	13.48	30.07
*Pre-trained side cams*s	21.43	4.35	15.74	27.75
Init zero	23.95	4.92	18.23	31.46
Persona 3	24.31	6.97	15.27	41.28
Persona 1	24.54	4.65	16.64	36.41
Persona 2	24.77	2.44	21.78	30.09
Persona 5	26.40	5.15	17.90	39.00
Persona 7	26.60	5.60	17.34	41.05
Persona 6	26.80	4.23	20.38	39.16

Tabla 12: Ranking de distancia longitudinal relativa recorrida fuera de la pista [m].

La tabla 13 muestra el tiempo transcurrido fuera de la pista. Se aprecia que los agentes tienden a permanecer menos tiempo fuera e la pista, aunque existe una gran traslape entre los resultados de agentes y humanos. Observando los resultados de distancia recorrida fuera de la pista y la velocidad a la que viajan los conductores, estos resultados tienen sentidos. En particular se observa que *init zero* con cámaras laterales pasa menos de 4 segundos fuera de la pista, aunque al mismo tiempo viaja a una velocidad promedio de 60 km/h.

En la tabla 14 se puede ver la distancia mínima observada entre los centros de masa de los vehículos. En la figura 29 se puede ver un diagrama de esta medición. Nuevamente los agentes tienen a permanecer más cerca, aunque también existe traslape con los valores obtenidos de los conductores humanos. Asumiendo que esta distancia mínima ocurre cuando el vehículo pasa junto al otro y considerando que su ancho es de 1.75 [m], se tiene que por lo menos hay 1 [m] de distancia entre cada vehículo, siendo una

Conductor	Promedio [s]	STD [s] (σ) [s]	Mi [s]n	Max [s]
Init zero side cams	3.45	1.31	1.75	6.85
Persona 7	4.16	1.76	1.95	8.90
Pre-trained	6.18	0.74	4.70	7.85
Persona 5	6.25	1.90	3.65	10.85
Persona 4	6.47	1.30	2.65	8.35
Pre-trained side cams	6.58	0.87	4.55	7.50
Persona 3	7.77	1.63	4.35	10.00
Persona 1	8.59	1.80	5.70	12.70
Init zero	8.87	2.03	6.65	12.60
Persona 6	9.71	2.15	6.20	15.00
Persona 2	9.95	1.96	6.75	13.85

Tabla 13: Ranking de tiempo recorrido fuera de la línea [s].

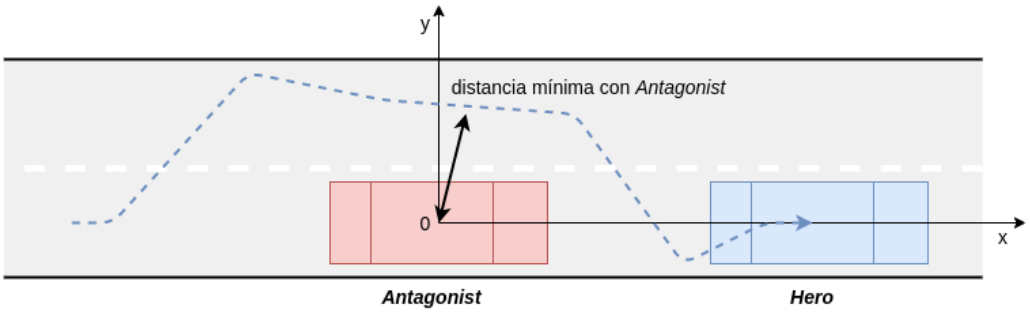


Figura 29: Distancia mínima entre *Antagonist* y *Hero*.

distancia razonable.

Conductor	Promedio [m]	STD [m] (σ)	Min [m]	Max [m]
Pre-trained	2.69	0.17	2.36	3.06
Persona 4	3.19	0.50	2.26	4.55
Init zero side cams	3.32	0.23	2.88	3.83
Init zero	3.33	0.50	2.29	4.08
Persona 1	3.58	0.50	2.46	4.47
Persona 3	3.60	0.47	2.79	4.74
Pre-trained side cams	3.60	0.43	3.19	4.33
Persona 7	3.64	0.39	2.91	4.53
Persona 6	3.76	0.16	3.37	4.08
Persona 2	3.76	0.16	3.58	4.08
Persona 5	4.10	0.37	3.55	4.79

Tabla 14: Ranking de distancia mínima entre *Hero* y *Antagonist* [m].

En las tablas 15 se puede ver la desviación máxima de la coordenada y, relativa a *Antagonist*. Consi-

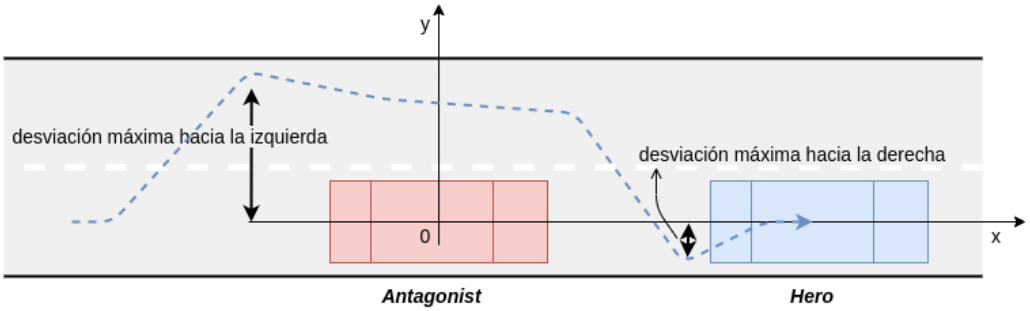


Figura 30: Medición desviación máxima y mínima de *Hero* en el eje y.

derando que y es perpendicular a la pista, esto se traduce en la desviación máxima del vehículo hacia la izquierda. En la figura 30 se observa un diagrama de dicha medición, junto con la desviación mínima de la coordenada y. Se observa que los agentes se mantienen dentro del rango observado por los humanos, aunque para los agentes, estos valores tienden a ser menores. Estos aún se mantienen dentro de un margen seguro con respecto al otro vehículo (>1.75 [m]). Del mismo modo, ningún valor máximo supera los 5 metros, lo cual consistiría en pasar a la siguiente pista (6 metros menos el metro de distancia entre el centro de masa y el extremo del vehículo).

Conductor	Promedio [m]	STD [m] (σ)	Min [m]	Max [m]
Pre-trained	3.15	0.13	2.86	3.36
Persona 4	3.68	0.47	2.81	4.82
Init zero	3.73	0.41	3.27	4.34
Init zero side cams	3.77	0.31	3.19	4.54
Persona 7	3.79	0.38	3.07	4.60
Persona 2	3.85	0.15	3.63	4.09
Persona 6	3.88	0.15	3.56	4.09
Persona 1	3.88	0.48	3.00	4.72
Persona 3	3.92	0.44	3.28	4.94
Pre-trained side cams	4.17	0.34	3.77	4.65
Persona 5	4.18	0.37	3.55	4.88

Tabla 15: Ranking de desviación máxima lateral del vehículo hacia la izquierda [m].

La tabla 16 representa la desviación mínima de la coordenada y, relativo a *Antagonist*, que igual al caso anterior, representa la desviación máxima hacia la derecha. Nuevamente, en la figura 30 se puede ver un diagrama de esta medición. Se observa que los agentes no tienen a superar $y = 0$, correspondiente al centro de la pista. Por otro lado los conductores humanos tienden a llegar hasta $y = -1$, siendo este casi el borde derecho de la pista, teniéndose casos en donde incluso fueron superados (Min). En este sentido, los agentes tienden a conducir mejor que los humanos.

De los resultados obtenidos, se aprecia que los agentes tienden a realizar la maniobra más apegados al vehículo que están adelantando, pero dentro de un rango seguro. Del mismo modo se desvían menos hacia los lados. Por otro lado, comparado con los conductores humanos, les cuesta mantener la velocidad establecida, esto debido a las razones mencionadas anteriormente, y la curvatura de sus trayectorias suele

Conductor	Promedio [m]	STD [m] (σ)	Min [m]	Max [m]
Init zero	-0.01	0.10	-0.43	0.05
Init zero side cams	-0.10	0.14	-0.51	0.01
Pre-trained side cams	-0.11	0.12	-0.36	0.00
Persona 3	-0.24	0.68	-1.69	0.67
Persona 6	-0.35	0.30	-1.18	-0.04
Persona 5	-0.38	0.38	-1.21	0.43
Persona 2	-0.74	1.19	-1.95	0.16
Persona 1	-0.82	0.54	-1.89	0.36
Persona 7	-0.92	0.43	-1.83	-0.02
Pre-trained	-0.95	0.16	-1.31	-0.73
Persona 4	-0.96	0.58	-2.06	0.06

Tabla 16: Ranking de desviación mínima lateral del vehículo, equivalente a la desviación máxima hacia la derecha [m].

ser más irregular. De todos modos, de acuerdo a las mediciones realizadas, los valores obtenidos para los agentes suelen ser similares a los de los humanos. Con respecto a cual de los agentes es mejor, no se puede decir con certeza, teniéndose que la mayoría de sus resultados están dentro de rangos aceptables y que estos cumplen con su objetivo.

4.2. Experimento 2: Complejización del Problema

4.2.1. Resultados de Episodios

4.2.1.1 Caso Simple

En la tabla 17 se observan los resultados obtenidos para el caso simple. En este caso, tanto los humanos como los agentes de *init zero* cumplieron con todos los 10 episodios. Por otro lado los agentes *pre-trained* y entrenado por un tercero fallaron en uno de los episodios. Finalmente el agente *pre-trained* con la antigua configuración presento peores resultados, fallando 3 de los 10 episodios.

Conductor	Éxito	Fracaso
Init zero	10	0
Init zero side cams	10	0
Persona 1	10	0
Persona 2	10	0
Persona 3	10	0
Persona 4	10	0
Pre-trained	9	1
Pre-trained side cams	9	1
Init zero other	9	1
Old pre-trained	7	3

Tabla 17: Resultado episodios caso simple.

4.2.1.2 Vehículo desde Atrás

En la tabla 18 se observan los resultados obtenidos para el caso en el que existe un vehículo detrás de *Hero*, el cual también se encuentra realizando la maniobra de adelanto. Para este caso existieron 3 posibles resultados, realizar el adelantamiento luego de que el vehículo de atrás realice la maniobra, realizar el adelantamiento antes de que el vehículo de atrás realice la maniobra o fallar el episodio. Para el primer resultado esto significa que el vehículo esperó detrás del vehículo de enfrente, hasta que el tercer vehículo concluyera su maniobra. Los dos primeros resultados se consideran exitosos, aunque se considera el primer resultado más correcto que el segundo, al esperar que el vehículo de atrás realice el adelanto.

Para el caso de los humanos, se observa que la mayoría completó los episodios exitosamente, aunque algunos de estos casos correspondieron a adelantamientos antes de que el vehículo de atrás avance. Los casos fallidos corresponden a episodios que se alargaron por mucho tiempo, y que los conductores prefirieron terminar.

Con respecto a los agentes, *init zero* logra los 10 episodios de manera correcta. Para los casos con cámaras laterales y *pre-trained* con una cámara retrovisor, estos cumplen todos o casi todos los episodios (siendo 10/10 y 9/10 respectivamente), aunque algunos de estos casos son adelantando antes que el otro vehículo. Finalmente los que presentaron peores resultados fueron *init zero other* y *old pre-trained*, fallando 2 episodios, y en los exitosos, la mayoría corresponden a adelantar antes que el otro vehículo. Una posible explicación a este comportamiento puede ser por falta de entrenamiento, considerando que este no fue un entrenamiento exhaustivo.

Conductor	Adelanto después	Adelanto antes	Fracaso	Tasa de éxito
Init zero	10	0	0	10/10
Persona 1	10	0	0	10/10
Pre-trained	8	1	1	9/10
Persona 3	8	0	2	8/10
Persona 4	8	1	1	8/10
Pre-trained side cams	7	3	0	10/10
Persona 2	7	3	0	10/10
Init zero side cams	6	4	0	10/10
Old pre-trained	3	5	2	8/10
Init zero other	2	5	3	7/10

Tabla 18: Resultados episodio vehículo desde atrás.

4.2.1.3 Vehículo de Frente

En la tabla 19 se observan los resultado obtenidos, en el que un vehículo viene conduciendo por la pista izquierda en dirección contraria. En este caso todas los conductores humanos junto con *init zero* completaron los 10 episodios de forma satisfactoria. Para los otros agentes, estos completaron 9 de los 10 episodios, a excepción de los casos de *pre-trained*, en donde fallaron 4 y 8 episodios respectivamente. En este caso (*old pre-trained*) obtuvo mejores resultados que su contra parte (*pre-trained*). Esto tiene

sentido ya que al tener la cámara por sobre los otros vehículos, es capaz de ver mejor al vehículo que se aproxima por la pista izquierda. Sin embargo, también se tienen agentes con cámaras lateral que tuvieron los mismos o incluso mejores resultados, los cuales utilizaron la configuración actual.

