DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI
ISSN: 2007-6363

Publicación Semestral Pädi Vol. 10 No. Especial 4 (2022) 145–150

Maniobra de rebase para vehículos autónomos usando una red neuronal convolucional
Passing maneuver for autonomous vehicles using a convolutional neural network

R.S. Arellano-Aguilar [a,*], O. González-Miranda [b], J.M. Ibarra-Zannatha [b]

[a] *UPIITA-IPN, 07340, CDMX, México.*
[b] *Departamento de Control Automático, CINVESTAV-IPN, 07360, CDMX, México.*

Resumen

El desarrollo de vehículos autónomos presenta una serie desafíos que, durante los últimos años, tanto la Academia como la Industria se han centrado en resolver. Entre las problemáticas de mayor interés se tienen las siguientes: La planificación de caminos, la generación de trayectorias, control del comportamiento y el seguimiento del camino. De ellas, la que presenta un mayor reto es el control del comportamiewnto, pues está encargado de realizar las maniobras que le permitan seguir con la ruta planeada a pesar de las contingencias de tráfico, por ejemplo: el rebase de otros vehículos, las maniobras de estacionamiento, la detección de las señales de tránsito, etc. En este trabajo nos interesamos al uso de técnicas metaheurísticas que contribuyan al control del comportamiento de un vehículo autónomo en las tareas de rebase de vehículos en carretera con base en la información del Lidar (radar láser). Entre los trabajos más destacados en los que podemos apoyarnos están los desarrollados por la empresa Nvidia, como el de *Bojarski et al* (Bojarski et al., 2016) en el cual se diseñó una red neuronal convolucional llamada *PilotNet* capaz de tomar las imágenes capturadas por la cámara y generar el control lateral del vehículo para mantenerlo conduciendo en el centro de su carril. En este trabajo se modificó la arquitectura original de la red *PilotNet* para que también pudiera procesar la información de un sensor LIDAR 2D montado en el auto y se entrenó para que, además de mantenerse conduciendo en su carril, también realizara la maniobra de rebase de otros vehículos tanto estáticos como en movimiento. El conjunto de datos de entrenamiento fue generado en el simulador de Gazebo con ejemplos de la maniobra de rebase realizada tanto por una persona como por un controlador PD con realimentación visual. Se comparó el desempeño del controlador PD con el de la red neuronal y se encontró que la red tiene un comportamiento más similar al que tendría una persona y es más rápida en realizar la maniobra. Además, no requiere de un sistema de toma de decisiones como el del controlador PD al cambiar de un carril a otro.

Palabras Clave: Vehículos Autónomos, Visión por computadora, Lidar 2D, RNC

Abstract

The development of autonomous vehicles has a lot of challenges that in recent years, both the Academy and the Industry have been trying to solve. Some of them are: The following of the road, passing of vehicles, the parking maneuvers, traffic signs detection, the creation of decision-making systems, etc. One of the most outstanding works is (Bojarski et al., 2016) in which a convolutional neural network called ”PilotNet” is designed. It can take images captured by a camera and generate the lateral control of a vehicle to keep it driving in the center of its lane. In this work, PilotNet’s network architecture was modified to process the data of a 2D LIDAR sensor mounted on the car and it was trained to carry out the passing maneuver from other vehicles both stationary and in motion, also the lane-keeping. The training data set was generated in the Gazebo simulator, with examples of the passing maneuver performed by both a person and a visual feedback PD controller. PD controller’s performance was compared versus neural network and it was found that the network has a behavior more similar to that of a person and is faster in performing the maneuver. In addition, it does not require a decision-making system like the PD controller when changing from one lane to another.

Keywords: Autonomous vehicles, Computer vision, Lidar 2D, CNN

*Autor para correspondencia: a.a.r.samuel99@gmail.com
Correo electrónico: a.a.r.samuel99@gmail.com (Rudyard Samuel Arellano-Aguilar), ogonzalez@ctrl.cinvestav.mx (Oscar González-Miranda),jibarra@cinvestav.mx (Juan Manuel Ibarra-Zannatha)

Historial del manuscrito: recibido el 20/05/2022, última versión-revisada recibida el 18/07/2022, aceptado el 09/08/2022 publicado el 05/10/2022
DOI: https://doi.org/10.29057/icbi.v10iEspecial4.9333

