Entorno de simulación para vehículos automatizados con CARLA

Asier Arizala, Daniel Campelo, Asier Zubizarreta
Depto. de Ingeniería de Sistemas y Automática, Universidad del País Vasco (UPV/EHU)
asier.arizala@ehu.eus, dcampelo001@ikasle.ehu.eus, asier.zubizarreta@ehu.eus
Joshué Pérez
TECNALIA, Basque Research and Technology Alliance (BRTA)
joshue.perez@tecnalia.com

Resumen

El interés alrededor de los vehículos automatizados (VA) ha crecido considerablemente en los últimos años debido a la necesidad de conseguir un método de transporte más eficiente y seguro. Sin embargo, el desarrollo de esta tecnología es una tarea muy compleja, ya que es necesario validar e integrar una gran variedad de funcionalidades. Así mismo, el número de escenarios particulares que se requieren estudiar para asegurar una exitosa automatización hace que el testeo en carreteras reales no sea viable. Debido a esto, ha aumentado el interés por invertir en el desarrollo de entornos de validación virtuales, pudiendo encontrar tanto soluciones comerciales como de código abierto. En este trabajo se propone un entorno de simulación para aplicaciones de vehículos automatizados basado en CARLA, en el que se integran, por un lado, un mapa de una manzana de Bilbao y, por otro, el modelo de un Renault Twizy. De esta forma, se introducen las bases para validar futuros desarrollos en esta misma ubicación real.

In recent years the interest in automated vehicles (AV) have increased due to the need of a safer and more efficient way of travelling. However, the validation of this technology is rather a complex task. Additionally, the amount of particular scenarios for which the technologies have to be tested makes the practical validation not a viable option. That is why recently the virtual testing environments are gaining a lot of popularity. In this work a testing environment is proposed using the open source simulator CARLA, in which a map of a part of Bilbao and a model of a Renault Twizy are integrated. Thanks to this work, future AV related work will be validated with real world data.

Palabras clave: Vehículos automatizados, Entornos de simulación, Control.

1 Introducción

En los últimos años tanto la comunidad de investigadores como las grandes compañías han realizado grandes inversiones de tiempo y dinero en el desarrollo de vehículos automatizados (VA) como una solución eficiente y segura para el transporte de personas y mercancías. Con el tiempo, la posibilidad de ser la primera marca o grupo de investigación en ofrecer esta tecnología ha generado una gran competitividad en el sector. El objetivo es lograr un vehículo con un nivel de automatización de nivel 5 según la SAE (Society of Automotive Engineers) [9]. Sin embargo, se trata de un problema muy complejo. Si bien el estudio sobre algoritmos independientes de la arquitectura general de los VA como los de reconocimiento de objetos [12], toma de decisiones [10], generación de trayectorias y control [1] es extenso hoy en día, el mayor problema reside en su interacción y validación en situaciones complejas.

A la hora de comprobar la validez de los diferentes módulos de un VA, tradicionalmente se realizan pruebas en pista. Sin embargo, todas estas funcionalidades necesitan superar una gran cantidad de situaciones para poder justificar que son seguras para el usuario final, por lo que la cantidad de horas requerida para tal fin hace inviable el uso de la anterior aproximación. Una de las soluciones a tal dilema es la utilización de entornos virtuales [11], donde el usuario posee un completo control sobre todos los agentes involucrados en cada simulación, reduciendo así tiempo y costes.

Dichos entornos o simuladores requieren por tanto de tres características principales. Primero, el realismo necesario para que los algoritmos de percepción testeados sean extrapolables a entornos reales. Segundo, modelos matemáticos capaces de replicar la dinámica vehicular de forma precisa. Dado que los algoritmos de control son dependientes de los modelos a controlar, la validez de estos dependerá de la semejanza entre el modelo virtual y el real. Tercero, una interfaz que ayude al usuario a generar los escenarios requeridos.