Conductor	Éxito	Fracaso
Init zero	10	0
Persona 1	10	0
Persona 2	10	0
Persona 3	10	0
Persona 4	10	0
Old pre-trained	9	1
Init zero side cams	9	1
Init zero other	9	1
Pre-trained side cams	6	4
Pre-trained	2	8

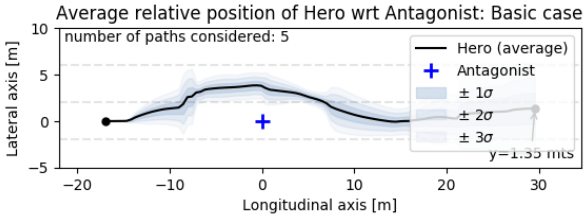
Tabla 19: Resultados episodio vehículo de frente.

4.2.2. Número de Colisiones

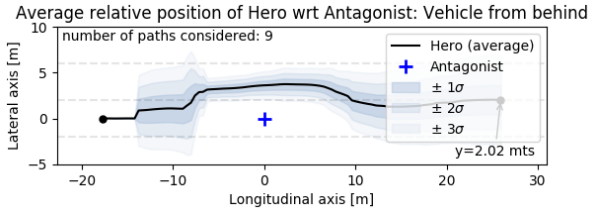
En la tabla 20 se observa el número de colisiones observada por casos y el total de éstas. Se observa que ambas configuraciones de *pre-trained* presentaron un mayor número de choques, siendo estos sobre 10. Después siguieron el agente entrenado por un tercero y el agente *pre-trained* con cámaras laterales, teniendo sobre 5 impactos. Finalmente los agentes que presentaron menos colisiones son ambos casos de *init zero*, con solo 2 choques. Cabe mencionar que estas colisiones suelen ser leves, producto de pequeñas variaciones en la dirección de los agentes, no impidiendo que se realice la maniobra completa. Por su parte, los humanos no presentaron ningún impacto. El mayor número de colisiones se observa en los casos más complejos.

Colisiones	Case Base	Vehículo desde atrás	Vehículo por delante	total
Old pre-trained	1	9	3	13
Pre-trained	1	3	7	11
Init zero other	1	5	1	7
Pre-trained side cams	1	0	4	5
Init zero	0	1	1	2
Init zero side cams	0	1	1	2
Persona 1	0	0	0	0
Persona 2	0	0	0	0
Persona 3	0	0	0	0
Persona 4	0	0	0	0

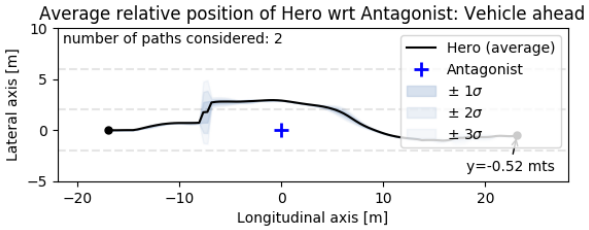
Tabla 20: Ranking de colisiones.



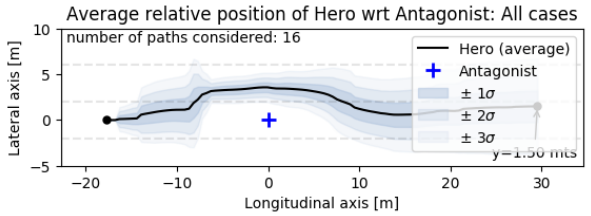
(a) *Pre-trained*, caso simple.



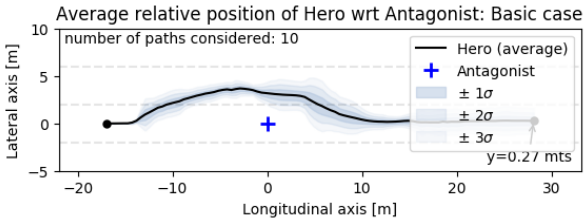
(b) *Pre-trained*, vehículo por detrás.



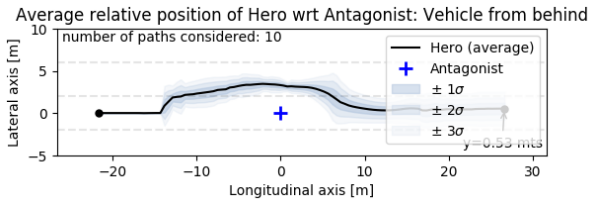
(c) *Pre-trained*, vehículo por delante.



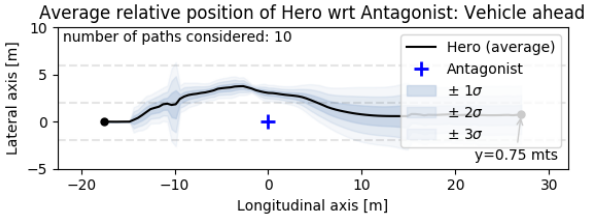
(d) *Pre-trained*, general.



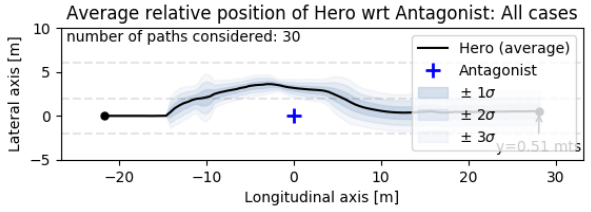
(e) *Init zero*, caso simple.



(f) *Init zero*, vehículo por detrás.

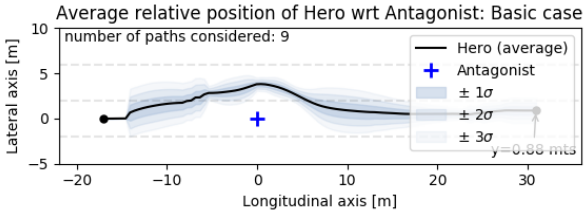


(g) *Init zero*, vehículo por delante.

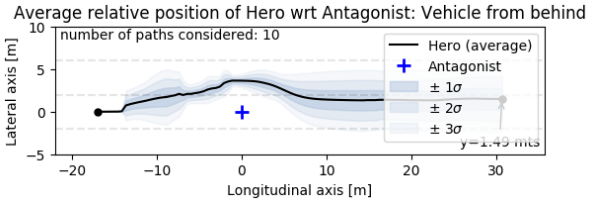


(h) *Init zero*, general.

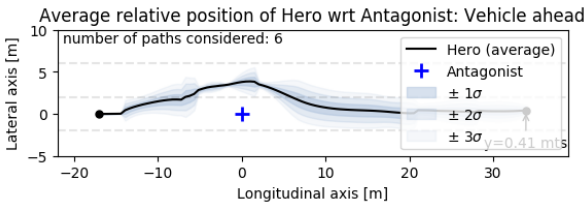
Figura 31: Trayectoria promedio de conducción agentes para caso simple, vehículo desde atrás y vehículo por adelante.



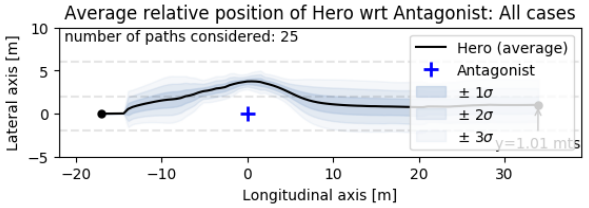
(i) *Pre-trained side cams*, caso simple.



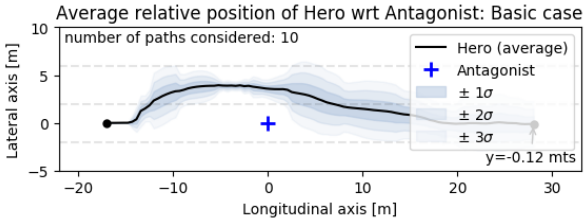
(j) *Pre-trained side cams*, vehículo por detras.



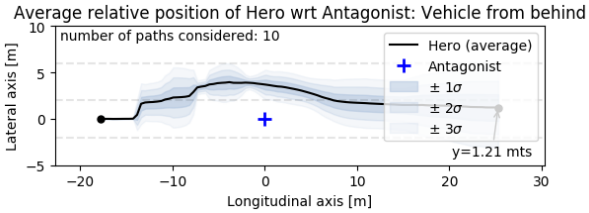
(k) *Pre-trained side cams*, vehículo por delante.



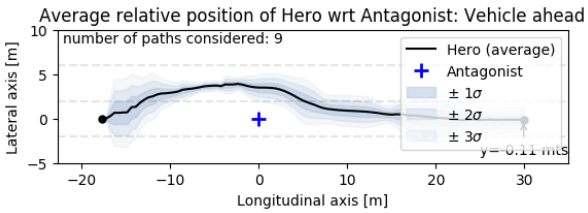
(l) *Pre-trained side cams*, general.



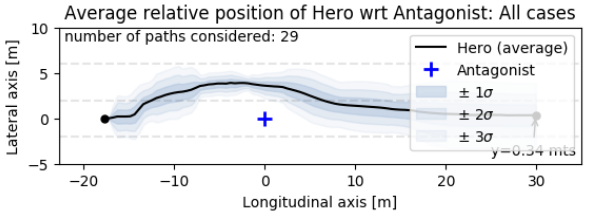
(m) *Init zero side cams*, caso simple.



(n) *Init zero side cams*, vehículo por detrás.

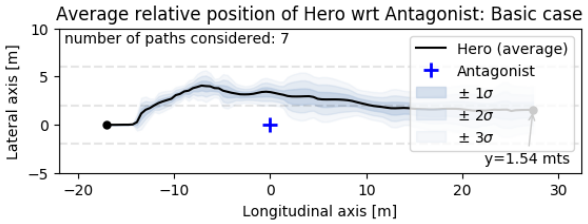


(ñ) *Init zero side cams*, vehículo por delante.

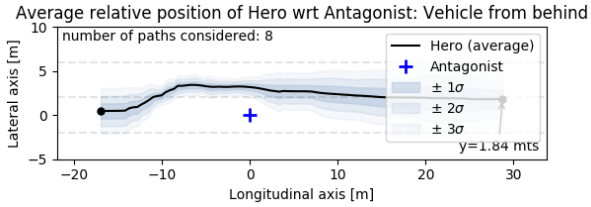


(o) *Init zero side cams*, general.

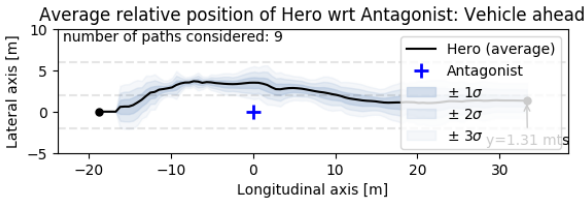
Figura 31: Trayectoria promedio de conducción agentes para caso simple, vehículo desde atrás y vehículo por adelante (cont.).



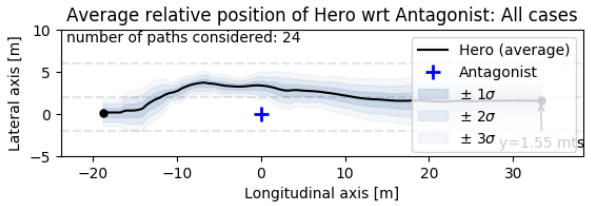
(a) *Old pre-trained*, caso simple.



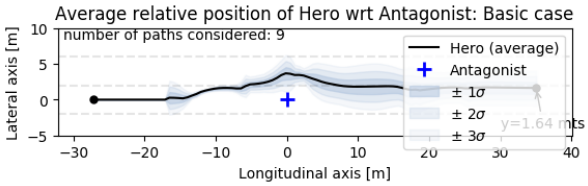
(b) *Old pre-trained*, vehículo por detrás.



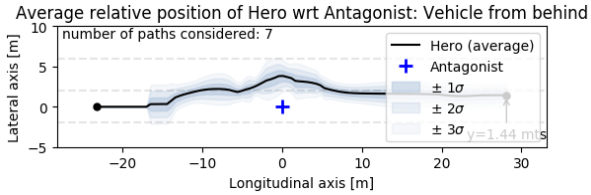
(c) *Old pre-trained*, vehículo por delante.



