ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

**GRADO EN INGENIERÍA DEL SOFTWARE**

**Estudio de la conducción automática simulada aplicando técnicas inteligentes**

**Study of simulated automatic driving by applying intelligent techniques**

Realizado por

**Olivier Gabana Gómez**

Tutorizado por

**Eduardo Guzmán de los Riscos**

Departamento

**Lenguajes y ciencias de la computación**

UNIVERSIDAD DE MÁLAGA

MÁLAGA, FEBRERO DE 2025

Fecha defensa: 7 de julio de 2025

Resumen

En este proyecto se busca encontrar las limitaciones de la últimamente popularizada Inteligencia artificial, en este caso será aplicada al mundo de la conducción autónoma, y la metodología puesta a prueba será el aprendizaje por refuerzo.

Para realizar este proyecto se hará uso de una herramienta llamada Carla, la cual nos permitirá crear un mundo en el cual se encontrará el vehículo autónomo, para así poder entrenarlo y encontrar sus limitaciones.

En primera instancia se entrenará al vehículo únicamente haciendo uso de sensores básicos, como lo son el sensor de colisión, cruzado de línea y obstáculos. Por lo tanto, la idea es ver hasta donde es capaz de llegar un coche que conduce "a ciegas".

En segunda instancia esta vez se le añadirá al vehículo un sensor mucho mas avanzado llamado LIDAR (Light Detection and Ranging), el cual genera una imagen de lo que se encuentra delante del vehículo lanzando varios rayos láser para así poder saber a qué distancia se encuentran los obstáculos.

**Palabras clave: Aprendizaje, Refuerzo, Inteligencia, Conducción, Automatización.**

Abstract

This project seeks to find the limitations of the recently popularized Artificial Intelligence, in this case it will be applied to the world of autonomous driving, and the methodology tested will be reinforcement learning.

To carry out this project, a tool called Carla will be used, which will allow us to create a world in which the autonomous vehicle will be located, in order to train it and find its limitations.

In the first instance, the vehicle will be trained only using basic sensors, such as the collision sensor, line crossing and obstacles. Therefore, the idea is to see how far a car that drives "blindly" can go.

In the second instance, this time a much more advanced sensor called LIDAR (Light Detection and Ranging) will be added to the vehicle, which generates an image of what is in front of the vehicle by firing several laser beams in order to know how far away the obstacles are.

**Keywords: Reinforcement, learning, autonomous**

Índice

Resumen 1

Abstract 1

Índice 1

Introducción 1

1.1 Motivación 1

1.2 Objetivos 1

1.3 Estructura de la memoria 3

Referencias 9

Manual de Instalación 11

Requerimientos: 11

1

Introducción

1.1 Motivación

La motivación de este proyecto surge de la gran revolución que estamos experimentado últimamente en cuanto a las inteligencias artificiales, como por ejemplo el mundialmente conocido ChatGpt, el cual te responde de una forma más o menos acertada a cualquier pregunta que tengas sobre cualquier tema, desde un suceso histórico hasta que te resuelva una integral y te muestre el procedimiento que ha realizado. También existen otras aplicaciones de inteligencia artificial, como en las últimas generaciones de tarjetas grafica de la empresa NVIDIA, la cual hace uso de esta para generar fotogramas extras a la hora de jugar a videojuegos para así mejorar la experiencia sin necesidad de tener un procesador gráfico tan potente.

En el mundo de la automoción existe un gran caso de éxito el cual no es ni más ni menos que el *autopilot* de Tesla, el cual es en parte una gran parte de la inspiración para la realización de este proyecto.

1.2 Objetivos

Existen 2 objetivos principales a la hora de realizar este proyecto, aprendizaje del funcionamiento de la inteligencia artificial y ver las capacidades reales de esta misma.

Ya que la inteligencia artificial se encuentra en un momento de gran crecimiento y no existen todavía una gran cantidad de profesionales especializados en esta materia, se considera un conocimiento bastante valioso en el mercado laboral, por esta razón uno de los objetivos principales es el aprendizaje del funcionamiento interno de la inteligencia artificial.