Con el tiempo, estas soluciones han ido adaptándose a la aparición de las nuevas tecnologías para adoptar las características mencionadas. De esta forma, se pueden separar 2 tipos de simuladores. Por un lado, los más antiguos y más centrados en la reproducción de la dinámica vehicular tales como CarSim [2] o DSpace [6] ofre-

https://doi.org/10.17979/spudc.9788497498043.435

cen la alta fidelidad de un software validado durante años. Por otro lado, aprovechando el gran avance en el modelado tridimensional de entornos que ha evolucionado con las generaciones de videojuegos, se ha desarrollado un segundo conjunto de simuladores que priorizan el realismo visual sobre la fidelidad de los modelos dinámicos. Uno de los primeros que impulsó el paso hacia esta nueva filosofía fue rFactor pro [8], una evolución del videojuego con el mismo nombre. Recientemente, grandes empresas en el ámbito del diseño gráfico como es Nvidia, han propuesto sus propios entornos de simulación [5] con integraciones de inteligencia artificial. Otros ejemplos son Cognata [3], Metamoto [7], o la plataforma open source CARLA [4], a la cual apoyan Intel y Toyota.

Una vez se dispone de un entorno de simulación es necesario realizar una interpretación de un escenario basado en lugares reales que más tarde pueda servir como método de validación. El realismo de dicho escenario es necesario para que los algoritmos de percepción sean completamente extrapolables, mientras que la fidelidad de los modelos dinámicos utilizados ayudan a los controladores. Se trata de un trabajo complejo que puede resultar en cuellos de botella durante proyectos a largo plazo.

En este trabajo se presenta un entorno de simulación para vehículos automatizados basado en CARLA que usa una representación de una manzana de Bilbao para la posterior validación de algoritmos relacionados en vehículos automatizados en prototipos reales. También introduce el modelo de un Renault Twizy para poder ratificar los resultados obtenidos en un futuro con el propio vehículo en el entorno real.

El resto del artículo está estructurado de la siguiente manera. En la sección 2 se define el entorno de simulación propuesto, detallando la geometría del mapa y el modelo de vehículo introducidos. En la sección 3 se presenta un caso de uso mediante el cual se demuestra la funcionalidad de la solución propuesta. Para finalizar, en la sección 4 se concretan las conclusiones y los trabajos futuros.

2 Entorno propuesto

A fin de conseguir un entorno donde probar algoritmos relacionados con VA, es necesario contar con entornos adaptados a los recursos disponibles, ya que el fin de cada desarrollo pasa por su integración en vehículos reales.

El entorno de simulación desarrollado (Figura 1) se ha implementado en CARLA, abarcando la introducción de un mapa y un vehículo propios, así como la inclusión de un controlador de seguimiento de trayectorias para realizar pruebas de rendimiento. En este caso, se ha optado por modelar una manzana de Bilbao, y un Renault Twizy, vehículo que posee el grupo de Automated Driving de Tecnalia para testear aplicaciones de VA.

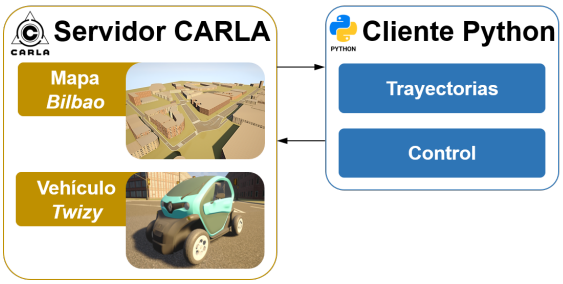

Figura 1: Entorno de simulación propuesto

2.1 CARLA

Presentado por primera vez en 2017 CARLA ha ganado popularidad rápidamente entre los desarrolladores de aplicaciones para VA. La posibilidad de realizar simulaciones en entornos realistas, gracias al desarrollo en Unreal Engine, su fácil accesibilidad, a través de tutoriales, manuales y foros, y el hecho de ser de código libre lo convierten en una propuesta muy atractiva.

Actualmente cuenta con una extensa biblioteca de modelos con un total de 8 mapas, 13 sensores, 31 vehículos, 26 peatones, y 90 objetos estáticos, que va creciendo con cada nueva actualización. Además, como es el caso de este trabajo, es posible añadir contenido personalizado gracias a la compatibilidad con programas externos como RoadRunner de Mathworks. Adicionalmente, cabe mencionar que utiliza una arquitectura cliente/servidor utilizando Python como lenguaje de programación.