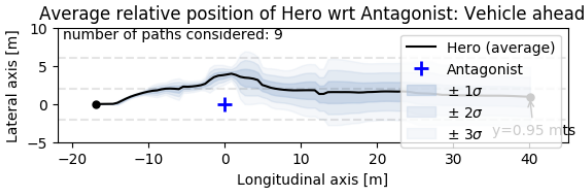
(d) *Old pre-trained*, general.



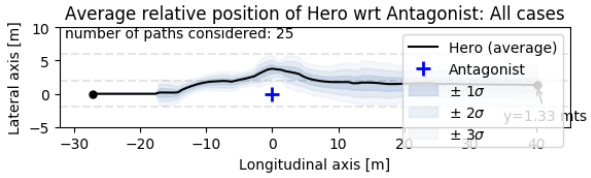
(e) *Init zero other*, caso simple.



(f) *Init zero other*, vehículo por detrás.

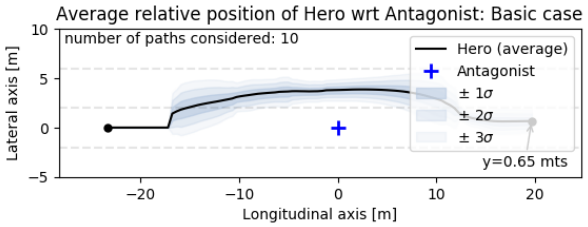


(g) *Init zero other*, vehículo por delante.

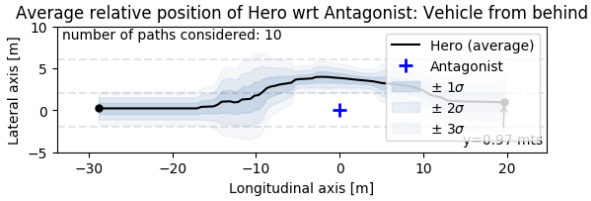


(h) *Init zero other*, general.

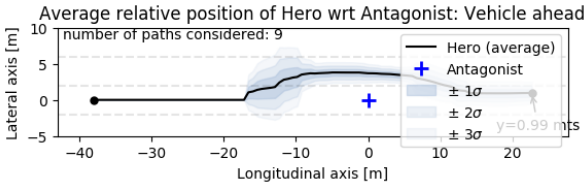
Figura 32: Trayectoria promedio de conducción agentes especiales (*old pre-trained* e *init zero other*) para caso simple, vehículo desde atrás y vehículo por adelante.



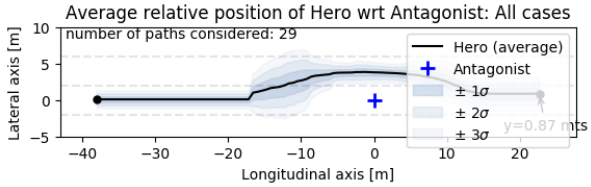
(a) Persona 1, caso simple.



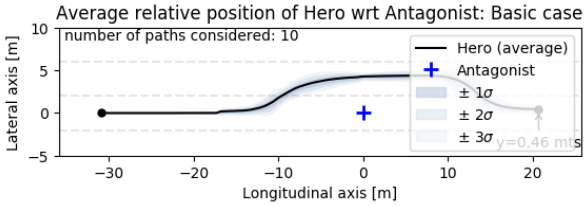
(b) Persona 1, vehículo por detrás.



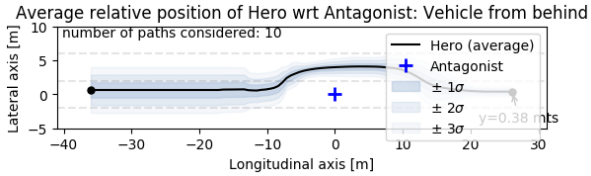
(c) Persona 1, vehículo por delante.



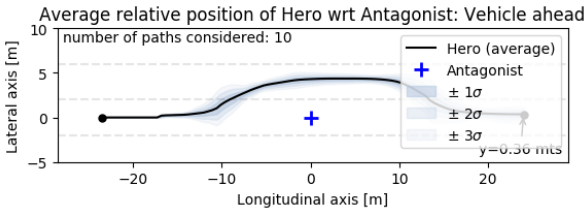
(d) Persona 1, general.



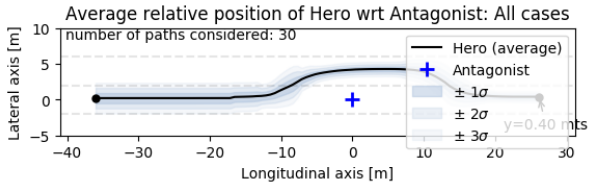
(e) Persona 2, caso simple.



(f) Persona 2, vehículo por detrás.

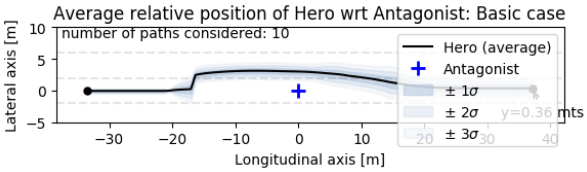


(g) Persona 2, vehículo por delante.

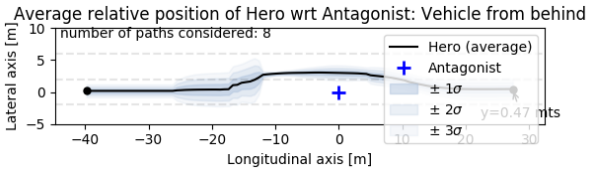


(h) Persona 2, general.

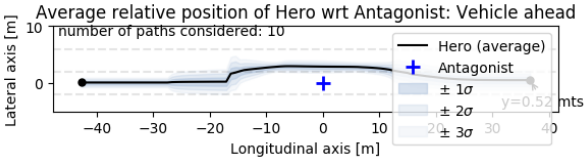
Figura 33: Trayectoria promedio de conducción humana para caso simple, vehículo desde atrás y vehículo por adelante.



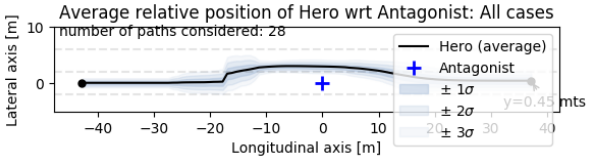
(i) Persona 3, caso simple.



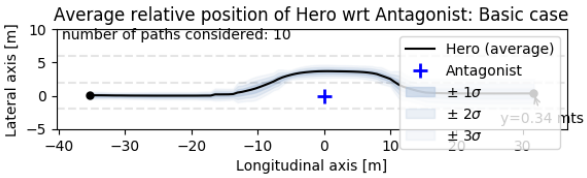
(j) Persona 3, vehículo por detras.



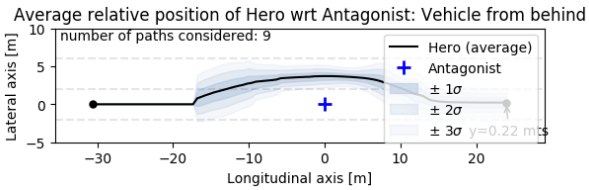
(k) Persona 3, vehículo por delante.



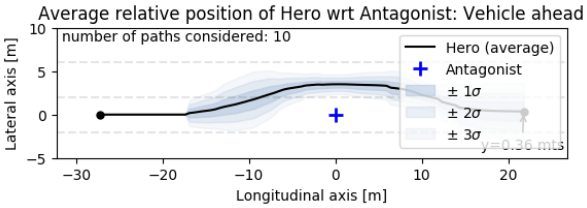
(l) Persona 3, general.



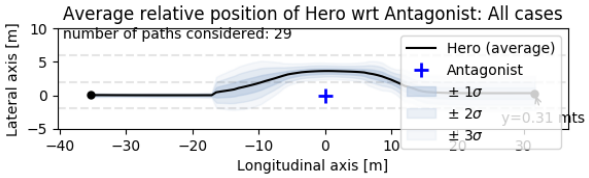
(m) Persona 4, caso simple.



(n) Persona 4, vehículo por detrás.



(ñ) Persona 4, vehículo por delante.



(o) Persona 4, general.

Figura 33: Trayectoria promedio de conducción humana para caso simple, vehículo desde atrás y vehículo por adelante (cont.).

4.2.3. Trayectoria Promedio

En las figuras 31, 32 y 33 se muestran las trayectorias promedios de los agentes y conductores humanos, similares a los vistos en la sección 4.1.1. En esta oportunidad están separados por caso, teniéndose un cuarto gráfico que considera todos los episodios. Como en este experimento se consideró un número fijo de episodios, en vez de un número fijo de observaciones válidas, estos gráficos fueron construidos en base a los episodios exitosos, variando el número de trayectorias consideradas. Se tiene que la figura 31 muestra los gráficos de los cuatro casos de agentes, mientras que la figura 32 muestra los casos especiales (*old pre-trained* e *init zero other*).

En este caso se observan resultados similares a los del experimento 1, aunque en esta oportunidad la trayectoria de los agentes muestra una forma más similar a la de un trapecio, similar a los humanos. Además la curvatura es más regular, aunque sigue sin ser tan regular como la de los humanos. También se observa que los tramos de más variación dependen del conductor y del caso, habiendo más diversidad. De todos modos las zonas de más variación sigue siendo en los cambios de pista. Del mismo modo se aprecia una mayor heterogeneidad en la forma de las trayectorias, que a diferencia del experimento 1, no existe un patrón definido para los humanos y otro para los agentes.

4.2.4. Trayectoria Promedio con respecto al Tiempo

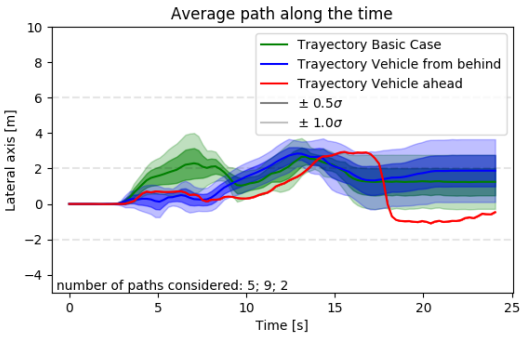
En este caso se volvió a hacer una interpolación de la trayectoria relativa de *Hero* con respecto a *Antagonist*, pero esta vez se hizo en la coordenada y sobre el tiempo. Nuevamente se tomo el promedio y desviación estándar con respecto a la coordenada y de los episodios considerados. En esta oportunidad se graficaron los 3 casos por separado, observando la progresión del vehículo en el eje y a lo largo del tiempo, en cada uno de estos casos.

En la figura 34 se observa la progresión de los agentes a lo largo del tiempo. Para los agentes *pre-trained*, *init zero* y sus versiones con cámaras laterales, estos suelen adelantar antes en el caso simple, luego en el caso del vehículo por atrás y finalmente en el caso del vehículo por delante. Para los casos de *pre-traned old* y *init zero other* no se observa un orden definido, lo que se debe a que existe una mayor variación en el tiempo en el que realizan el adelantamiento. Dejando de lado a los casos con pocos episodios exitosos, se observa que existe una mayor variación en el tiempo para cuando un vehículo intentando adelantar desde atrás.

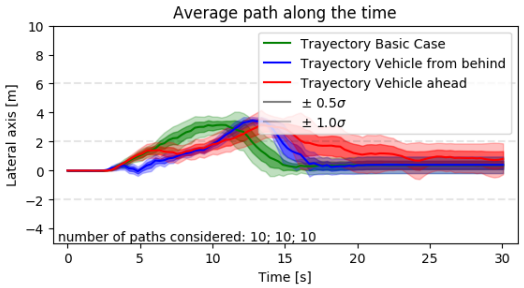
En la figura 35 se observa la progresión de los conductores humanos a lo largo del tiempo. En este caso se adelanta antes en el caso simple y con un vehículo por delante, adelantando después en el otro caso. Pese a que en el segundo caso se realiza la maniobra de adelantamiento después, este sigue siendo al mismo tiempo que los agentes. Se observa que para el caso del vehículo de frente, los humanos tienden a iniciar la maniobra antes.