1. Introducción

El problema de la conducción autónoma es tan complejo que se suele separar en varios más sencillos como: Las maniobras de estacionamiento, el reconocimiento de señales de tránsito, el desarrollo de sistemas de toma de desiciones, el rebase de otros vehículos, la conducción dentro de un carril, etc. Respecto a la conducción en carretera manteniendo el carril existen numerosos trabajos con diversas metodologías. Por ejemplo, la universidad de Stanford desarrolló el conocido método *Stanley* Thrun et al. (2006) el cual es un método geométrico con el que se mide la pose del vehículo con respecto a la carretera y se usa esa información para controlar la dirección del auto. Cabe resaltar que este es el controlador de dirección que usaron en el DARPA Grand Challenge del 2005 para ganar la competición.

Otra metodología muy recurrente para hacer el control lateral en vehículos autónomos son los controladores basados en el modelo dinámico de la bicicleta mencionado en Rajamani (2011). Con este se han propuesto controladores PID como en Ackermann et al. (1995) para conducir un autobús 0305 a una velocidad máxima de 20 m/s; ó controladores predictivos como en Carvalho et al. (2013) donde se presentan resultados experimentales favorables al conducir un automóvil Jaguar Type-S a 80 Km/h; el cual puede evitar obstáculos estáticos y rebasar obstáculos en movimiento a 36 Km/h; en un camino con nieve.

En paralelo, se han desarrollado técnicas metaheurísticas las cuales no requieren de un modelo matemático para definir una ley de control; en su lugar se captura la experiencia de algún piloto experto y se usa dicha experiencia para entrenar algún modelo conexionista. A este tipo de arquitecturas se les conoce como "*End-to-end learning*" y se usan para mapear las señales que vienen de los sensores a las señales de control, que en este caso son la velocidad y el ángulo de dirección.

De las metodologías de este tipo una de las más destcables es la propuesta por Bojarski et al. (2016, 2017). En su trabajo utilizan una red neuronal convolucional (RNC) para procesar las imágenes capturadas por una cámara a bordo y así obtener el ángulo de dirección del vehículo. La red ya entrenada se implementó en el vehículo Dave 2 y pudieron conducir en carretera, en caminos húmedos o con nive e incluso en senderos sin señalización. Sin embargo, este método solo se limita a mantener el vehículo conduciéndose en su carril y no explora la realización de otras maniobras de conducción.

Desde entonces se han publicado numerosos trabajos en los que se emplean RNC para el "*End-to-end Learning*". Por ejemplo en el trabajo de Wu et al. (2019), se combina una RNC recurrente con un módulo de integración espaciotemporal multiescala, para procesar hasta 10 imágenes sucesivas del camino y así obtener el ángulo de dirección tanto el actual; como el de 4 iteraciones futuras. Esta red fue implementada en un vehículo autónomo real y probada en un ambiente urbano. Aunque en sus resultados se reporta un excelente desempeño (un RMSE de 0.0544 *rad* comparándolo con el desempeño de un ser humano); La red solo hacía el control lateral del vehículo para mantenerlo en su carril a una velocidad relativamente baja (a una velocidad promedio de 5 km/h) y tampoco puede relizar alguna otra maniobra.

En contraste, en los trabajos de Codevilla et al. (2018) y Hubschneider et al. (2017) se presentan arquitecturas similares a la PilotNet de NVIDIA para llevar a cabo otras maniobras de conducción como la evasión de obstáculos o navegar en intersecciones. En Codevilla et al. (2018) la red propuesta se implementó en un vehículo a escala $1:5$ y se probó en un camino con numerosas intersecciones. Además de las imágenes de la cámara, su red tenía como entrada una sucesión de vectores los cuales le indicaban a la red hacia qué dirección debía girar en cada intersección; de ese modo el vehículo pudo moverse de una posición inicial hacia una meta. Por otro lado en el trabajo de Hubschneider et al. (2017) la red fue implementada en un vehículo real con tres cámaras en el frente; de modo que cada imagen pasaba por una sucesión de capas de convolución hasta que se concatenaban en la capa de aplanamiento. Ademas, en una de las capas densas se agregaron dos entradas adicionales correspondientes a las luces direccionales del auto. Con esto se codificaba la intención del conductor de cambiar de carril y así poder realizar la evasión de obstáculos y la navegación en estacionamientos. A pesar del buen desempeño de ambas redes ninguna de las dos hace la tarea de toma de desiciones para elegir entre mantenerse en el carril o evadir algún obstáculo.