Dicho esto, otra de sus ventajas es la compatibilidad con ROS que proporciona el CARLA-ROS-bridge, lo cual permite integrar soluciones que se vayan a aplicar en hardware directamente.

2.2 Mapa introducido

Se ha modelado un área aproximada de 0,35 km^2 de Bilbao comprendiendo el paso por la Escuela de Ingeniería hasta la Plaza Sagrado Corazón y sus intersecciones. El trazado contiene el conjunto de semáforos, señales y pasos de peatones, así como una zona de parking adicional para pruebas específicas.

El modelado del mapa se ha realizado siguiendo la secuencia de la Figura 2. Para la generación de carreteras se ha utilizado el software RoadRun-

https://doi.org/10.17979/spudc.9788497498043.435

ner. Se trata una aplicación de MathWorks que permite de forma sencilla crear escenarios y que admite la importación de imágenes, archivos OSM o nubes de puntos como plantillas. En este caso, se ha utilizado un archivo OSM de la zona escogida, puesto que se tratan de archivos precisos con puntos de referencia que facilitan el trabajo.

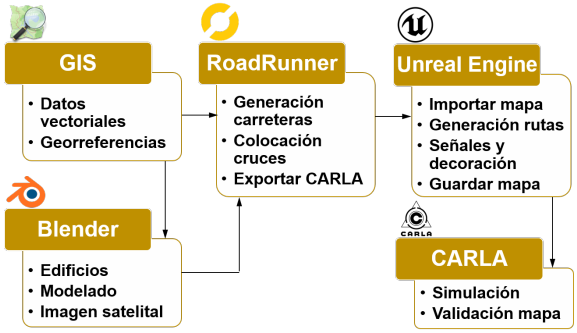

Figura 2: Generación de mapa para CARLA

En cuanto a los edificios, se ha utilizado Blender para modelarlos. Este programa de código abierto, comúnmente utilizado en la industria del videojuego, permite al usuario recrear objetos o personajes mediante mallas estructuradas en poliedros que más tarde se introducirán en un espacio virtual gracias a los motores gráficos como Unreal Engine. Dado que se requiere cierta precisión y coherencia con las carreteras generadas se ha utilizado el mismo archivo OSM para esta tarea también. Se han añadido texturas a los laterales de los edificios con el fin de servir de referencia a aplicaciones de visión artificial.

Finalmente, se han introducido los semáforos, señales y pasos de peatones correspondientes a lo largo de la carretera utilizando los assets de la biblioteca de CARLA como plantilla.

2.3 Modelo de vehículo

La introducción de un nuevo vehículo en CARLA tiene una vertiente doble: por un lado es necesario realizar un modelado que resulte adecuado y por otro llevar a cabo las configuraciones de planos (*blueprints*) en Unreal Engine. En el entorno propuesto se ha incorporado un Renault Twizy.

Con tal de llevar a cabo el modelado físico, se ha hecho uso de la herramienta de modelado 3D Blender una vez más. Este programa también permite la generación de esqueletos que posibilitan el movimiento relativo entre segmentos mallados. En el caso de un vehículo es necesario definir uniones entre el chasis y las ruedas que permitan simulaciones más realistas de la suspensión del vehículo. Esta tarea se ha llevado