4.2.5. Posición Media antes de Adelantar

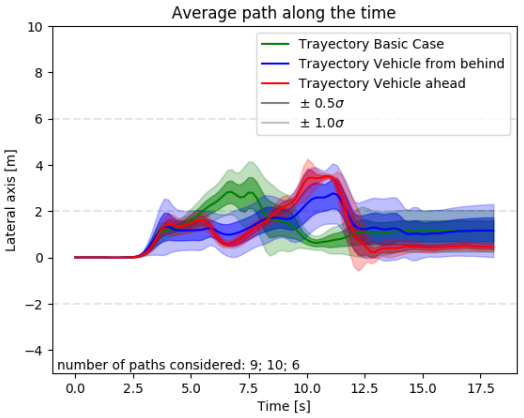
Considerando que para los casos con el vehículo adelantando por detrás y viajando en dirección contraria por la pista izquierdo, *Hero* tiene que pasa por un período en el que tiene que esperar detrás del



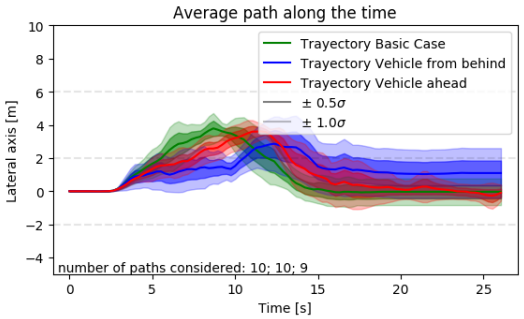
(a) *Pre-trained*



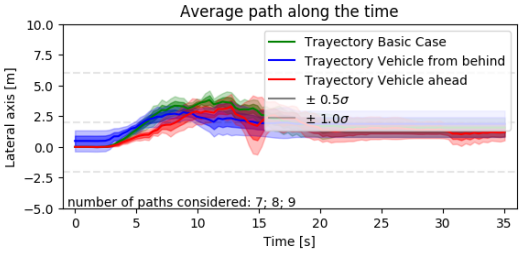
(b) *Init zero*



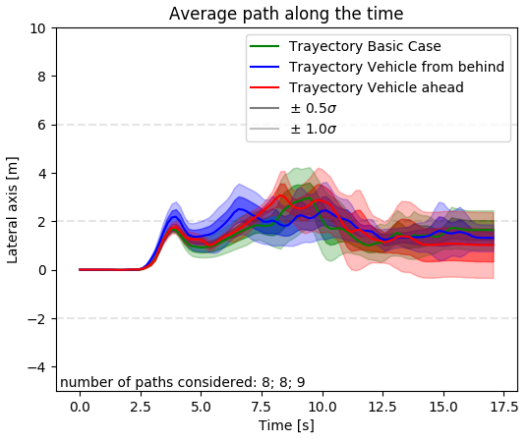
(c) *Pre-trained side cams*



(d) *Init zero side cams*

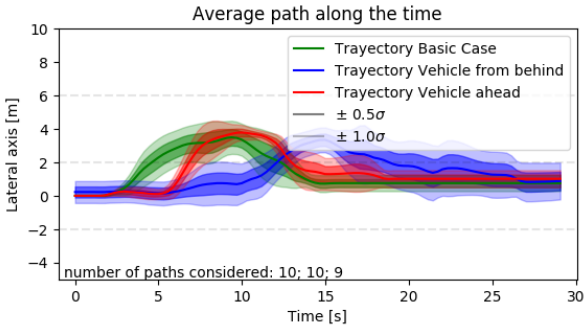


(e) *Old pre-trained*

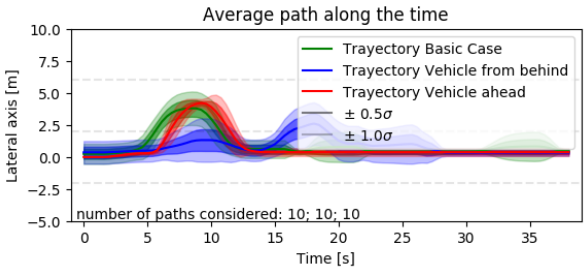


(f) *Init zero other*

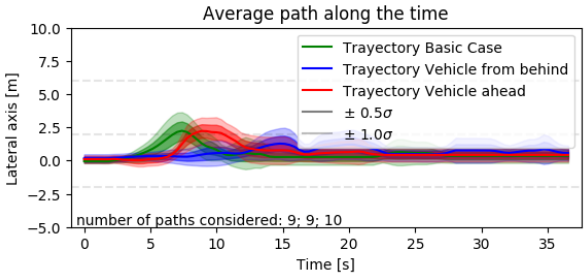
Figura 34: Trayectoria media de agentes con respecto al tiempo. En verde caso simple, en azul vehículo desde atrás y en verde vehículo de frente.



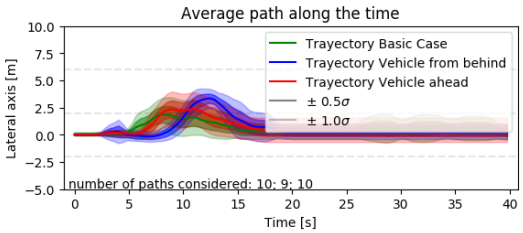
(a) Persona 1



(b) Persona 2



(c) Persona 3



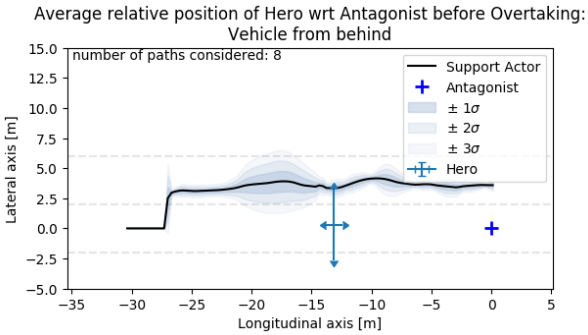
(d) Persona 4

Figura 35: Trayectoria media de conducción humana con respecto al tiempo. En verde caso simple, en azul vehículo desde atrás y en verde vehículo de frente.

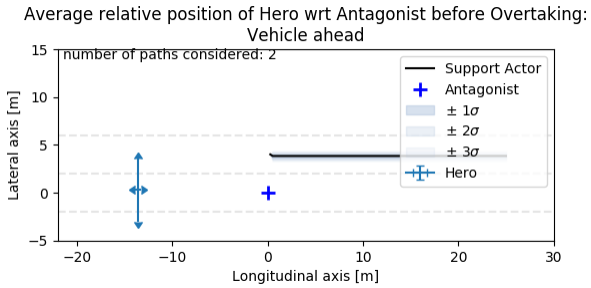
vehículo a adelantar. Para poder visualizar esto, se graficó la posición relativa media y desviación estándar de *Hero*. Para el caso del vehículo por detrás, solo se consideraron los casos en el que efectivamente se espera al vehículo, y no cuando se adelanto antes que este. También se graficó la trayectoria media de *Support Actor*, que toma el rol del tercer vehículo.

En la figura 36 se observan las posiciones medias de los agentes. A la izquierda están los gráficos para el caso del vehículo adelantando por detrás, y a la derecha los gráficos para cuando viene un vehículo en dirección contraria por la pista izquierda. Para el caso del vehículo por detrás, se observa que los agentes tienden a mantenerse dentro de la pista, a aproximadamente 15 [m] de distancia del otro vehículo. Los agentes *old pre-trained* y *init zero other* sin embargo, muestran una mayor variación en su posición lateral. Esto puede explicar el gran número de colisiones observadas. Lo mismo se observa para el caso del vehículo por delante, observándose la misma relación entre la variación en la posición lateral y el número de colisiones.

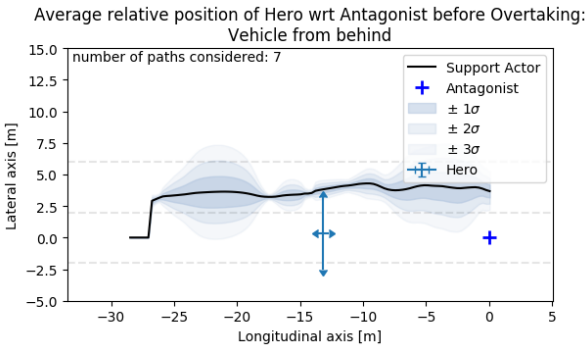
En la figura 37 se observan las posiciones medias de los conductores humanos. Del mismo modo, a la izquierda están los gráficos para el caso del vehículo adelantando por detrás, y a la derecha los gráficos para cuando viene un vehículo en dirección contraria por la pista izquierda. En este caso se observa una mayor variación a lo largo del eje lateral, pero en este caso no se registran colisiones. Esto se puede deber



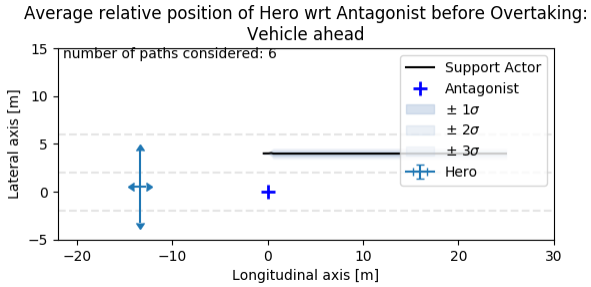
(a) *Pre-trained*, vehículo por detrás.



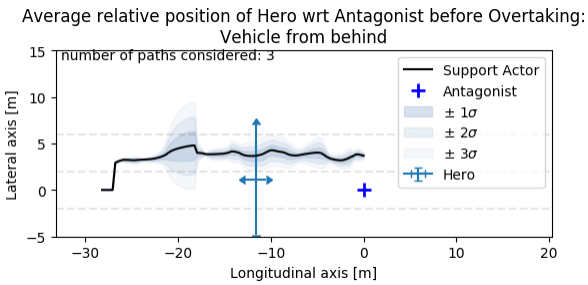
(b) *Pre-trained*, vehículo por delante.



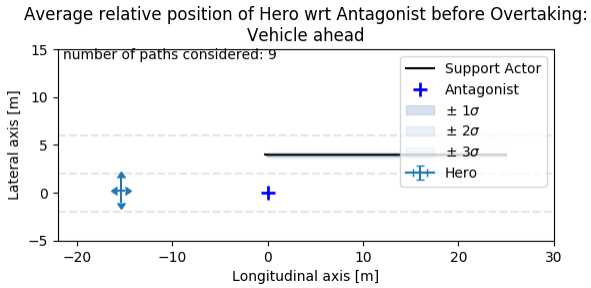
(c) *Pre-trained side cams*, vehículo por detrás.



(d) *Pre-trained side cams*, vehículo por delante.

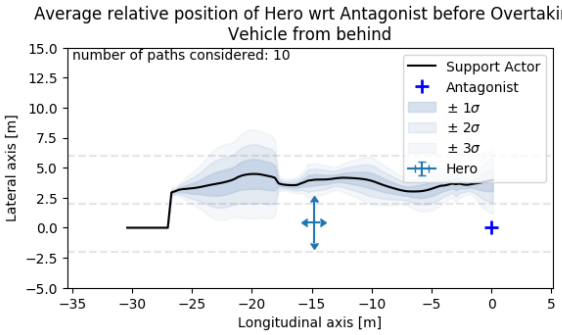


(e) *Old pre-trained*, vehículo por detrás.

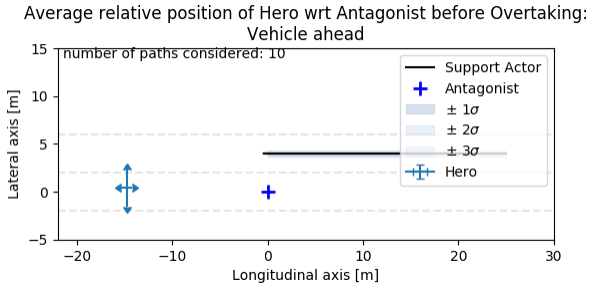


(f) *Old pre-trained*, vehículo por delante.

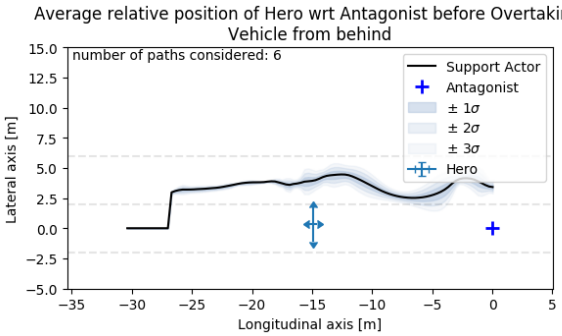
Figura 36: Posición media y desviación estándar de los agentes antes de adelantar. A la izquierda para el caso de vehículo desde atrás, a la derecha para el caso de vehículo por adelante.



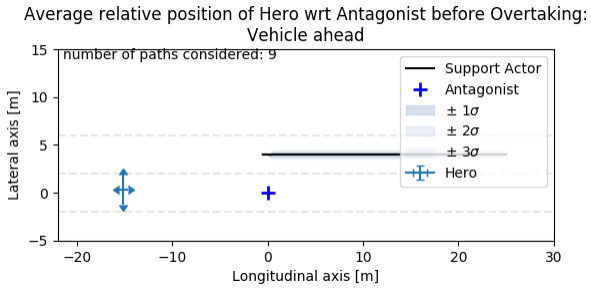
(g) *Init zero*, vehículo por detrás.