En este trabajo proponemos una RNC para que el vehículo a escala AutoMiny lleve a cabo las maniobras de seguimiento del carril, evasión de obstáculos y rebase de vehículos en movimiento en un ambiente simulado. Esta red tiene como entradas las imágenes adquiridas por la cámara a bordo así como los datos del lidar 2D. Como salida se obtienen la velocidad del auto y el ángulo de dirección.

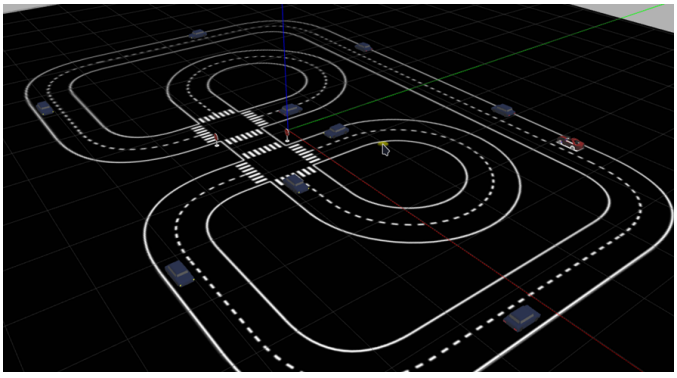

Figura 1: Entorno virtual en el que se recopilaron los datos manuales de la maniobra de rebase.

2. El vehículo AutoMiny

El vehículo autónomo utilizado en este trabajo es el AutoMiny, desarrollado por la Universidad Libre de Berlín Berlin (2020). Se trata de un vehículo a escala 1:10 el cual cuenta con: una cámara RGBD Intel Real Sense D435, un sensor inercial de 9 grados de libertad BOSCH BNO055, un sensor lidar RPLIDAR A2M8 de 2D y 360°, un motor sin escobillas de Faulhaber para darle tracción a las 4 ruedas, un servomotor Adafruit

para controlar la dirección del coche y una computadora Intel NUC8I5BEK con el sistema operativo Ubuntu 18.04 y el software de ROS Melodic instalado. Es importante señalar que este proyecto se ha realizado bajo ROS lo cual permite utilizar el modelo disponible en ROS en alguno de los simuladores que ofrece (rViz, Gazebo, etc.) o, indistintamente utlizar directamente el prototipo real. En este caso se utilizó el simulador desarrollado por el equipo Eagle Knights del Laboratorio de Robótica del ITAM ITAM (2019). Es importante señalar que las diferencias entre los experimentos realizados con el prototipo y los realizados en simulación suelen aparecer cuando se usa visión artificial. En el caso actual en que se utiliza el Lidar no hemos encontrado diferencias entre el simulador y el prototipo. Finalmente, comentamos que, para poder realizar maniobras de rebase se requiere de almenos un par de prototipos, pero como solamente contamos con uno, hemos decidido por el momento trabajar en simulación.

3. Diseño de la red neuronal convolucional

Se implementó una red neuronal basada en la PilotNet de Nvidia Bojarski et al. (2016, 2017), sin embargo, le incluimos una modificación importante en su arquitectura que permita incluir la información de un sensor LIDAR a fin de que esté en medidad de realizar maniobras de rebase.

3.1. Generación de datos de entrenamiento

Cabe mencionar que en nuestros primeros intentos de desarrollo de un controlador de comportamiento incluyeron un controlador visual que, en combinación con una máquina de estados, se encarga de la maniobra de rebase. Ahora, con la finalidad de entrenar la red, se recabaron datos sobre el comportamiento que se desea recrear con la RNC mtilizando dicho contraldor. Además, se recabó una seguna base de datos utilizando para ello los generados por un conductor experto.

3.1.1. Recolección de datos generados por un conductor experto

Para la generación de datos de entrenamiento, primero se teleoperó el automóvil en una pista con obstáculos estáticos, mientras se recoletaba la información de las variables a controlar (velocidad y dirección) y las entradas para la red neuronal (LIDAR e imágenes). La figura 1 muestra dicha pista. Se le dieron aproximadamente 4 vueltas a la pista, y se pudieron generar alrededor de 12,636 imágenes, cada una con sus respectivos valores de: dirección, velocidad y un vector con la información del sensor LIDAR.