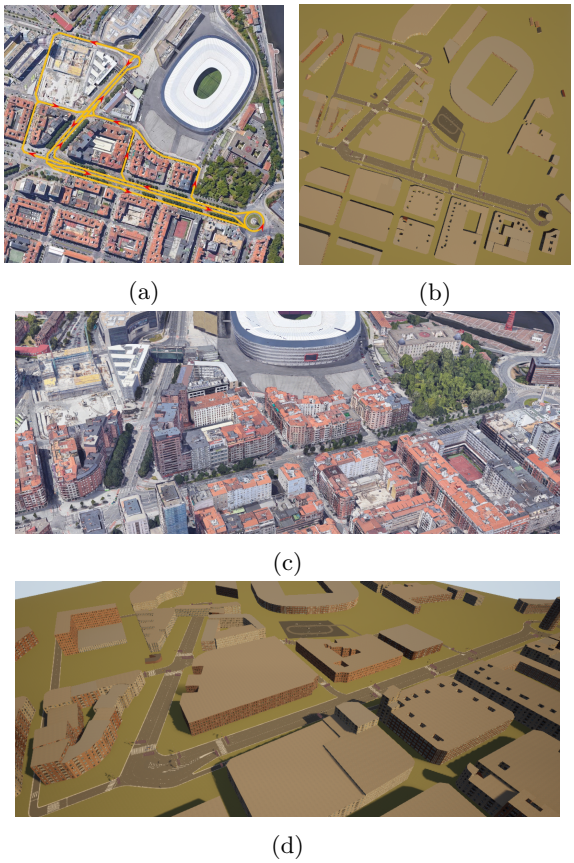

Figura 3: Comparativa del tramo de mapa entre la imagen hecha por satélite y la hecha en el simulador: (a) y (b) en vista aérea; (c) y (d) en perspectiva

a cabo utilizando uno de los modelos 3D que se pueden encontrar en la biblioteca de CARLA como plantilla, ya que facilita la integración con el modelo matemático que se utiliza para simular la dinámica vehicular. Una vez en Unreal Engine, se han añadido animaciones a las ruedas relacionando la velocidad en cada instante del vehículo con su giro. El resultado se muestra en la Figura 4, la cual compara una imagen del vehículo real con el modelado dentro del simulador.

En cuanto al modelado dinámico del vehículo, se hace uso de PhysX, un modelo multicuerpo altamente parametrizado desarrollado por Nvidia. Dado que se busca una representación del vehículo real se han establecido los parámetros en base a las especificaciones del Twizy (Tabla 1).

donde I_z es el momento de inercia, m_{chasis} y m_{ruedas} la masa del vehículo y de cada rueda respectivamente, K_{susp} y C_{susp} la rigidez y amortiguación de suspensión frontal/trasera, δ_{max} el giro máximo de las ruedas, C_{freno_max} el par máximo de frenada, rpm_{max} las revoluciones máximas del motor, $\mu_{transmi}$ la relación de transmisión y C_{drag} el coeficiente de arrastre.

https://doi.org/10.17979/spudc.9788497498043.435

(a)

(b)

Figura 4: Comparativa entre un Renault Twizy real (a) y el modelo virtual en Unreal Engine (b)

2.4 Controlador

Se han implementado dos controladores desacoplados en el entorno de simulación para resolver independientemente el problema de seguimiento de una velocidad de referencia (controlador longitudinal) y el problema de seguimiento de trayectoria (controlador lateral). Ambos controladores son parte de las funcionalidades básicas de un entorno de simulación y pueden ser fácilmente reemplazables por módulos externos utilizando ROS. Es por eso que se han elegido diseños basados en el estado del arte. A continuación, se expone una descripción más detallada de los mismos.

Parámetro	Unidad	Valor
I_z	kg m^2	243, 18
m_{chasis}	kg	611
m_{ruedas}	kg	8
K_{susp_front}	N/m	5840
K_{susp_rear}	N/m	8100
C_{susp_front}	Ns/m	660
C_{susp_rear}	Ns/m	1400
δ_{max}	rad	0, 7
C_{freno_max}	Nm	360
rpm_{max}	rpm	2100
$\mu_{transmi}$		0, 108
C_{drag}		0, 64

Tabla 1: Parámetros dinámicos principales del Renault Twizy

2.4.1 Controlador longitudinal

Para conseguir las señales de control del acelerador y el freno del vehículo se ha implementado un controlador PID de rango partido, utilizando los valores negativos en el freno y los positivos en el acelerador:

$$u = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \quad (1)$$

donde K_p, K_i y K_d son las ponderaciones proporcional, integral y derivativa respectivamente, es decir, los valores a ajustar del controlador, y $e(t)$ es la diferencia entre la velocidad longitudinal del vehículo y la de referencia establecida. $u(t)$ es la señal de control que se aplicará al vehículo más tarde.

Es posible conseguir un vector de velocidad de cada actor dentro del simulador gracias a la API de CARLA. Dicho vector corresponde a los valores de la velocidad desglosado en los componentes cartesianos y relativos al sistema de referencia global del mapa con el que se esté trabajando. De esta forma, la velocidad total del vehículo será el módulo del vector y la velocidad longitudinal será su proyección sobre el eje longitudinal del vehículo.

$$v_{long} = v_{tot} cos(\phi_v - \phi_{veh}) \quad (2)$$

donde ϕ_v es la orientación del vector de velocidad y ϕ_{veh} es la orientación del vehículo en el mapa.