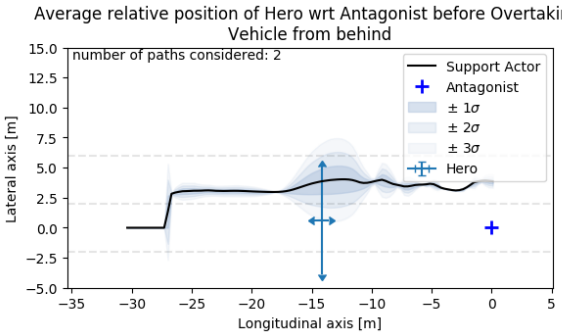
(h) *Init zero*, vehículo por delante.



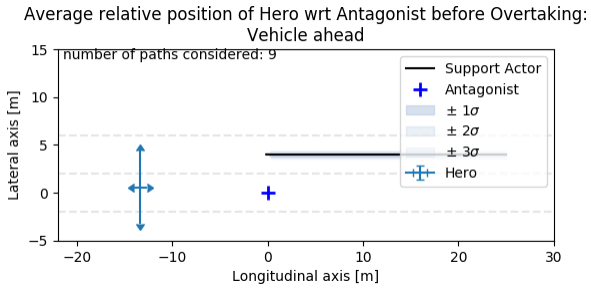
(i) *Init zero side cams*, vehículo por detrás.



(j) *Init zero side cams*, vehículo por delante.



(k) *Init zero other*, vehículo por detrás.



(l) *Init zero other*, vehículo por delante.

Figura 36: Posición media y desviación estándar de los agentes antes de adelantar. A la izquierda para el caso de vehículo desde atrás, a la derecha para el caso de vehículo por adelante (cont.).

a que comienzan a realizar la maniobra antes de que *Support Actor* pase por la posición $x = 0$, que es hasta donde se realizó la medición, o a que tienden a moverse a la izquierda, pero son mejores para evitar al otro vehículo.

4.2.6. Mediciones y Rankings

Los indicadores utilizados son los mismos a las realizadas en el experimento 1, aunque también se realizaron mediciones cuando en vehículo se encuentra esperando su turno para realizar la maniobra.

En la tabla 21 se observan los datos asociados a la velocidad de los vehículos. Se aprecia que en este caso los conductores humanos tienen más problemas para respetar la velocidad máxima, notando que ahora los agentes presentan mejores resultados. De todos modos se encuentran dentro de un rango estrecho, entre 37 a 45 km/h. Se cree que se tienen más problemas para respetar la velocidad debido a que al ser un mapa más estrecho, los conductores tienen más dificultad para estar pendientes de la velocidad. También los agentes presentan mejores resultados debido a que se entrenaron por más tiempo, teniéndose más tiempo para corregir los excesos de velocidad.

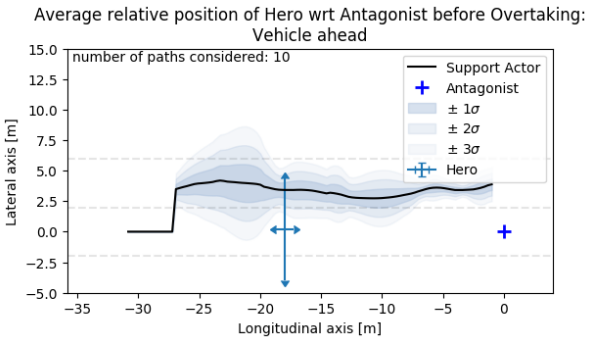
Conductor	Promedio [km/h]	STD [km/h] (σ)	Min [km/h]	Max [km/h]
Init zero	37.36	13.23	0	65.69
Pre-trained	38.9	16.53	0	77.25
Old pre-trained	41.32	16.1	0	87.02
Init zero side cams	41.36	14.55	0	70.86
Persona 1	41.43	17.34	0	83.69
Persona 4	42.71	18.3	0	100.84
Pre-trained side cams	43.12	19.35	0	92.48
Persona 2	44.27	17.27	0	81.22
Init zero other	44.43	20.61	0	101.81
Persona 3	45.61	27.22	0	138.79

Tabla 21: Ranking de velocidad [km/h].

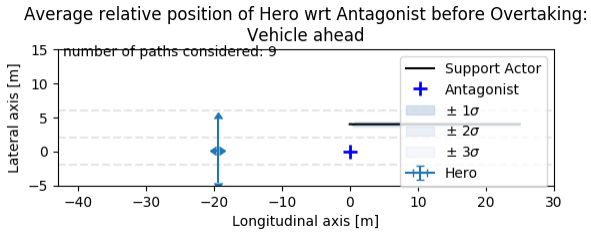
En las tablas 22 y 23 se observa la distancia relativa a la que se deja y retoma la pista. Nuevamente los agentes realizan esta maniobra antes que los conductores humanos, pero dentro de un margen de como lo haría un humano. Al ver los valores mínimos, se observa que existen casos en que los agentes se acercan demasiado a *Antagonist*, siendo los casos en los que ocurren las colisiones. En particular se observa *old pre-trained*, el cual presenta malos resultados.

En las tablas 24 y 25 se observa la distancia absoluta y relativa recorrida fuera de la pista, respectivamente. Nuevamente se presentan resultados similares, en donde para la distancia absoluta de los valores obtenidos para los agentes tienden a ubicarse en los extremos, y en la distancia relativa tienden a estar más cerca de *Antagonist*.

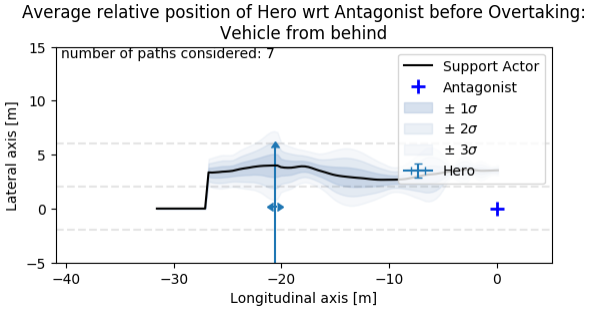
En la tabla 26 se observa el tiempo recorrido fuera de la pista. En este caso no se aprecian diferencias significativas entre humanos y agentes, observándose una distribución uniforme entre ambos tipos de conductores dentro del conjunto de muestras. De esto se aprecia que pese a que los agentes tienden a



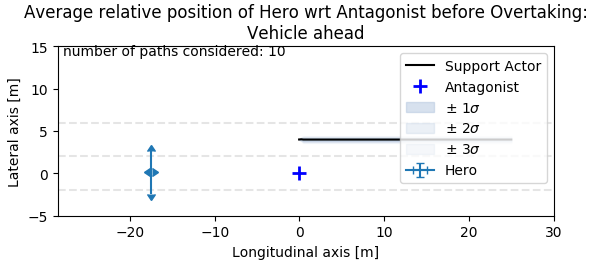
(a) Persona 1, vehículo por detrás.



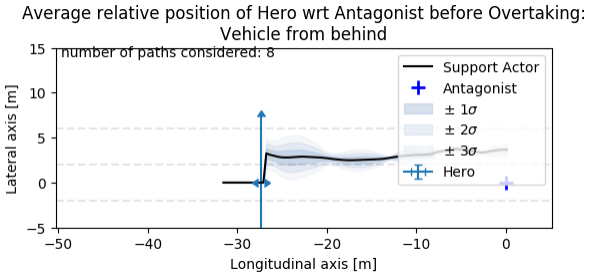
(b) Persona 1, vehículo por delante.



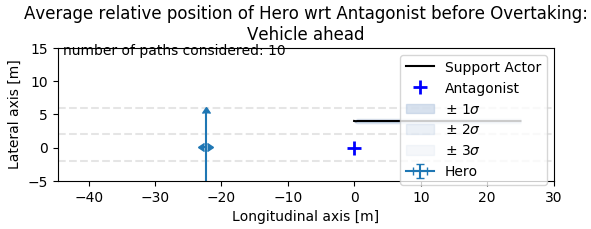
(c) Persona 2, vehículo por detrás.



(d) Persona 2, vehículo por delante.

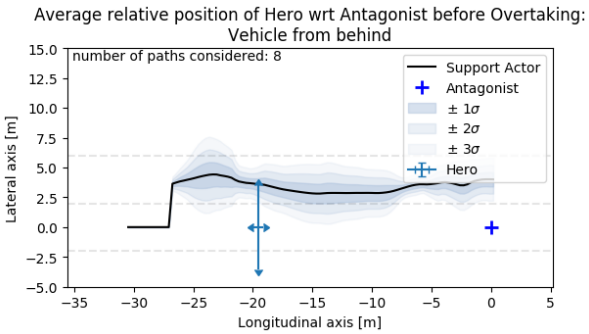


(e) Persona 3, vehículo por detrás.

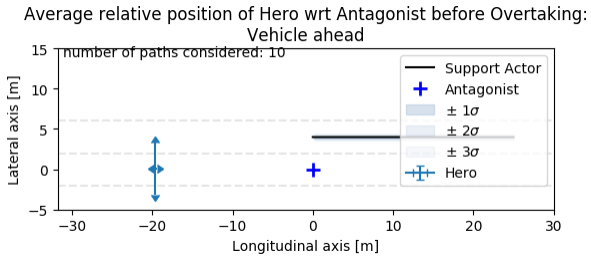


(f) Persona 3, vehículo por delante.

Figura 37: Posición media y desviación estándar de conductores humanos antes de adelantar. A la izquierda para el caso de vehículo desde atrás, a la derecha para el caso de vehículo por adelante.



(g) Persona 4, vehículo por detrás.



(h) Persona 4, vehículo por delante.

Figura 37: Posición media y desviación estándar de conductores humanos antes de adelantar. A la izquierda para el caso de vehículo desde atrás, a la derecha para el caso de vehículo por adelante (cont.).

Conductor	Promedio [m]	STD [m] (σ)	Min [m] (abs)	Max [m] (abs)
Init zero other	-7.61	3.49	-12.39	-0.48
Pre-trained side cams	-8.62	3.34	-2.83	-16.09
Persona 2	-8.92	1.27	-5.69	-10.93
Persona 4	-10.05	3.14	-5.27	-18.51
Init zero	-10.79	1.96	-6.8	-14.54
Pre-trained	-11.07	2.82	-8.16	-16.97
Init zero side cams	-12.12	2.2	-7.6	-17.17
Old pre-trained	-12.21	3.05	-0.09	-16.64
Persona 1	-13.91	3.16	-5.74	-18.96
Persona 3	-17.41	4.1	-10.73	-27.06

Tabla 22: Ranking distancia relativa longitudinal a la que dejan la pista [m].

Conductor	Promedio [m]	STD [m] (σ)	Min [m]	Max [m]
Init zero	5.71	1.37	3.82	9.46
Pre-trained side cams	6.39	2.16	3.94	13.12
Init zero side cams	6.45	3.22	2.36	17.94
Pre-trained	7.97	1.85	5.14	11.06
Old pre-trained	8.01	5.06	0	19.86
Init zero	9.49	6.43	2.07	30.33
Persona 1	10.31	2.14	4.6	13.11
Persona 4	10.43	2.16	6.02	15.36
Persona 3	12	3.95	3.9	19.83
Persona 2	13.7	1.46	9.47	17.15

Tabla 23: Ranking de distancia relativa longitudinal a la que se retoma la pista [m].

permanecer cerca de *Antagonist*, estos toman el mismo tiempo en realizar la maniobra.

Conductor	Promedio [m]	STD [m] (σ)	Min [m]	Max [m]
Pre-trained side cams	50.69	17.97	33.18	107.62
Init zero other	57.77	20.42	26.08	109.01
Pre-trained	59.27	9.05	46.23	80.32
Persona 4	60.84	13.13	30.28	91.48
Persona 3	62.19	10.76	40.55	103.33
Init zero	67.65	14.27	40.14	91.53
Persona 2	71.38	7.14	52.94	83.45
Init zero side cams	76.53	32.43	31.98	174.36
Old pre-trained	94.21	37.11	0.81	176.4
Persona 1	97.51	34.27	53.51	225.86

Tabla 24: Ranking distancia longitudinal recorrida fuera de la pista [m].

Conductor	Promedio [m]	STD [m] (σ)	Min [m]	Max [m]
Pre-trained side cams	15.11	3.64	10.44	23.35
Init zero	16.56	2.5	13.21	21.36
Init zero other	17.68	7.13	7.72	41.17
Init zero side cams	18.47	4.13	12.86	30.21
Pre-trained	18.61	4.05	15.36	28.03
Old pre-trained	20.13	7.5	0.09	33.47
Persona 4	20.48	3.87	12.15	28.29
Persona 2	22.62	1.99	17.48	25.81
Persona 1	24.26	3.95	10.34	30.64
Persona 3	29.41	5.94	19.04	46.89

Tabla 25: Ranking distancia longitudinal relativa recorrida fuera de la pista [m].