Últimamente se intenta aplicar inteligencia artificial a todos los sectores, pero, ¿es realmente tan "inteligente" como parece?, esta pregunta es considerada bastante importante para así poder saber si es simplemente una innovación que estará un tiempo de moda o si ha llegado para quedarse en el mundo de la tecnología. Por lo tanto uno de los objetivos de este proyecto es ver la capacidades reales de la inteligencia artificial y si tiene la capacidad real de suplantar tareas que actualmente son solo realizadas por personas, como por ejemplo la conducción (aunque ya existen sistemas bastante desarrollados como el antes mencionado "autopilot" de Tesla), o si en cambio es algo mucho más simple de lo que pensamos y por ejemplo aplicado al caso de la conducción solo sería capaz de conducir en trazadas muy simples sin una gran cantidad de tráfico.

1.3 Estructura de la memoria

En primer lugar, se hará una explicación sobre todos las herramientas y tecnologías utilizadas para el desarrollo de este proyecto, para así poder contextualizar correctamente toda la explicación sobre el desarrollo de software.

En este apartado sobre las herramientas utilizadas, se hará un resumen sobre la herramienta Carla, la cual nos permite simular todo el entorno en el que se desenvolverá nuestro vehículo autónomo, Carla permite crear tráfico artificial, cambiar entre ciudades, alterar el clima, y muchas otras funciones que hacen posible que se le dedique la mayoría del tiempo al desarrollo de la inteligencia artificial en sí.

También se hablará de todas las herramientas utilizas para el desarrollo del "cerebro" de nuestro vehículo autónomo, principalmente librerías de Python, sin las cuales el desarrollo del software sería mucho más complejo.

En segundo lugar, se hablará sobre la metodología de trabajo que se ha utilizado durante todo el desarrollo del proyecto, es este caso la metodología utilizada será una metodología de tipo ágil.

*Una* ***metodología ágil*** *es un enfoque flexible e iterativo para la gestión de proyectos y desarrollo de software que prioriza la colaboración, la entrega continua y la adaptación al cambio. Se basa en principios del* ***Manifiesto Ágil****, como la comunicación cercana con el cliente, la mejora continua y la entrega de valor en ciclos cortos (sprints). Ejemplos comunes incluyen* ***Scrum, Kanban y XP (Extreme Programming)***

En tercer lugar, se hablará de todo el desarrollo del software del proyecto, se dividirá en las diferentes fases en las cuales se ha desarrollado el proyecto. En cada una de estas fases se explicarán todas las decisiones tomadas desde el principio hasta el final de la fase, para así poder entender todas las decisiones de diseño y modificaciones que se han ido haciendo a lo largo del desarrollo de la fase, para así poder obtener los mejores resultados posibles.

En la primera fase, como antes se ha mencionado, solo se hará uso de sensores más "simples", como lo pueden ser el sensor de colisión, cruzado de línea, ... Y en vez de usar una red neuronal con una alta complejidad, solo se hará uso de aprendizaje por refuerzo ( no profundo), por lo que entonces se hará uso Qtables las cuales serán tratadas con más profundidad más adelante. El objetivo de esta fase es ver hasta que punto con un sistema lo más "sencillo" posible, es capaz de conducir un coche autónomo.

En la segunda fase, se añadirá a todo nuestro set de sensores un nuevo sensor, el cual es el LIDAR (Light Detection and Ranging), el cual es un sensor mucho más complejo, y es capaz de generar imágenes con puntos de colores que simulan un láser, para así poder saber la distancia a la que se encuentran los objetos del alrededor el vehículo.

Además en esta fase no se hará uso de aprendizaje por refuerzo como en la fase anterior, sino que se hará uso de aprendizaje por refuerzo profundo, el cual es un método bastante más complejo que el anterior, gracias al cual deberíamos de obtener mucho mejores resultados que con el anterior. Sin embargo no son todo ventajas, ya que la complejidad del desarrollo del software aumenta bastante, al igual que en la anterior fase, se tratará más en profundidad más adelante para así poder comprender las diferencias.