Dada la superior capacidad de frenado a la de aceleración de los vehículos se ha establecido una ponderación a la acción de control del freno de $0, 4$.

2.4.2 Controlador lateral

El objetivo del controlador lateral es el de generar un valor de giro de ruedas para que el vehículo pueda seguir con el menor error posible la referencia marcada por el centro de la carretera. Para ello, se ha implementado un controlador MPC. El modelo utilizado para la predicción es el modelo cinemático o modelo Ackerman representado por el sistema de ecuaciones 3:

$$\begin{aligned} x_{k+1} &= x_k + v_k cos(\phi_k) T_s \\ y_{k+1} &= y_k + v_k sen(\phi_k) T_s \\ \phi_{k+1} &= \phi_k + \frac{v_k}{L} tan(\delta_k) T_s \end{aligned} \quad (3)$$

donde $\dot{x}$ y $\dot{y}$ son la velocidad longitudinal y lateral del eje trasero del vehículo respectivamente, v es la velocidad total del eje trasero, ϕ es la orientación del vehículo respecto al sistema de referencia global, L es la batalla del vehículo y δ

https://doi.org/10.17979/spudc.9788497498043.435

es el ángulo de giro de las ruedas representadas en el centro del eje delantero. dt es el tiempo de muestreo utilizado en las simulaciones.

Por lo tanto, se trata de un MPC no lineal cuya ley de control consiste en resolver en cada instante de control discreto k el siguiente problema de optimización,

$$\begin{array}{rcl} V & = & \min_{\Delta u^+} \; J(\Delta u^+, \hat{x}(k)) \\ & & s.a. \\ & & A_c \, u^+ \leq b_c \end{array} \tag{4}$$

donde $\Delta u^+ = \left[\begin{array}{cccc} \Delta\delta(k) & \dots & \Delta\delta(k+N) \end{array} \right]^T$ es la secuencia de variaciones en la acción de control a calcular a lo largo del horizonte de predicción $H = N\,T_s$, siendo T_s el tiempo de muestreo del controlador; $\hat{x}(k)$ es el estado medido en el instante k; y J es la ecuación de coste a minimizar,

$$J = (\hat{y} - w)^T \, Q \, (\hat{y} - w) + \Delta u^{+T} R \Delta u^+ \tag{5}$$

donde $\hat{y} = \left[\begin{array}{cccc} y(k+1) & \dots & y(k+N+1) \end{array} \right]^T$ es la posición lateral relativa al sistema de referencia del vehículo en cada iteración, y w la referencia a lo largo del horizonte de predicción H transformada al mismo sistema de referencia. Las matrices Q y R ponderan el error de seguimiento y la acción de control respectivamente.

Puesto que el modelo está representado en el sistema de coordenadas locales del vehículo, en cada instante las condiciones iniciales del problema de optimización son nulas.

Por último, se han implementado restricciones de entrada en el controlador, de acuerdo con las limitaciones físicas del vehículo, en el que las ruedas no pueden superar un ángulo de giro de 0,7 rad.

$$-0,7 \leq u \leq 0,7 \tag{6}$$

3 Caso de uso

En esta sección, se ha diseñado un caso de uso con el objetivo de comprobar la utilidad del entorno de simulación propuesto en situaciones básicas de conducción automática. Se ha realizado un caso de seguimiento de líneas en el que el vehículo debe recorrer parte del mapa sin salirse de la carretera.

Se ha realizado una selección heurística de los parámetros de los controladores, siendo los del controlador longitudinal los siguientes:

$$K_p = 1,0 \;\; ; \;\; K_i = 0,2 \;\; ; \;\; K_d = 0 \tag{7}$$

En cuanto a las ponderaciones del MPC del controlador lateral, los valores son los siguientes:

$$Q = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{N \mathrm{X} N} \tag{8}$$

$$R = \begin{bmatrix} 10 & 0 & \cdots & 0 \\ 0 & 10 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 10 \end{bmatrix}_{N-1 \mathrm{X} N-1} \tag{9}$$

Para finalizar se ha establecido un horizonte de predicción $H = 12$ y un tiempo de muestreo $T_s = 0.15$

La simulaciones se han llevado a cabo en un ordenador Dell Precision con una tarjeta gráfica NVIDIA Quaddro P1000 y un sistema operativo Windows 10.