Conductor	Promedio [s]	STD [s] (σ)	Min [s]	Max [s]
Persona 3	3.05	0.73	1.55	5.15
Pre-trained side cams	3.43	1.48	1.8	8.35
Init zero other	3.81	1.35	1.8	6.2
Persona 4	4.09	1.15	1.5	6.75
Pre-trained	4.32	0.67	3.3	5.7
Persona 2	4.67	0.59	3.35	5.65
Init zero	5.57	1.32	2.95	8.55
Init zero side cams	5.89	1.93	3.8	13.3
Persona 1	6.83	2.1	3.9	11.2
Old pre-trained	7.35	3.47	0.05	15.4

Tabla 26: Ranking tiempo recorrido fuera de la pista [s].

En la tabla 27 se observa la distancia mínima entre *Hero* y *Antagonist*. Como era de esperarse, los agentes tienden a posicionarse más cerca que los conductores humanos. También se observan distancias correspondientes a colisiones, pero no en todos los agentes. En estos casos probablemente los agentes chocaron con *Support Actor* en vez de *Antagonist*.

Conductor	Promedio	STD (σ)	Min	Max
Persona 3	2.97	0.39	2.34	3.69
Init zero	3.07	0.27	2.38	3.48
Old pre-trained	3.16	0.49	1.94	3.96
Init zero other	3.27	0.51	2.04	4.01
Init zero side cams	3.45	0.46	2.38	4.3
Pre-trained	3.49	0.5	2.88	4.41
Pre-trained side cams	3.53	0.32	2.75	4.11
Persona 4	3.61	0.41	2.58	4.33
Persona 1	3.76	0.37	2.97	4.52
Persona 2	4.17	0.33	3.06	4.68

Tabla 27: Ranking distancia más cercana a *Antagonist* [m].

En las tablas 28 y 29 se muestran los valores máximos y mínimos de la coordenada y, representando la desviación máxima hacia la izquierda y la derecha respectivamente. En esta oportunidad los agentes presentan una mayor desviación hacia la izquierda que los conductores humanos, mientras que no se observa una tendencia por desviarse hacia la derecha.

Conductor	Promedio [m]	STD [m] (σ)	Min [m]	Max [m]
Persona 3	3.18	0.38	2.52	4.1
Pre-trained	3.74	0.61	2.7	4.6
Persona 4	3.77	0.36	2.83	4.41
Init zero	3.78	0.31	2.96	4.68
Pre-trained side cams	3.95	0.31	3.51	5.08
Persona 1	4.1	0.53	3.04	5.46
Old pre-trained	4.15	0.47	3.31	5.06
Init zero other	4.15	0.66	2.5	5.69
Init zero side cams	4.26	0.58	3.74	5.94
Persona 2	4.35	0.31	3.2	4.83

Tabla 28: Ranking distancia longitudinal máxima (máxima desviación hacia la izquierda) [m].

Dentro de los resultados obtenidos se ve nuevamente que los agentes tienden a comportarse como los conductores humanos, con la excepción de que viajan más cerca de *Antagonist* y se aprecia que la curvatura de la trayectoria que describen es más irregular. En el caso de *old pre-trained*, en donde se tenía la cámara a una mayor altura, se observaron peores resultados que con la nueva configuración. Una posible explicación a esto puede ser que al estar más elevado, los elementos en la imagen se ven más pequeños, por lo que le cuesta más a la red reconocer los diferentes posibles estados. Esto también se apreció al utilizar D-COACH, en donde se tuvo una mayor dificultad para que realizara la acción correcta, lo cual se refleja en los resultados. Un ejemplo de esto fue el caso en que se encontraba detrás de otro vehículo, en donde se dificultó enseñarle cuando debía esperar al vehículo detrás y cuando debía proceder con la maniobra.

Conductor	Promedio [m]	STD [m] (σ)	Min [m]	Max [m]
Persona 2	0.01	0.18	-0.31	0.67
Old pre-trained	-0.01	1	-4	2.3
Init zero other	-0.08	0.18	-0.79	-0.01
Persona 1	-0.14	0.18	-0.52	0.01
Pre-trained side cams	-0.15	0.29	-1.25	0
Persona 3	-0.16	0.41	-1.05	0.94
Init zero	-0.31	0.49	-2.41	-0.01
Persona 4	-0.32	0.23	-0.75	0.22
Init zero side cams	-0.41	0.47	-1.94	0
Pre-trained	-0.53	0.45	-1.1	-0.01

Tabla 29: Ranking distancia longitudinal mínima (máxima desviación hacia la derecha) [m].

Con respecto al agente entrenado por un tercero *init zero other*, también se obtuvieron peores resultados. Esto se puede deber a que al ser una tareas más complejas, se debe tener cierta práctica y estar consciente de como se comporta la red. En este caso se buscó probar D-COACH con alguien sin experiencia previa, para poder ver cuanto podía aprender la red bajo estas condiciones.

Otro punto a considerar el la diferencia de información entre la persona y el agente. Por un lado el agente solo percibe el tiempo presente, mientras que el humano es capaz de retener información del pasado. Un ejemplo de esto es cuando el humano entrega *feedback* buscando seguir a *Antagonist* cuando este dobla en una intersección. Al no verse en las cámaras, el agente entiende que cuando esta solo frente a una intersección, este debe doblar.

En el caso de los 4 agentes principales, siendo estos *pre-trained*, *init zero*, *pre-trained* con cámaras laterales e *init zero* con cámaras laterales, los que mostraron un mejor desempeño fueron los casos de *init zero*. En particular el que mostró mejores resultados fue *init zero* sin las cámaras laterales, sin embargo esta cuenta con un punto ciego mientras realiza el adelanto.

4.3. Experimento 3: Entrenamiento Extendido

Para los agentes con mejores resultados del segundo escenario experimental, se les siguió entrenando hasta que uno de estos realizara todos los episodios de prueba de forma exitosa y sin registrar colisiones. Estos agentes corresponden a *init zero* e *init zero* con cámaras laterales. Esto se entrenaron otros 30 episodios y se realizó un proceso de ajuste que duro aproximadamente 2 horas por agente. El caso *init zero* con cámaras laterales es el que presentó mejores resultados, consiguiendo realizar todos los episodios sin chocar. *Init zero* mostró un empeoramiento de su conducta, observándose un aumento en el número de colisiones y una disminución de episodios exitosos.

4.3.1. Colisiones

En la tabla 30 se observan las colisiones obtenidas por los agentes luego de volver a ser entrenados. También se muestra en que número aumentaron o disminuyeron, con respecto al caso anterior. Se puede ver que *init zero* presenta peores resultados, incrementando en 5 el número de choques totales. Por otro

lado, para *init zero* con cámaras laterales, se observaban 1 colisión para el caso simple, 1 colisión cuando había un vehículo detrás y 2 colisiones cuando había un vehículo delante. Luego de volver a ser entrenado, estos números bajaron a 0.

Conductor	Caso simple	Vehículo detrás	Vehículo delante	Total
Init zero	0	4 (+3)	3 (+2)	7 (+5)
Init zero side cams	0	0 (-1)	0 (-1)	0 (-2)

Tabla 30: Colisiones de mejores agentes luego de la segunda ronda de entrenamientos.

4.3.2. Resultados de Episodios

En las tablas 31, 32 y 33 se observan los resultados obtenidos de cada caso, mostrando en cuando aumentaron o disminuyeron, con respecto a los resultados obtenidos anteriormente, antes de volver a ser entrenados. Se observa que *init zero* disminuyó el número de episodios exitosos en cada uno de los casos. De cumplir de forma exitosa los 30 episodios, este bajo a un total de 23 episodios. Para *init zero* con cámaras laterales, se observa que este es el agente que cumple con todos los episodios de forma exitosa.

Conductor	Éxito	Fracaso
Init zero	7 (-3)	3 (+4)
Init zero side cams	10	0

Tabla 31: Resultados caso simple, luego de segunda ronda de entrenamientos.

Conductor	Adelanta después	Adelanta antes	Fracaso
Init zero	7 (-3)	0	3 (+3)
Init zero side cams	10 (+4)	0 (-2)	0 (-2)

Tabla 32: Resultados caso con vehículo adelantando desde atrás, luego de segunda ronda de entrenamientos.

Conductor	Éxito	Fracaso
Init zero	9 (-1)	1 (+1)
Init zero side cams	10 (+1)	0 (+1)

Tabla 33: Resultados caso con vehículo conduciendo por la pista izquierda en dirección contraria, luego de segunda ronda de entrenamientos.

4.3.3. Mediciones

En las tablas 34 y 35 se observan las mediciones realizadas a los agentes luego de volver a ser entrenados. Al compararlos con los valores obtenidos en el experimento 2, estos no varían significativamente.

Pese a que para el experimento 2 *init zero* presentó mejores resultados con respecto a *init zero* con cámaras laterales, al seguir siendo entrenado este incluso terminó con un peor desempeño. Esto puede significar que los resultados observados inicialmente no fueron una consecuencia directa del *feedback* entregado por el humano, sino que fueron debido a una conducta aprendida de forma no intencional, que finalmente fue eliminada al seguir con el entrenamiento. El experimento 3 originalmente tenía como objetivo mejorar el desempeño de los mejores agentes, sin embargo también sirvió para eliminar aleatoriedades presentes en sus comportamientos.

Init zero	Promedio	STD (σ)	Min	Max
Velocidad [km/h]	41.67	13.88	0	72.62
Pos. relat. x se deja pista [m]	-10.92	2.61	-15.39	-4.91
Pos. relat. x se retoma pista [m]	6.16	2.73	0.61	14.63
Distancia fuera de la pista [m]	84.08	20.45	51.84	117.92
Distancia relat. fuera de la pista [m]	17.12	3.32	11.11	27.66
Tiempo fuera de la pista [s]	6.31	1.86	3.45	10.6
Distancia mínima a *Antagonist* [m]	2.91	0.36	2.01	3.82
Posición lateral máxima [m]	3.74	0.55	3.04	5.8
Posición lateral mínima [m]	-0.71	1.11	-1.93	-0.01

Tabla 34: Mediciones realizadas a *init zero*, luego de la segunda ronda de entrenamientos.

Init zero side cams	Promedio	STD (σ)	Min	Max
Velocidad [km/h]	42.29	15.72	0	77.1
Pos. relat. x se deja pista [m]	-11.13	2.4	-15.14	-4.96
Pos. relat. x se retoma pista [m]	6.24	2.23	3.85	11.57
Distancia fuera de la pista [m]	65.28	13.89	30.76	86.3
Distancia r. fuera de la pista [m]	17.58	2.82	12.41	25.11
Tiempo fuera de la pista [s]	4.55	1.12	2.15	6.2
Distancia minima a *Antagonist* [m]	3.94	0.23	3.2	4.29
Posición lateral máxima [m]	4.68	1.05	3.72	8.51
Posición lateral mínima [m]	-0.33	0.36	-1.61	0.15

Tabla 35: Mediciones realizadas a *init zero* con cámaras laterales, luego de la segunda ronda de entrenamientos.

Los resultados de *init zero* se pueden deber a que cuenta con puntos ciegos, existiendo casos que *init zero* es incapaz de diferenciar. Para esto, simplemente se tiene que esperar a que el agente se comporte de una forma conveniente para ambos casos, no habiendo sido necesariamente entrenado con esto en consideración (y no debería). Un ejemplo de esto es cuando existe un vehículo detrás intentando adelantar. Cuando este se encuentra al lado de *Hero*, *init zero* es incapaz de verlo, por lo que no puede diferenciar entre esto y el caso en que *Hero* se encuentra solo con *Antagonist*, en donde debería comenzar a realizar la maniobra. Para que el episodio sea exitoso, *init zero* debería iniciar el adelantamiento lo suficientemente lento como para dejar pasar al otro vehículo, aunque el profesor podría no ser consciente de esto. Otro ejemplo de esto es cuando se está junto a *Antagonist*, como se puede ver en la figura 38, no pudiendo

saber si lo esta adelantando o si esta solo en la pista. Más aun, en estos casos podría no existir o no encontrarse una conducta que se ajuste a ambos, teniéndose que al entrenar uno, se esta empeorando el desempeño en otro. Inclusive se podría dar el escenario en el que al intentar obtener un buen desempeño en ambos casos, ambos terminen empeorando. Esto explicaría los resultados de *init zero*, en donde al seguir siendo entrenado se puede haber perdido una de estas conductas convenientes, pero aprendida de forma inconsciente. Por su parte, *init zero* con cámaras laterales no cuenta con estos puntos ciegos, por lo que puede diferenciar cada caso y actuar acorde a las circunstancias, explicando su mejora con más entrenamientos y obteniendo mejores resultados que *init zero*.