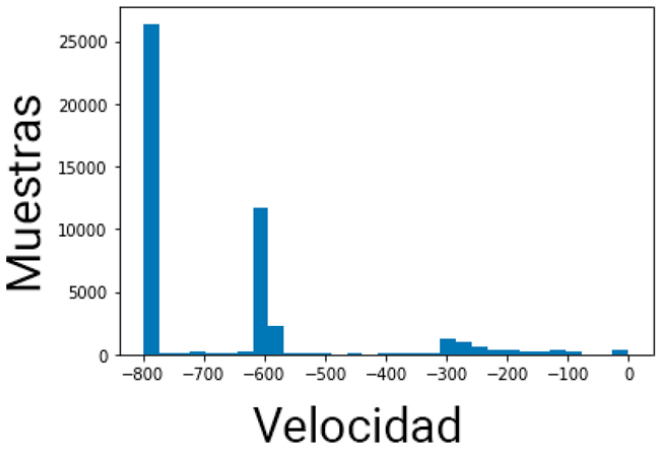

Figura 2: Histograma de las muestras de velocidad en los datos.

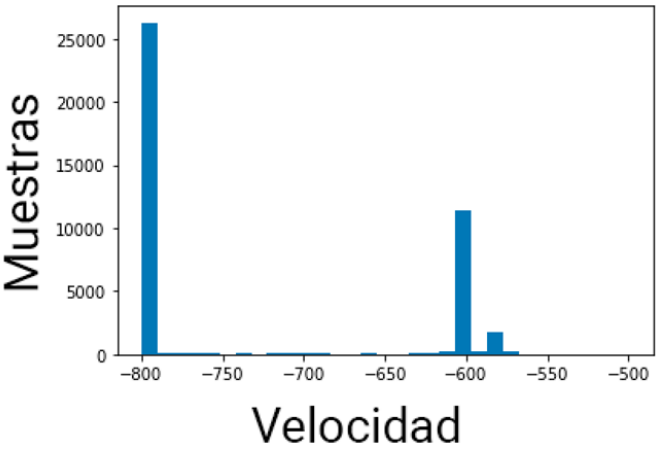

Figura 3: Histograma de las muestras de velocidad en los datos después de limpiar los datos.

3.1.2. Recolección de datos generados por un controlador con realimentación visual

Así mismo, la red neuronal se diseñó no solo para que sea capaz de rebasar objetos estáticos; también se entrenó para que pueda rebasar a otro auto en movimiento. Para recolectar estos datos, se simuló la maniobra de rebase variando las posiciones iniciales de ambos autos y se usó un controlador con realimentación visual programado para hacer esta tarea de forma autónoma. Así se generaron 4,633 imágenes más; con sus recpectivos valores de dirección, velocidad y vector de distancias medidas por el LIDAR.

3.1.3. Hardware y software utilizado

Debido a que el entrenamiento de redes neuronales suele ser computacionalmente costoso y a falta del hardware requerido, se utilizó el servicio de Google Colaboratory, el cual se puede usar de forma gratuita con ciertas restricciones y se puede usar una GPU de hasta 12GB de VRAM a través de Jupyter Notebooks y Python. Por esta razón se subió el dataset a Google Drive, desde donde se cargó para el entrenamiento.

3.2. Limpieza de los datos

Inicialmente, los datos de la velocidad del auto dibujaban el histograma de la figura 2. Estos se muestran en unidades

relativas y, por cómo funciona el simulador, toma valores negativos para que el vehículo avance y positivos, para que se mueva en reversa. Se observa que hay una pequeña concentración de datos en -300, esto es porque el controlador visual mide la velocidad del coche que va a rebasar antes de rebasarlo, este es un comportamiento que no tienen los pilotos humanos y el cual la red no pudo copiar; por esta razón se removieron esos datos, dejando únicamente los datos más representativos de la maniobra de rebase, que en este caso coinciden con aquellos datos donde la velocidad es más alta.

También, se mezclaron los datos de manera aleatoria para impedir que dicha información sea procesada de manera secuencial, evitandor así que la red pueda memorizar el conjunto de entrenamiento. El histograma de los datos de velocidad resultó como el que se muestra en la figura 3.