3.1 Seguimiento de líneas

El caso de seguimiento de línea pone a prueba la propuesta de control integrada en el entorno de simulación para seguir la carretera a una velocidad constante de 30km/h. Se ha elegido esta velocidad como caso límite en entornos urbanos.

Para ello, primero es necesario generar la referencia que se va a utilizar. Esta referencia está definida en el centro del carril, ya que es el espacio más seguro para viajar de un vehículo. Como no se dispone de un algoritmo de percepción integrado en la plataforma, es necesario extraerla de los *waypoints* definidos intrínsecamente en las propias carreteras con la API de CARLA. Así mismo, es necesario ejecutar esta función de extracción de referencia en cada una de las iteraciones.

A fin de medir la calidad de los controladores implementados se recoge una serie de parámetros relativos al seguimiento de trayectorias, así como, el estado del vehículo en cada iteración y los errores lateral y longitudinal.

En la Figura 5 se presentan los resultados de la simulación realizada superpuestos con una imagen del mapa simulado. En el gráfico se pueden ver la referencia a seguir en azul con la trayectoria que ha recorrido el vehículo en gris. Es posible comprobar que el vehículo es capaz de realizar el recorrido sin salir de la carretera.

El error lateral, mostrado en la Figura 6, demuestra un crecimiento en las curvas, dándose el pico más alto en la de menor curvatura con un valor ab-

https://doi.org/10.17979/spudc.9788497498043.435

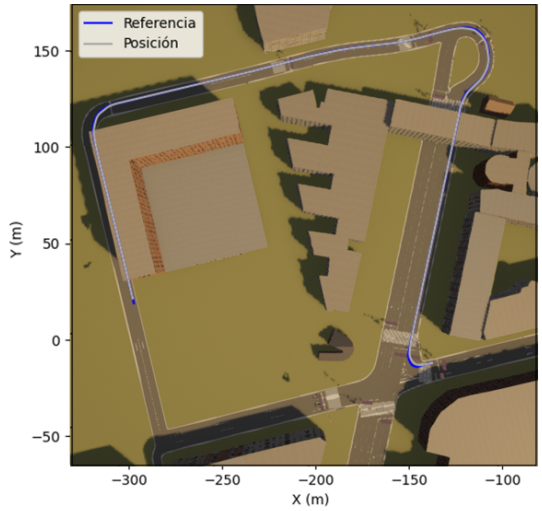

Figura 5: Resultado del seguimiento de trayectorias

soluto de $1m$. En este caso, los valores negativos representan un desviación lateral hacia la derecha de la referencia, mientras que los positivos son los desviados hacia la izquierda.

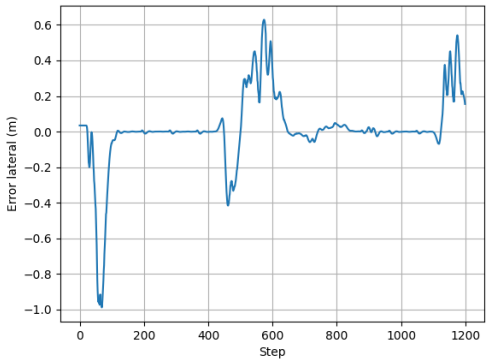

Figura 6: Error lateral en cada instante de la simulación

Por otro lado, el PID implementado para el seguimiento de velocidad consigue mantener una velocidad constante a lo largo de la mayoría del trayecto, a pesar de contar con error estático, como se observa en la Figura 7.