En último lugar se analizarán los resultados obtenidos, para así poder comparar las metodologías utilizadas, y ver si realmente merece la pena crear una red neuronal compleja. En este apartado también se tratará la pregunta planteada al principio (¿es realmente tan "inteligente" como parece?), ya que es una de las principales motivaciones de la realización de este proyecto.

2

Tecnologías a utilizar

Carla

Para la realización de este proyecto se hará uso del antes mencionado Carla, un software que nos permitirá generar un mundo con todo el tráfico simulado y añadir nuestro coche autónomo para poder entrenarlo.

Existen otros Framework y programas que también nos permitirían realizar el proyecto, como por ejemplo: LGSVL, NVIDIA DRIVE Sim, Microsoft AirSim; pero, ¿Porque hemos elegido Carla y no otro Software?, a continuación se enumerarán las mayores ventajas que tiene Carla frente a los otros Software que se encuentran en el mercado:

1. Carla es gratuito lo cual es una gran ventaja, ya que reduce el coste de desarrollo considerablemente frente a otros simuladores como lo puedo ser NVIDIA DRIVE Sim.
2. Otra gran ventaja de Carla es que nos aporta una simulación muy realista frente a otros simuladores como por ejemplo LGSVL, el cual no tiene una mala calidad, pero no es comparable a la de Carla ya que es mucho más completa incluyendo por ejemplo fenómenos meteorológicos, sombras más detalladas, ...
3. Carla soporta una mayor cantidad y calidad de kit de sensores, ya que incluye sensores avanzados como el LIDAR, Cámaras de profundidad, GPS... Por lo tanto, en este aspecto es el claro ganador frente a otros AirSim, el cual está más enfocado en drones que en vehículos, y frente a LGSVL, el cual no tiene un kit de sensores tan completo.
4. La integración nativa de TensorFlow, PyTorch y OpenAI Gym le da una gran ventaja frente de AirSim, Carla trae integrados estos elementos ya que esta pensado precisamente para el entrenamiento de coches aútonomos, al contrario de AirSim.
5. Carla es Open-Source, lo cual es una ventaja de agradecer, ya que puedes modificar la simulación en todo lo que desees, al contrario que NVIDIA DRIVE Sim, que no puedes modificar nada en absoluto de la simulación.
6. La cantidad de información que puedes encontrar en la web (desde vídeos, webs, foros, documentación en la página oficial) es abrumante en comparación con los otros Software, ya que gracias a las ventajas que han sido mencionadas anteriormente, es la mejor opción a elegir, hay una gran cantidad de personas que comentan resultadas y muestran como ellos han configurado su coche autónomo para que pueda circular solo. Cabe mencionar la gran documentación que se puede encontrar en la página oficial de carla, ya que no solo muestra las funciones que tienen las clases de los actores que hay en la simulación, sino que también te guía paso a paso para poder arrancar y configurar tu simulación sin problemas.

Por estas razones es por la que hemos elegido a Carla frente a otras alternativas que se pueden encontrar en el mercado.

Python 3.7 y librerías

El lenguaje en el que se programará nuestro vehículo autónomo será sin duda alguna Python, ya que es el lenguaje más preparado para la programación de inteligencia artificial con diferencia.

3

Metodología de trabajo

Introducción a las metodologías ágiles

Como antes se ha mencionado en el apartado de *Estructura de la memoria*, el proyecto será desarrollado en base a una metodología ágil, la cual nos permitirá cierta flexibilidad a la hora de desarrollar el proyecto, ya que esta nos permite modificar cosas que han sido realizadas al principio el proyecto, pero que a lo largo del desarrollo de fases posteriores suponen un problema y deben de ser cambiadas para así poder terminar de alcanzar todos los objetivos del proyecto.