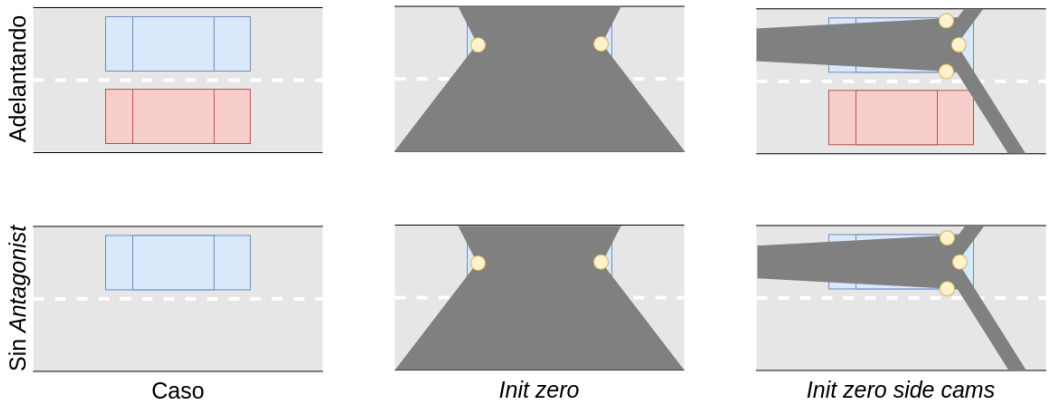


Figura 38: Diferencias en la percepción entre *init zero* e *init zero* con cámaras laterales. La fila superior corresponde al caso en el que *Hero*se encuentra junto con *Antagonist*, mientras que la fila inferior corresponde al caso sin *Antagonist*. La columna izquierda muestra los casos, la columna central muestra la percepción de *init zero* y la columna derecha muestra la percepción de *init zero* con cámaras laterales. Se observa que *init zero* con cámaras laterales es capaz de diferencias ambos casos, mientras que *init zero* no.

4.4. Respuesta Oscilatoria y Post Procesamiento

Pese a que en principio, D-COACH es capaz de estimar la magnitud de las respuestas, se observó que esta puede tender a tener comportamientos oscilatorios. En este trabajo de memoria, esto puede ocurrir sobre la señal de *steer*, el cual frente a pequeñas variaciones o a gran velocidad, se puede producir grandes cambios en la orientación. Estos cambios se traduce en un cambio considerable de lo que observan los sensores y en consecuencia del estado. El comportamiento oscilatorio ocurre cuando se desea ir en línea recta y la corrección inicial es muy grande, se realiza de manera reiterada o se viaja a altas velocidades. Esto afecta el comportamiento dinámico del vehículo y del simulador. Si el vehículo se desvía hacia la izquierda, naturalmente se corregiría para que gire a la derecha, pero si ocurre uno de los casos mencionados anteriormente, se puede terminar con el vehículo desviado hacia la derecha, en vez de apuntar hacia el frente, luego se vuelve a corregir hacia la izquierda, resultando en la misma posición inicial. entrando en un bucle. Esto se produciría antes de realizar la corrección de la magnitud. Si este ciclo resulta ser demasiado rápido, es difícil para el usuario regular la magnitud, considerando que se tendría que dar las señales exactas cuando los valores de *steer* son positivos y negativos. Para solucionar esto se podría hacer que el simulador fuera más lento, sin embargo esto podría resultar en el aprendizaje de una conducta que no es *human-like*. Entregar *feedbacks* equivocados resultaría en un incremento en la magnitud de la oscilación,

lo cual, como el rango de velocidad no fue limitada durante el entrenamiento, hacia que el vehículo se hiciera especialmente inestable a grandes velocidades. En la práctica, cuando este comportamiento aparece, intentar corregir esto resulta en una amplificación de la onda.

Frente a esto, en un principio se consideró realizar un post-procesamiento sobre la salida de la red. Las opciones estudiadas fueron añadir filtros de frecuencia, aplicar un umbral y usar una función de transformación. Para el filtro de frecuencia y el umbral, se observó que como existen diferencias entre las acciones que entrega la red y las acciones que ve el usuario, esto puede resultar en una corrección errada, considerando que el *feedback* se realiza sobre el espacio de respuesta de la red. Para el caso del filtro de frecuencia, también se observa que como añade un retardo en la respuesta, esto misma produce efectos oscilatorios en la respuesta de la red, proporcional al *delay* generado. Por otro lado, como el umbral puede ocultar comportamientos oscilatorios pequeños, se puede producir que durante el entrenamiento esta conducta se amplifique, haciéndose visible solo cuando la amplitud es mayor al umbral, que como se indicó anteriormente, pueden resultar difíciles de corregir. De esto se observa agregar comportamientos no lineales en D-COACH no da buenos resultados.

Para el caso de la función de transformación, no se tiene el problema de las diferencias entre las respuestas de la red y el agente, pudiéndose usar la función inversa para hacer la conversión. Se consideró utilizar una función logit centrada en 0, con un rango entre $[-1, 1]$. Logit se define como:

$$logit(x) = Alog(\frac{x}{1-x}) + B \tag{11}$$

Siendo A y B variables por medio de las cuales se puede controlar el rango de la función. Con esto se buscaba atenuar la variación de *steer* para valores bajos, preservando el rango de respuesta y mejorando la sensibilidad. Sin embargo, debido a que se debía hacer la transformación inversa, se eliminaba la linealidad de la corrección, lo cual no es recomendable para el entrenamiento de redes neuronales.

Finalmente no se aplicó ninguno de los pre-procesamientos mencionados, solucionando el problema de las oscilaciones escogiendo un valor para la magnitud del error de *feedback* e bajo. Esto fue en torno a 0.5 para las señales de *throttle* y *brake*, y entre 0.1 y 0.01 para la señal de *steer*.

Una situación similar ocurre para estados que ocurren de manera rápida. En estos casos el tiempo de reacción es demasiado corto, dentro del cual uno debe alcanzar a evaluar la acción y entregar *feedback* si no es apropiado. En la práctica, esto requiere que el usuario reconozca dichos momentos y se prepare para dar el *feedback* deseado cuando esto suceda. Esto además conlleva a que se requiere repetir el mismo escenario varias veces, no pudiéndose dar múltiples *feedbacks* en un periodo tan reducido de tiempo. Lo mismo puede ocurrir con comportamientos no deseados del agente de corta duración, lo cuales se puede deber a un mal entrenamiento. Considerando que en problemas complejos los entrenamientos pueden llevar un tiempo considerable, empezar de nuevo puede no ser una buena opción. Incluso puede que este comportamiento tiende a aparecer en el entrenamiento por características propias del sistema.

Una solución podría ser guardar todos los estados y acciones, e identificar dichos momentos que ocurren demasiado rápido. Luego, de manera más controlada, se podría entregar los *feedbacks* correspondientes para eliminar estas conductas.

4.5. Modelo RGB-*Depth*

El principal problema encontrado en este modelo fue su gran costo computacional, generando una simulación lenta. Este problema se puede deber a que al momento de diseñar el modelo 1, este se hizo únicamente buscando el mejor AE, sin considerar el rendimiento en términos computacionales, ni ver que tan capaz era de controlar a *Hero* o responder a D-COACH. Un ejemplo de esto es que se escogió un AE por canal, en vez de un único AE, el cual, pese a que entregó mejores resultados como AE, resultó demasiado pesado como para ser usado con D-COACH. Para trabajos futuros, si se desea volver a diseñar el modelo 1, se recomienda partir con un modelo ligero, posiblemente con el menor número de canales y resolución pequeña. Posterior a esto, evaluar el modelo en base a su desempeño con D-COACH y como controlador de *Hero*, en vez de que tan bien funciona como AE, y en base a esto agregar más canales, aumentar la resolución, etc.

4.6. Analisis de sensibilidad

Cabe mencionar que dentro de todas las redes utilizadas, no se alcanzó a realizar un análisis de sensibilidad, quedando propuesto como trabajo futuro a realizar.

5. Conclusiones

Se logró entrenar un conjunto de agentes, los cuales fueron capaces de realizar la maniobra de adelanto en diferentes escenarios, con mayor o menor éxito. En particular, el agente *init zero* con cámaras laterales fue capaz de realizar todos los casos con los que fue evaluado. Estos episodios correspondieron a adelantar al vehículo sin ningún tipo de obstáculo, adelantar a un vehículo mientras otro intenta realizar la misma maniobra desde atrás, y adelantar a un vehículo mientras otro viene en la pista izquierda y en dirección contraria. Dentro de los logros esperados fue realizar todas estas estas maniobras sin chocar, lo cual se fue conseguido.

Se logró obtener mediciones de personas conduciendo, realizando los mismos escenarios experimentales que los agentes, con los cuales se pudo contrastar sus comportamiento. De acuerdo a las comparaciones realizadas, se aprecia que los agentes tienen un comportamiento *human-like*, siendo similar al mostrado por los humanos, logrando la segunda parte del objetivo general. Las mayores diferencias observables son las trayectorias descritas por los conductores y que los agentes tienden a mantenerse más cerca del vehículo adelantado. Con respecto a las trayectorias, la de los humanos tiende a asemejarse a la forma de trapecio. En el caso de los agentes, para el primer experimento, estas tienen una forma triangular, mientras que en segundo experimento, estas son más parecidas a las humanas. Por otro lado, los agentes tienden a describir trayectorias más irregulares, correspondientes a pequeños oscilaciones realizados mientras se mueven en línea recta.

Para poder realizar las comparaciones, se tomaron las trayectorias relativas de *Hero* con respecto a *Antagonist*, se interpolaron en el dominio del espacio y se promediaron. Este gráfico parece mostrar el comportamiento general de los conductores, con lo cual se pudo realizar las comparaciones y determinar el *human-likeness* de los agentes. Para el segundo escenario experimental, además del gráfico anterior, se realizaron 2 gráficos extra. El primero gráfico también corresponde a un promedio de las trayectorias, aunque en este caso se extrapoló con respecto al tiempo y separaró de acuerdo a los casos estudiados. Por medio de este gráfico se consiguió observar en que momento iniciaban y finalizaban la maniobra, de acuerdo a cada caso. El segundo gráfico muestra el promedio y desviación estándar de la posición del vehículo antes de realizar la maniobra, para los casos en donde había un vehículo de frente y detrás, considerando que en estos casos antes se tenía que esperar. Además muestra la trayectoria promedio de *Support Actor* durante este intervalo. Nuevamente, este gráfico sirvió para observar el comportamiento de los conductores y ser comparados entre ellos. Además de estos gráficos, se tomaron una serie de estadísticas, las cuales también sirvieron.

Se logro adaptar el código de D-COACH para ser usado con CARLA, junto con implementar los módulos requeridos para realizar los experimentos.

Se pudo determinar el tipo de sensores a utilizar, junto con su configuración dentro del vehículo. Para el sistema final implementado, se usaron cámaras de segmentación semántica y sensores que determinan el estado cinemático del vehículo. La mejor configuración encontrada para las cámaras corresponden a la configuración del agente *init zero* con cámaras laterales, el cual cuenta con una cámara frontal y dos cámaras laterales, que emulan los espejos retrovisores laterales de un vehículo. Esta configuración no tiene puntos ciegos considerables, a diferencia de la otra configuración considerada, que contaba con una cámara frontal y una cámara trasera. Esto lo que lo llevó a completar todos los episodios de forma exitosa.

La resolución de la cámara frontal fue de 800x600, mientras que la resolución de las cámaras laterales fueron de 100x75. Dentro del sistema, las observaciones de ambas cámaras laterales se consideran como una sola imagen, esto para no cambiar el sistema cuando solo se tiene una cámara trasera. Ambas cámaras se encuentran a 1.5 [m] de altura, lo cual se observó dio mejores resultados que a 2.4 [m] de altura, configuración utilizada inicialmente.

Se consiguió que el vehículo fuera capaz de conducir dentro de la pista por cuenta propia. Para esto, se implementó un sistema compuesto por un *multiplex* y 2 agentes, a partir de los cuales se obtiene la acción de *Hero*. *Multiplex* es el encargado de escoger cual acción de los 2 agentes es entregado. el primer agentes está encargado de conducir dentro de la pista, por lo general en línea recta, y doblando solo cuando es requerido. Este agente utiliza dos controladores PID, uno para determinar la respuesta longitudinal y el otro para determinar la respuesta lateral del vehículo. El segundo agente corresponde al que realiza la maniobra de adelantamiento, y es en el cual se centra este trabajo de memoria.