4 Conclusiones

En este trabajo se presenta un entorno de simulación basado en CARLA, en el que se introduce un mapa de la ciudad de Bilbao y el modelo dinámico de un Renault Twizy como punto de partida para futuras validaciones de algoritmos de vehículos automatizados. Dicho entorno es puesto a prueba mediante un caso de uso de seguimiento

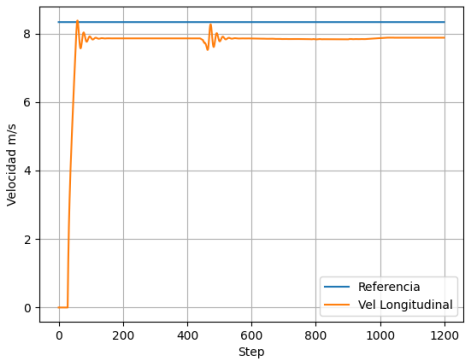

Figura 7: Seguimiento de la velocidad en cada instante

de línea con la integración de controladores independientes longitudinal y lateral que demuestran resultados aceptables.

Como trabajos futuros, se estima la implementación de sensores en CARLA para la generación de la trayectoria vehicular en casos de tráfico urbano, así como el testeo de diferentes algoritmos de aparcamiento, control cooperativo, evasión de obstáculos y toma de decisiones ante contingencias.

Se establece así una base para la futura simulación de situaciones de tráfico en entornos urbanos que serán posible validarlas con modelos reales.

Agradecimientos

Este trabajo ha sido financiado por el proyecto del Gobierno Vasco ELKARTEK KK-2021/00123 y el programa de Formación de Personal Investiador de la UPV/EHU y Tecnalia 2019.

References

[1] Sofiane Bacha et al. “A review on vehicle modeling and control technics used for autonomous vehicle path following”. In: *International Conference on Green Energy and Conversion Systems, GECS 2017* (2017). DOI: 10.1109/GECS.2017.8066221.

[2] R.F. Benekohal and Joseph Treiterer. “CARSIM: CAR-following model for SIMulation of traffic in normal and stop-and-go conditions”. en. In: *Transportation research record* 1194 (1988), pp. 99–111. ISSN: 0361-1981. URL: http://dx.doi.org/.

[3] cognata. *cognata*. https://www.cognata.com/. Online; accessed 12 April 2021. 2021.

[4] Alexey Dosovitskiy et al. “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the 1st Annual Conference on*

https://doi.org/10.17979/spudc.9788497498043.435

Robot Learning. Ed. by Sergey Levine, Vincent Vanhoucke, and Ken Goldberg. Vol. 78. Proceedings of Machine Learning Research. PMLR, Nov. 2017, pp. 1–16. URL: http://proceedings.mlr.press/v78/dosovitskiy17a.html.

[5] *Drive Constelation Nvidia*. https://www.nvidia.com/en-us/self-driving-cars/simulation/. Online; accessed 09 June 2021.

[6] dSpace. *dSpace*. https://www.dspace.com/en/inc/home/products/sw/automotive_simulation_models.cfm. Online; accessed 12 April 2021. 2021.

[7] Metamoto. *Metamoto*. https://www.metamoto.com/. Online; accessed 12 April 2021. 2021.

[8] rFactor. *rFactor Pro*. https://www.rfpro.com/rfactor-pro-and-concurrent-team-up/. Online; accessed 12 April 2021. 2014.

[9] SAE. *SAE levels of driving automation*. https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic. Online; accessed 13 December 2020. 2019.

[10] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. "Planning and Decision-Making for Autonomous Vehicles". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1.1 (2018), pp. 187–210. ISSN: 2573-5144. DOI: 10.1146/annurev-control-060117-105157.

[11] Soheil Sohrabi et al. "Quantifying the automated vehicle safety performance: A scoping review of the literature, evaluation of methods, and directions for future research". In: *Accident Analysis and Prevention* 152.February (2021), p. 106003. ISSN: 00014575. DOI: 10.1016/j.aap.2021.106003. URL: https://doi.org/10.1016/j.aap.2021.106003.

[12] Jessica Van Brummelen et al. "Autonomous vehicle perception: The technology of today and tomorrow". In: *Transportation Research Part C: Emerging Technologies* 89.March (2018), pp. 384–406. ISSN: 0968090X. DOI: 10.1016/j.trc.2018.02.012. URL: https://doi.org/10.1016/j.trc.2018.02.012.

https://doi.org/10.17979/spudc.9788497498043.435