Las metodologías ágiles no solo tienen esta ventaja de poder modificar algo que en primera instancia no hemos resuelto de la mejor manera posible con respecto a otros tipos de metodologías usadas en el pasado como por ejemplo la metodología en cascada, la cual entre otras cosas también tenía como desventaja la falta de interacción con el cliente, sino que también, tiene muchas otras ventajas como, por ejemplo:

* Es posible realizar cambios en los requisitos a lo largo del proyecto, lo cual es una gran ventaja, ya que, si a lo largo del desarrollo aparece una nueva idea de gran utilidad o un cambio de algún requisito que no se planteó correctamente con anterioridad, se podría modificar sin grandes dificultadas y sin que supusiese un gran costo para el proyecto.
* Iteraciones cortas (con respecto a metodologías no ágiles como la metodología en cascada), lo que supone una ventaja, ya que al final de cada iteración se debe revisar todo lo que se ha hecho y se pueden plantear modificaciones y posibles nuevas funcionalidades para poder alcanzar las metas deseadas.
* Continuo Feedback del desarrollo, no solo los desarrolladores se comunicarán entre ellos al final de cada sprint, para que así todo el mundo este al tanto del proyecto, sino que también se mantendrá continuamente comunicado al cliente de como esta avanzando el desarrollo del proyecto, para así poder corregir cualquier posible malentendido con el cliente que se haya producido en fases tempranas y poder corregirlo cuanto antes.
* Todas las ventajas antes mencionadas, que en general nos ayudan a poder abordar los problemas de forma temprana no solo nos ayuda a ahorrar tiempo, lo que se traduce en un ahorro del coste total del proyecto, ya que se necesitaran menos horas de trabajo para poder realizarlo. Por lo tanto otra gran ventaja de esta metodología serían los menores costes de desarrollo.

Funcionamiento y etapas de las metodologías ágiles

Para explicar como son las fases de las metodologías ágiles, vamos a hacer una breve explicación sobre el funcionamiento de una de las metodologías ágiles más aplicadas actualmente la cual es Scrum, y es en la que esta basada la metodología que se va a emplear en el este proyecto. Antes de comenzar con las fases, cabe recalcar que cada miembro perteneciente al grupo desarrolla un rol en concreto (Jefe de proyecto, responsable de calidad,...) para que así cada persona se centre en un aspecto en concreto.

La primera fase de todas es la creación del product backlog, en esta fase el jefe de proyecto se encargará de obtener todos los requisitos necesarios para el desarrollo del proyecto.

Una vez obtenidos todos los requisitos del proyecto (por el momento), se organizarán todos los Sprits, estos sprints son ciclos de entre 1 y 2 semanas, en cada Sprint se insertaran las tareas que se deben de realizar en ese periodo de tiempo, habrá breve reunión diaria para que los trabajadores expongan todas las dificultades o dudas que estén encontrando, y al final de cada sprint habrá una reunión más extensa en la cual se podrán modificar requisitos si es necesarios, ya que es una metodología ágil.

En el momento que todos los sprints esten completos, significara que no quedan tareas por realizar en el product backlog, por lo tanto el proyecto estará finalizado.

Aplicación en este proyecto

Para el desarrollo de este proyecto, al ser solo 1 desarrollador, no se podrá exprimir al máximo esta metodología, aún así, es imprescindible el uso se esta ya que aporta una gran flexibilidad a la hora del desarrollo.

En el product backlog se encontrarán todos los objetivos que se desean alcanzar, los cuales son:

* Configuración del simulador Carla
* Creación del coche autónomo
* Acoplamiento de los sensores simples y lectura de sus valores
* Creación de la primera fase IA, para conducción autónoma
* Testeo y configuración de parámetros primera fase IA
* Entrenamiento final primera fase IA
* Acoplamiento del LIDAR al coche autónomo y procesamiento de la información
* Testeo y configuración de parámetros segunda fase IA
* Entrenamiento final segunda fase IA
* Comparación de resultados

Las tareas serán repartidas en varios sprints, y cada sprints debería de ser una duración aproximada de 1-2 semanas.

Al final de cada sprint se revisará todo el trabajo hecho y se incluirán más requisitos si se considera necesario.