3.3. Arquitectura de la red

La arquitectura propuesta se basa en la de PilotNet, sin embargo, se incluyó una serie de capas convolucionales que permitan procesar las señales del LIDAR como si se tratase de una imagen. Para construir estas últimas cada vector de 360 elementos generado por el LIDAR se normalizó y se construyó un tensor de $20 \times 360 \times 3$ repitiendo el mismo renglón 20 veces y la misma capa de profundidad 3 veces. Estas pseudoimágenes se procesaron por separado en una segunda entrada de la red y posteriormente se unieron a los otros datos en la etapa de aplanamiento. Todas las capas de la red utilizan la función de activación "elu" y se añadió una función de relleno. El resto de la arquitectura se detalla en la tabla 1 y la figura 4 muestra cómo se concatenan las dos entradas. En rojo se encierran las secciones que se agregaron a la red PilotNet.

3.4. Entrenamiento de la red

La red fue entrenada por aproximadamente 3 épocas con un tamaño de lote de 1024, posteriormente, se "congelaron" las capas densas de la red, esto para entrenar únicamente las capas convolucionales durante otras 3 épocas, después se realizó el proceso inverso, se "congelaron" las capas convolucionales y se entrenaron únicamente las capas densas, esto permitió reducir el error drásticamente y se hicieron algunas pruebas en la pista. Posteriormente, para mejorar el desempeño de la red se volvió a entrenar esta vez con un tamaño de lote de 8 durante 4 épocas. En esta etapa el error se redujo aún más, se detuvo el entrenamiento de la red y se procedió a hacer pruebas. La gráfica de la figura 5 muestra el desempeño de la red a lo largo del entrenamiento.

4. Resultados

Después del entrenamiento se realizaron pruebas en el simulador para evadir automóviles estacionados y rebasar coches en movimiento. Respecto a la esta última, el vehículo por rebasar se movía a 0.48 m/s; mientras el autoMiny se mantuvo moviéndose a 0.96 m/s. En ambas pruebas se utilizó la misma RNC y en ambos casos el vehículo llevó a cabo las maniobras sin problemas. Las figuras 6 y 7 muestran dos aspectos del ambiente simulado. A continuación, se compara el desempeño de la RNC, el conductor experto y el controlador con realimentación visual.

De la figura 8 se observa que tanto la RNC como el piloto son capaces de realizar la maniobra en casi la mitad del tiempo que el controlador visual. También se puede resaltar que, al inicio de la maniobra la red tiene un comportamiento mucho más parecido al del piloto; pero después, en la reincorporación al carril original, el comportamiento de la red es más parecido al controlador visual pues tiene un sobreimpulso muy parecido al final.

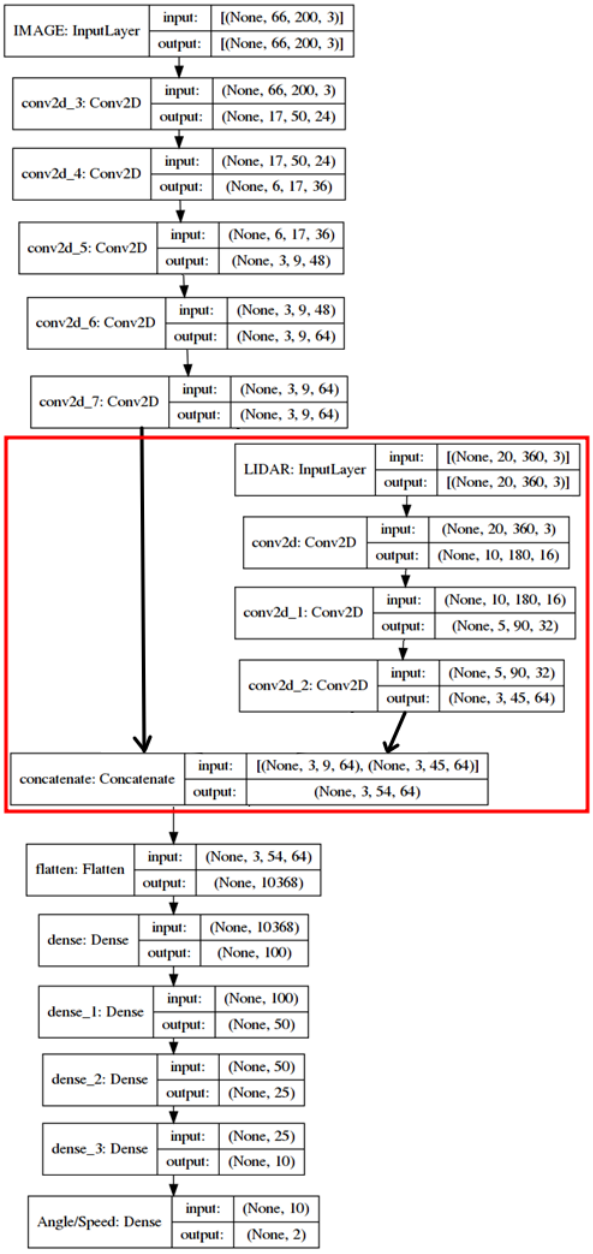

Figura 4: Arquitectura de la red neuronal usada.