Con respecto al agente, estas fueron sus principales características y consideraciones realizadas:

1. El agente utiliza como entrada imágenes obtenidas de cámaras ubicadas en torno al vehículo e información cinemática del mismo. Como salida entrega las señales de *throttle*, *brake* y *steer*, con la cual se controla a *Hero*. Esta respuesta es generada por una red *fully connected*, la cual es entrenada con D-COACH.

2. Se consideraron 2 modelos, uno que utiliza imágenes RGB y de profundidad, y otro que utiliza imágenes de segmentación semántica. De ambos modelos se escogió el que utiliza imágenes de segmentación semántica.

3. Antes de entrar en la red que genera la respuesta del agente, las imágenes de segmentación semántica son pre-procesadas. Estas son transformadas a *one-hot-key*, re-escaladas y transformadas en un histograma acumulativo. Para el re-escalamiento se consideraron 3 métodos: tomar 1 si existía un 1 en el área original, tomar 1 si la mayoría era 1, o tomar el promedio del área. Finalmente se escogió el tercer método, el cual mostró mejores resultados. Por medio del histograma acumulativo se logró reducir aun más la dimensionalidad, conservando las propiedades espaciales de la imagen.

4. Antes de entrar en la red que genera la respuesta del agente, las variables cinemáticas del vehículo son ingresadas a una segunda red, con la cual se aumenta su dimensionalidad. Esto se realiza ya que el número de variables que las componene son considerablemente menos que las generadas por las cámaras, aun después de ser re-escaladas. Al aumentar su tamaño, se reduce el tiempo de entrenamiento y se evita tener pesos sinápticos demasiado grandes en la red principal. La red utilizada también corresponde a una red *fully connected*.

5. Se evaluaron diferentes métodos de post-procesamiento de la salida de la red, con el objetivo de añadir características deseadas en su respuesta que no pudiesen o no se tuvieran que entrenar en la red. Estos fueron la aplicación de un umbral, filtros de frecuencia y función de transformación. Se encontró que ninguno de estos métodos resultó útil, ya que producían efectos indeseados en la señal, como comportamientos oscilatorios.

6. A nivel de la red, se decidió combinar las señales de *throttle* y *brake* en la señal *throttle/brake*. Esto es debido a que no es recomendado activar ambas señales al mismo tiempo, ya que genera

esfuerzos innecesarios en el vehículo. Por esto mismo tampoco se considera *human-like*. Además de esto se tiene que en cuanto *brake* tiene un valor distinto a 0 el vehículo se detiene. Debido a esto se debía entrenando constantemente al agente para que la señal fuera 0. En este caso *throttle* toma el valor de *throttle/brake* cuando este es positivo y *brake* toma el valor absoluto de *throttle/brake* cuando este es negativo. Si una de estas señales no tomar el valor de *throttle/brake* entonces es 0. *throttle/brake* está en el rango $[-1, +1]$.

7. Se realizó un pre-entrenamiento en la red para comenzar los entrenamientos con un comportamiento más estable, en contraposición de iniciar con una respuesta aleatoria. Se probaron 2 pre-entrenamientos: pre-entrenamiento con una red PID y pre-entrenamiento con respuesta cero. En el primer pre-entrenamiento se entrenó la red para que se comportara como el agente PID, el cual solo conduce dentro de la pista. En el segundo pre-entrenamiento se entrenó para que no hiciera nada (respuesta cero). Se observó que el segundo método resultó más conveniente. Esto es debido a que sirve para cualquier maniobra que se desee realizar, no se requiere diseñar otro agente que realice la maniobra por otros medios y solo requiere de una base de datos del estado. Por otro lado, no se observó un aumento considerable en el número de entrenamientos requeridos para converger con respecto al otro método.

El uso de imágenes provenientes de sensores RGB y profundidad, para los cuales se utiliza un AE para reducir su dimensionalidad, tuvo que ser descartado. El motivo de esto fue porque el trabajo se centró en obtener un buen AE, sin considerar su uso con D-COACH. Al momento de ser probado, el programa iba demasiado lento, impidiendo su correcto entrenamiento. También se probó pre-entrenar los AE, aunque con resultados similares. Se deja propuesto seguir explorando esta alternativa.

Se determinó una arquitectura para las redes del agente, aunque no se realizó un análisis de sensibilidad. Estas son la red de re-escalamiento de las variables cinemáticas y la red principal, encargada de generar la respuesta del agente, mencionadas anteriormente. La arquitectura de la red de escalamiento consta de una capa de entrada (4 neuronas), una capa oculta (k/4 neuronas) y una capa de salida (k/2 neuronas), mientras que la red principal tiene una capa de entrada (n neuronas), dos ocultas (2n/3 y 4n/9 neuronas) y una de salida (2 neuronas). Se tiene que k corresponde a la dimensionalidad de las imágenes pre-procesadas y n el número total de entradas de la red principal, siendo este igual a 3k/2.

Con respecto al *feedback*, se determinó que estos fueran desacoplados, siendo cada señal un aumento o disminución de una de las señales entregadas por el agente. También se encontró que un buen error de magnitud para el *feedback* fue de 0.5 para la señal *throttle/brake*, y entre 0.1 y 0.01 para la señal de *steer*. Con este error se evitaban respuestas oscilatorias.

Uno de los objetivos de este trabajo era el uso de D-COACH en un problema más desafiante. Pese a tener resultados satisfactorios, se hacen las siguientes observaciones:

1. A diferencia de los casos simples, su entrenamiento no fue rápido, requiriendo un tiempo de entrenamiento considerable. Esto se debió a que al existir múltiples posibles escenarios, si algunos de estos son similares, pero requieren de una conducta diferente, entrenar el agente para realizar exitosamente uno, podría significar empeorar su desempeño en otro. Es por esto que se debió entrenar de forma reiterada, hasta que conseguía buenos resultados en todos los escenarios.

Ante esto se podría dividir el problema más complejo en un conjunto de problemas más simples, los cuales convergen más rápido, y tener otra red que detecte en cual de estos problemas se está actualmente. Con esto en vez de tener que entrenar una única red por un prolongado periodo de tiempo, se pueden entrenar múltiples redes que realizan tareas especificas en un periodo de tiempo más corto. Con este enfoque el problema se centraría en como dividir los problemas.

2. Existe el peligro de que las observaciones realizadas por los sensores no permitan hacer la distinción entre diferentes escenarios, los cuales pueden requerir de la realización de acciones diferentes. Estas acciones podrían no ser compatibles entre sí o no ser las acciones que está entrenando el profesor humano.

Otro problema encontrado es el entrenamiento de la red para eventos poco frecuentes, pero que eventualmente ocurren, o que suceden demasiado rápido, no dando tiempo para dar el *feedback* correspondiente. Una solución a esto podría ser guardar los estados y acciones, identificar donde ocurren dichos eventos y entregar el *feedback* correspondiente de manera más controlada. Un inconveniente similar es cuando el agente presenta un comportamiento oscilatorio, o entrega respuestas indeseadas de corta duración. Si estas oscilaciones son demasiado rápidas, es muy difícil para el usuario eliminarlas, ocurriendo generalmente lo contrario, incrementando su amplitud. Es por esto que se debe evitar generar este tipo de comportamientos, a la hora de entrar con D-COACH. Mientras se diseñaba el sistema al probar diferentes valores para la magnitud del error, se encontró que valores bajos de esta magnitud permiten reducir este tipo de oscilaciones.

Finalmente se debe tener en cuenta que a diferencia de una persona, el agente diseñado solo podía percibir eventos actuales. Esto es una limitante de la solución propuesta, ya que mientras un usuario puede reconocer el contexto en el que se encuentra, basado en acontecimientos pasados, el agente solo percibe el tiempo presente, por lo que puede malinterpretar las correcciones que se le están dando. Es por esto que para problemas más complejos, se recomienda utilizar *Long Short-Term Memory* (LSTM) [16].

5.1. Trabajo Futuro

Como se mencionó, el agente que mejores resultados dio fue *init zero* con cámaras laterales. Sin embargo, hace falta realizar un análisis de sensibilidad, para poder determinar cuales son realmente los mejores parámetros de su red. También se podría buscar una forma de obtener una trayectoria que describa curvas más regulares, considerando que en la vida real, un movimiento sinuoso, por muy pequeño que sea, puede ser peligroso. Aun si estos no resultan.

Con respecto al uso de imágenes RGB y profundidad, como se mencionó, tuvo que ser descartado, debido a que el enfoque utilizado no fue el correcto, y no disponiendo de más tiempo para seguir con esta opción. En vez de buscar un auto-encoder que entregue una imagen lo más parecida a la entrada, lo cual podría no tener ninguna relación con el desempeño final de la red, se podría partir con una red simple, y a partir de eso ir complejizando. Se podría partir con una imagen en escala de grises más la imagen de profundidad. Del mismo modo, partir con un espacio de latencia pequeño y a partir de eso ir aumentando su tamaño. También se deja propuesto la implementación del sistema con LIDAR.

En los entrenamientos, estos se realizaron en torno a vehículos que solo eran controlados por máquinas. Considerando que uno de los problemas de los vehículos autónomos es su respuesta poco natural

ante otros conductores humanos, se podría considerar realizar entrenamiento en donde también estén involucrados vehículos controlados por humanos.

En esta memoria se centró en la maniobra de adelanto, sin embargo esto se puede extender a otras maniobras realizadas por un vehículo. Se propone entrenar la maniobra de estacionamiento paralela, para el cual existen algoritmos que resuelven este problema de forma directa, con los cuales se podrían comparar resultados. Eventualmente se podría tener un vehículo totalmente autónomo, en donde cada acción es realizada por un agente diferente entrenado con D-COACH. Como se vio, este tiende a tener un comportamiento humano, lo cual es una ventaja con respecto a otros sistemas.

6. Bibliografía

[1] R. Pérez-Dattari, C. Celemin, J. R. del Solar, and J. Kober, "Interactive learning with corrective feedback for policies based on deep neural networks," September 2018.

[2] C. Celemin and J. R. del Solar, "Coach: Learning continuous actions from corrective advice communicated by humans," in *2015 International Conference on Advanced Robotics (ICAR)*, pp. 581–586, July 2015.

[3] R. Pérez-Dattari, C. Celemin, J. R. del Solar, and J. Kober, "Continuous control for high-dimensional state spaces: An interactive learning approach," April 2019.

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, November 2017.

[5] CONASET, *Manual del Conductor Clase B*, p. 10. Automóvil Club de Chile, July 2012.

[6] S. Haykin, *Neural Networks and Learning Machines*, ch. Introduction and Multilayer Perceptrons, pp. 1–46 and 122–229. Person, 3rd edition ed., 2009.

[7] VectorZero, "Roadrunner," 2020. Accessed: Feburery 13, 2020. [Online]. Available: `https://www.vectorzero.io/roadrunner`.

[8] A. J. Hawkins, "The world's first robot car death was the result of human error — and it can happen again," *The Verge*, November 2019. Accessed: March 23, 2020. [Online]. Available: `https://www.theverge.com/2019/11/20/20973971/uber-self-driving-car-crash-investigation-human-error-results`.

[9] M. Cruickshank, "Google autonomous vehicle causes first accident," *The Manufacturer*, March 2016. Accessed: March 23, 2020. [Online]. Available: `https://www.themanufacturer.com/articles/google-autonomous-vehicle-causes-first-accident/`.

[10] J. Hirsch and J. Serna, "Humans at fault in self-driving car crashes," *Los Angeles Time*, May 2015. Accessed: March 23, 2020. [Online]. Available: `https://www.latimes.com/business/la-fi-self-driving-accidents-20150512-story.html`.

[11] S. Hecker, D. Dai, and L. V. Gool, "Learning accurate, comfortable and human-like driving," March 2019.

[12] Q. Zhu, Z. Huang, Z. Sun, D. Liu, and B. Dai, "Reinforcement learning based throttle and brake control for autonomous vehicle following," in *2017 Chinese Automation Congress (CAC)*, pp. 6657–6662, October 2017.

[13] J. Naranjo, C. González, R. Garcia, and T. Pedro, "Cooperative throttle and brake fuzzy control for acc+stopgo maneuvers," *Vehicular Technology, IEEE Transactions on*, vol. 56, pp. 1623 – 1630, 08 2007.

[14] X. Zhang, J. Zhao, and G. Wang, "Coordinated throttle and brake switching control and rcp design based on dspace for intelligent vehicle," in *2014 Seventh International Symposium on Computational Intelligence and Design*, vol. 1, pp. 577–580, December 2014.

[15] X. Zhongpu and Z. Dongbin, "Hybrid feedback control of vehicle longitudinal acceleration," in *Proceedings of the 31st Chinese Control Conference*, pp. 7292–7297, July 2012.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.