4

Desarrollo del Software

Instalación, configuración y funcionamiento de Carla

Para comenzar con la explicación del desarrollo del proyecto, hay que empezar con una explicación del funcionamiento de la herramienta principal de la que se va a hacer uso, la cual es Carla.

Vamos a empezar explicando como es la instalación de este Software. Hay que dirigirse a su repositorio de github (https://github.com/carla-simulator/carla/releases) y descargar la versión deseada, en nuestro caso será la versión 0.9.15.

Una vez descargada la versión deseada, hay que descomprimir el archivo que hemos descargado, el cual nos generará una carpeta con ciertos archivos y otros subdirectorios.

El principal archivo en el que estamos interesados es CarlaUE5.exe, el cual es un archivo ejecutable que nos ejecutará el servidor de la simulación y nos generará una ventana, que no es más que un espectador mediante el cual podremos movernos y ver lo que está ocurriendo en nuestro servidor a tiempo real. Cabe recalcar que este servidor y su espectador consumen una gran cantidad de recursos computacionales, en especial recursos gráficos. Los requisitos recomendados para la ejecución son los siguientes:

* Al menos 6 GB de VRam, lo recomendado es 8 GB
* Sistema linux o Windows
* Al menos 20 GB de espacio libre de almacenamiento
* Se recomiendan 32 GB de RAM
* Un procesador de al menos 4 núcleos a una velocidad de 2.5Ghz

En la carpeta en la que extrajimos la aplicación también tenemos el proyecto de UnrealEngine del cual fue renderizado nuestra simulación, por lo tanto si se desea modificar cualquier cosa de la simulación o incluso crear nuevos mapas o vehículos se puede realizar sin ningún tipo de problemas, lo cual es una gran ventaja como se mencionó en capítulos anteriores.

Además de la todo lo que es la simulación en sí, también tenemos un conjunto de scripts de ejemplo en python (en el directorio /PythonAPI/examples) que nos permiten realizar varias pruebas para ver como se comporta la simulación, uno de los más útiles es manual\_control.py el cual nos generará una ventana con un vehículo el cual podremos manejar con nuestros periféricos y que además no da toda la información que se puede obtener a partir de los sensores básicos que se le pueden añadir al vehículo.

A continuación, se enumerará las funcionalidades principales que presenta Carla, para que así se puedan comprender las limitaciones y capacidades de este (cabe recalcar que todas estas configuraciones se hacen desde scripts python):

* Para poder conectarnos al servidor y ejecutar todas las acciones necesarias debemos de ejecutar las siguientes línea de código carla.Client('localhost', 2000) (por defecto el servidor se ejecuta en el puerto 2000), cliente.set\_timeout(5.0), las cuales nos permiten crear el cliente para posteriormente poder realizar cambios en nuestra simulación.
* Para poder obtener el mundo que se está simulando actualmente tenemos el comando cliente.get\_world()
* Como antes se ha mencionado, no existe un solo mundo que podemos simular, sino que existe una amplia variedad para las diferentes necesidades que pueden tener los usuarios, incluso si se desea una mayor selección de mapas se puede descargar gratuitamente un paquete de extensión en el cual podemos encontrar más mundos para la simulacion. Si se desea echar un vistazo a todos los mapas disponibles, se encuentra una previsualización en la siguiente url, la cual pertenece a la documentación de carla https://carla.readthedocs.io/en/0.9.15/tuto\_first\_steps/ , en el apartado llamado Choose your map. Para poder cambiar el mundo de la simulación es necesario ejecutar el siguiente comando: cliente.load\_world('Nombre del mapa'), todos los mapas disponibles se deben encontrar en el directorio "HDMaps".
* Si se desea también se puede modificar la meteorología de la simulación, para obtener la meteorología actual es mediante el siguiente comando world.get\_weather(), y si se desea establecer cierta meteorología se puede realizar de la siguiente manera, weather.sun\_altitude\_angle = -30, weather.fog\_density = 65, weather.fog\_distance = 10, world.set\_weather(weather).
* Existe la posibilidad de grabar lo que ocurre en la simulación desde un punto de vista en concreto, esto se puede gracias al comando client.replay\_file("recording01.log", start, duration, camera)
* En el caso de tener recursos insuficientes para la ejecución fluida del servidor, existen varias opciones para poder ejecutar la simulación, la primera de ellas es ejecutar la simulación con una menor calidad, para ello a la hora de ejecutar el servidor es necesario hacerlo de la siguiente manera desde el terminal, ./CarlaUE4.sh -quality-level=Low. También es posible no renderizar el servidor, con lo cual se ahorrarían una gran cantidad de recursos gráficos, para realizar esto es necesario ejecutar el script llamado config.py con la opción --no-rendering, este script lo podemos encontrar en la carpeta examples antes mencionada.
* Para obtener todos los posibles actores, ya sean vehículos o sensores, necesitamos obtener la libreria con todas las blueprints, esto se hace con el siguiente comando, world.get\_blueprint\_library().
* Si se necesita obtener todos los puntos del mundo en los que se spawnearán los actores, si se ejecuta el siguiente comando se obtendrá una lista con todos los posibles puntos, world.get\_map().get\_spawn\_points().