Tabla 1: Arquitectura de la red

Imagen				Lidar			
Tipo de Capa	Tamaño Entrada	Tamaño Salida	Parametros [Filtros, Tamaño de kernel, Stride]	Tipo de Capa	Tamaño Entrada	Tamaño Salida	Parametros
Input Layer	[66,200,3]	[66,200,3]	NA	Input Layer	[20,360,3]	[20,360,3]	NA
Conv2D	[66,200,3]	[17,50,24]	24, (5, 5), (4,4)	Conv2D	[20,360,3]	[10,180,16]	16, (5, 5), (2,2)
Conv2D	[17,50,24]	[6,17,36]	36, (5, 5), (3,3)	Conv2D	[10,180,16]	[5,90,32]	32, (3, 3), (2,2)
Conv2D	[6,17,36]	[3,9,48]	48, (5, 5), (2,2)	Conv2D	[5,90,32]	[3,45,64]	64, (3, 3), (2,2)
Conv2D	[3,9,48]	[3,9,64]	64, (5, 5), (1,1)				
Conv2D	[3,9,64]	[3,9,64]	64, (3, 3), (1,1)				

Tipo de Capa	Tamaño de Entrada	Tamaño de Salida	Parámetros
Concatenate	[(3,9,64),(3,45,64)]	[3,54,64]	axis = 2
Flatten	[3,54,64]	[10368]	NA
Dense	[10368]	[100]	NA
Dense	[100]	[50]	NA
Dense	[50]	[25]	NA
Dense	[25]	[10]	NA
Dense	[10]	[2]	NA

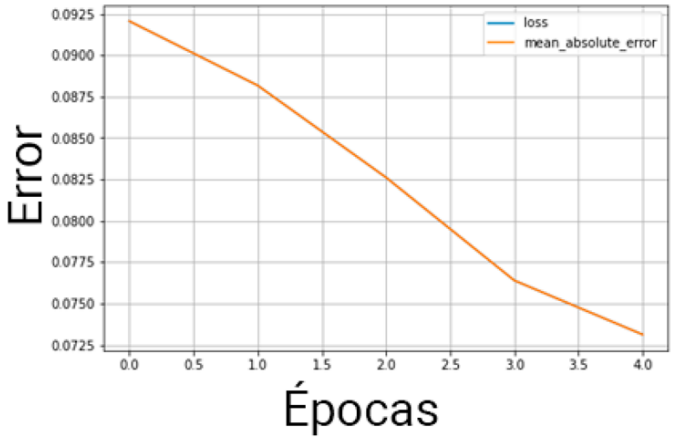

Figura 5: Entrenamiento de la red en la últimas 4 épocas.

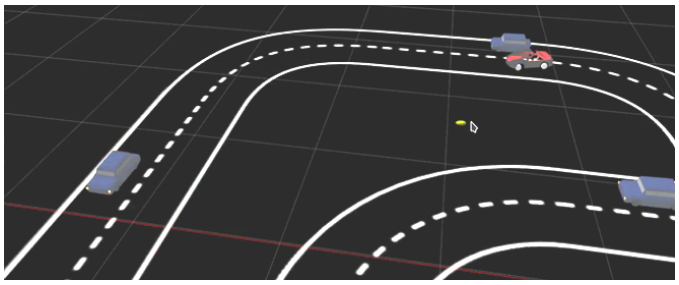

Figura 6: Red neuronal rebasando objetos estáticos.

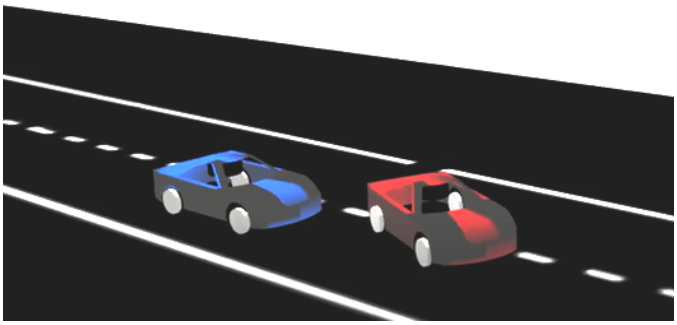

Figura 7: Red neuronal realizando la maniobra de rebase.

5. Conclusiones

Se observa que la naturaleza continua de las redes neuronales convolucionales le permite al auto conducir de manera similar a como lo haría un piloto humano únicamente basandose en la información de la cámara y el LIDAR. La red diseñada utilizó la red PiloNet de NVIDIA con una ingeniosa modificación que permite considerar la información de 360 vectores polares proporcionados por el Lidar; a nuestro conocimiento no hay en la literatura ejmplos como este. Aunque se logró realizar con éxito la evasión de autos estáticos y en movimiento; aún no esta claro si la red será capaz de realizar tareas más complejas. Es posible que, al incluir información temporal o algún tipo de memoria, la red pudiera llevar a cabo dichas tareas y extrapolar lo que ha aprendido a otros tipos de carreteras.

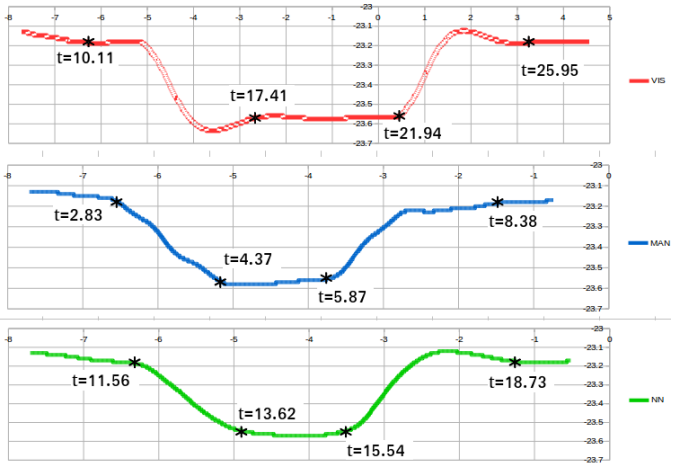

Figura 8: Comparación entre las trayectorias dibujadas por el vehículo al rebasar otro coche en movimiento mientras es controlado por: el controlador visual (en rojo), el piloto (en azul) y la RNC entrenada (en verde).

Referencias

Ackermann, J., Guldner, J., Sienel, W., Steinhauser, R., and Utkin, V. I. (1995). Linear and nonlinear controller design for robust automatic steering. *IEEE Transactions on control systems technology*, 3(1):132–143.

Berlin, F. U. (2020). Autominy An autonomous model car for education. `https://autominy.github.io/AutoMiny/`.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. (2016). End to end learning for self-driving cars. *CoRR*, abs/1604.07316.

Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L. D., and Muller, U. (2017). Explaining how a deep neural network trained with end-to-end learning steers a car. *CoRR*, abs/1704.07911.

Carvalho, A., Gao, Y., Gray, A., Tseng, H. E., and Borrelli, F. (2013). Predictive control of an autonomous ground vehicle using an iterative linearization approach. In *16th International IEEE conference on intelligent transportation systems (ITSC 2013)*, pages 2335–2340. IEEE.

Codevilla, F., Müller, M., López, A., Koltun, V., and Dosovitskiy, A. (2018). End-to-end driving via conditional imitation learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4693–4700. IEEE.

Hubschneider, C., Bauer, A., Weber, M., and Zöllner, J. M. (2017). Adding navigation to the equation: Turning decisions for end-to-end vehicle control. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE.

ITAM (2019). ITAM-Robotica Eagle Knights-Wiki. `https://github.com/ITAM-Robotica/Eagle_Knights-Wiki/wiki/Running-the-Simulation/`.

Rajamani, R. (2011). *Vehicle dynamics and control*. Springer Science & Business Media.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., et al. (2006). Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692.

Wu, T., Luo, A., Huang, R., Cheng, H., and Zhao, Y. (2019). End-to-end driving model for steering control of autonomous vehicles with future spatiotemporal features. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 950–955. IEEE.