El respawn y configuración, ya sean NPCs o nuestro vehículo autónomo, se puede considerar como configuración del servidor, pero al ser algo más específico e importante, se explicará más detenidamente en los próximos apartados.

Creación, y puesta en marcha del vehículo autónomo

Una vez tenemos arrancado y configurado nuestro servidor, es el momento de spawnear todos los actores de nuestra simulación.

Primero, antes de spawnear nuestro vehículo autónomo, debemos de spawnear todos los NPCs (non-playable character, es decir, vehículos que forman parte de la simulación y que se mueven solos de forma autónoma y aleatoria) para así poder dotar de realismo a nuestra simulación.

Existen varias formas de spawnear estos vehículos y peatones. La primera de ellas es haciendo uso del generate\_traffic.py (que podemos encontrar en la carpeta examples) acompañado de -n para especificar el numero de vehículos, y -w para especificar el número de peatones, en caso de que no se especifique alguno de ellos se spawnearán 30 vehículos y 10 peatones. La segunda opción que tenemos es añadirlo en nuestro propio script, para spawnear los vehículos primero hay que saber en que punto queremos que aparezca el actor, y después para que aparezca en el mapa hay que ejecutar el siguiente comando, world.try\_spawn\_actor(random.choice(vehicle\_blueprints), random.choice(spawn\_points)) (en este caso será un vehículo aleatorio en un punto aleatorio), y para que el vehículo circule de forma aleatoria alrededor del mapa es necesario el siguiente comando, vehiculo.set\_autopilot(True). En el caso de los peatones es necesario spawnear el propio peaton, el cual se hace igual que el vehiculo anterior, pero además hay que spawnear un controlador que lo maneje, esto se hace de la siguiente manera, walker\_controller\_bp = world.get\_blueprint\_library().find('controller.ai.walker')

world.SpawnActor(walker\_controller\_bp, carla.Transform(), parent\_walker), ai\_controller.start()

ai\_controller.go\_to\_location(world.get\_random\_location\_from\_navigation())

ai\_controller.set\_max\_speed(1 + random.random()).

Una vez hemos inicializado todo el tráfico de la ciudad, es hora de crear nuestro coche autónomo y añadirle todos los sensores para poder obtener toda la información del entorno que posteriormente utilizaremos para el entramiento de nuestra inteligencia artificial.

Explicación teórica Fase 1

Programación y configuración Fase 1

Resultados obtenidos Fase 1

Instalación (y funcionamiento) del sensor LIDAR en el vehículo autónomo

Explicación teórica Fase 2

Programación y configuración Fase 2

Resultados obtenidos Fase 2

5

Comparación de resultados entre fases

5

Conclusiones y líneas futuras

Referencias

https://agilemanifesto.org/ -- definición de metodologías agiles

Apéndice A

Manual de uso

Requerimientos: