Revista EIA

Enhancing mobile robot navigation: integrating reactive autonomy through deep learning and fuzzy behavior

Revista EIA
ISSN 1794-1237
e-ISSN 2463-0950
Año XIX/ Volumen 21/ Edición N.42
Julio - diciembre de 2024
Reia4229 pp. 1-14

Publicación científica semestral
Universidad EIA, Envigado, Colombia

Julián López-Velásquez, M. Sc.[1]
Gustavo A. Acosta-Amaya, Ph. D.[1]
Jovani A. Jiménez-Builes, Ph. D.[2]

1. Politécnico Colombiano Jaime Isaza Cadavid, Colombia
2. Universidad Nacional de Colombia

Para citar este artículo / To reference this article /
López-Velásquez, J.; Acosta-Amaya, G. A.; Jiménez-Builes, J. A.
Enhancing mobile robot navigation: integrating reactive autonomy through deep learning and fuzzy behavior
Revista EIA, 21(42), Reia4229. pp. 1-14.
https://doi.org/10.24050/reia.v21i42.1764

Autor de correspondencia:
Jovani A. Jiménez-Builes, Ph. D.
Licenciado en Docencia de Computadores
Universidad Nacional de Colombia
jajimen1@unal.edu.co

Recibido: 24-02-2024
Aceptado: 27-05-2024
Disponible online: 01-07-2024

Abstract

Objective: This study aimed to develop a control architecture for reactive autonomous navigation of a mobile robot by integrating Deep Learning techniques and fuzzy behaviors based on traffic signal recognition. **Materials**: The research utilized transfer learning with the Inception V3 network as a base for training a neural network to identify traffic signals. The experiments were conducted using a Donkey-Car, an Ackermann-steering-type open-source mobile robot, with inherent computational limitations. **Results**: The implementation of the transfer learning technique yielded a satisfactory result, achieving a high accuracy of 96.2% in identifying traffic signals. However, challenges were encountered due to delays in frames per second (FPS) during testing tracks, attributed to the Raspberry Pi's limited computational capacity. **Conclusions**: By combining Deep Learning and fuzzy behaviors, the study demonstrated the effectiveness of the control architecture in enhancing the robot's autonomous navigation capabilities. The integration of pre-trained models and fuzzy logic provided adaptability and responsiveness to dynamic traffic scenarios. Future research could focus on optimizing system parameters and exploring applications in more complex environments to further advance autonomous robotics and artificial intelligence technologies.

Keywords: Autonomous Navigation, Deep Learning, Fuzzy Behaviors, Control Architecture, Neural Networks, Artificial Intelligence.

Navegación autónoma reactiva de un robot móvil basada en aprendizaje profundo y comportamientos difusos

Resumen

Objetivo: este estudio tuvo como objetivo desarrollar una arquitectura de control para la navegación autónoma reactiva de un robot móvil mediante la integración de técnicas de Deep Learning y comportamientos difusos basados en el reconocimiento de señales de tráfico. **Materiales**: la investigación utilizó transfer learning con la red Inception V3 como base para entrenar una red neuronal en la identificación de señales de tráfico. Los experimentos se llevaron a cabo utilizando un Donkey-Car, un robot móvil de código abierto tipo Ackermann, con limitaciones computacionales inherentes. **Resultados**: la implementación de la técnica de transfer learning arrojó un resultado satisfactorio, logrando una alta precisión del 96.2% en la identificación de señales de tráfico. No obstante, se encontraron desafíos debido a retrasos en los cuadros por segundo (FPS) durante las pruebas, atribuidos a la capacidad computacional limitada de la Raspberry Pi. **Conclusiones**: al combinar Deep Learning y comportamientos difusos, el estudio demostró la efectividad de la arquitectura de control en mejorar las capacidades de navegación autónoma del robot. La integración de modelos pre-entrenados y lógica difusa proporcionó adaptabilidad y capacidad de respuesta a escenarios de tráfico dinámicos. Investigaciones futuras podrían centrarse en optimizar los parámetros del sistema y explorar aplicaciones en entornos más complejos para avanzar aún más en las tecnologías de robótica autónoma e inteligencia artificial.

Palabras clave: navegación autónoma, aprendizaje profundo, comportamientos difusos, arquitectura de control, redes neuronales, inteligencia artificial.

1. Introduction

The term artificial intelligence (AI) is considered to have been originated for the first time at the Dartmouth conference in 1955 (McCarthy, *et al*., 1955). Since then, the approach of this technology has gone through several transitions over more than 60 years. It initially focused on general problem solving (GPS) (Newell, *et al*., 1958) without much success, leading to a winter on the subject, in part due to the 1973 Lighthill (1973) report, however, in 1982, the Japanese decided to commit to a highly ambitious project called "Fifth Generation Computer Systems" (FGCS) (Treleaven & Lima, 1982), which generated an impetus for a renew interest in AI on the western countries, marking the path for the United States and Europe to focus on the development of Intelligent Knowledge-Based Systems (IKBS) (Blacklock, 1986) also called "Expert systems".

Lastly, in the last decade, driven mainly by the exponential growth of Data and systems processing capabilities. Machine Learning (ML) and Deep Learning (DL) are considered highly relevant research topics, essentially due to the applications that have generated a wide impact on the world (Bengio, 2016; Soori, *et al*., 2023). In the last five years, society is on the cusp of a new and profound technological revolution: Autonomous driving of vehicles (Qian, *et al.*, 2024). It is still early to predict the impact on societies or industry (Sharifani & Amini, 2023); however, we will be able to witness immense changes.

The main objective of the present project is to structure a control architecture based on Deep Learning and fuzzy behaviors, taking advantage of the great variety of open source tools available on the web, along with the wonderful community behind it (Bachute & Subhedar, 2021; Vinolia, *et al*., 2023). The architecture is designed so the robot detects obstacles through image recognition using Convolutional Neural Networks (CNN) models, and right after the detection, based on a set of established fuzzy behaviors the robot will take a specific action (Afif, *et al*., 2020). The control architecture will be implemented in an Ackermann-steering-type open source mobile robot known as Donkey-Car (DonkeyCar, 2024), designed to apply autonomous driving in small-scale cars, therefore, there are limitations inherent to the computational capacity of the robot.

In order to achieve the objective, the project will be divided in four sections as follow: in section two a brief description of some general Deep Learning alternatives used in autonomous navigation will be given, selecting the one that adapts the most to the characteristics of the project, as well as the establishment of a set of fuzzy behaviors based on the recognition of traffic signals and subsequently both strategies shall be combined in a control architecture. Hardware and software tools used on the development of the project, and a description of the test circuit used for the experiments are discussed in section three. Results and conclusions are presented in section four.

2. Background: DL algorithms and fuzzy behaviors

2.1. DL Alternatives for autonomous navigation of a robot

The neural network algorithms for object detection available have to meet two main conditions to be implemented as a solution to this project. On one hand, it has to be lightweight, due to inherent limitations in the hardware, and on the other hand, it has to be compatible with the ROS architecture.

We will focus our attention in the following algorithms, first, "You Only Look Once (YOLO)" one of the most known options, which besides of being open source and work with only CPU and/or GPU, it has the main characteristic of using a single convolutional neural network (CNN) to predict which objects are present and delimit their location using selector boxes (Redmon, *et al*., 2016; Dahirou & Zheng, 2021) On the other hand, we have Inception V3 algorithm, based on (Szegedy, *et al*., 2016). It was designed with the main objective of solving the computational expense problem of its previous versions through a dimension reduction with 1x1 stacked convolutions. Like the previous one, it also relies on CNN and can be run with CPU and/or GPU.

Table 1. Characteristics of the algorithms. Source: Authors' own creation.

	YOLO V2	Inception V3
Framework	Darknet	TensorFlow PyTorch
Neural Network	CNN	CNN
Language	C++ Python	Python
ROS compatible?	Yes (Bjelonic, 2024)	Yes (OTL, 2024)

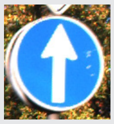
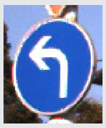
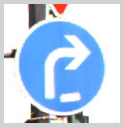
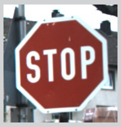
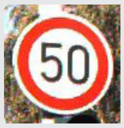
Despite the fact that both options previously mentioned comply with the condition of being compatible with ROS, the most noticeable difference took place during its execution, that means, once the algorithms were launched as ROS nodes, it was easy to read the bandwidth in the packets transmission and the publication rate, therefore, after a brief analysis, it was determined that ROS-Inception and YOLO presented a similar bandwidth transmission of the Raspicam node, which corresponds to the images, both similar to the 350 KB/s, however, the main difference took place during the node that identifies and labels the traffic signals, where ROS-Inception displayed a performance of 11.02 B/s transmission-wise compared to 201.11 B/s of YOLO, therefore, it is also theorized that due to Darknet framework higher resource consumption within the Raspberry Pi, there was a low performance of the frames per second (FPS), as consequence, generating a significant delay in the detection and classification of traffic signals, and hence, low performance in the completion of the fuzzy behaviors. Specific objectives of this project.

2.2. Fuzzy behaviors

The second part of the control architecture consists of the set of fuzzy behaviors based on the recognition of signals used in traffic, it means, depending on the traffic signal that the algorithm defined in the previous section identifies, a series of actions will be executed that will modify the direction and speed of movement of the robot (Kahraman, *et al*., 2020; Lin, *et al*., 2021).

The traffic signs, along with a brief description of the action the robot will carry out, are defined in table 2.

Table 2. Set of fuzzy behaviors. Source: Authors' adaptation.

Description	Image	Description	Image
The robot moves straight		The robot turns left	
The robot turns right		The robot slows down until it stops completely	
Linear velocity of the robot increases 50%			

2.3. Control architecture

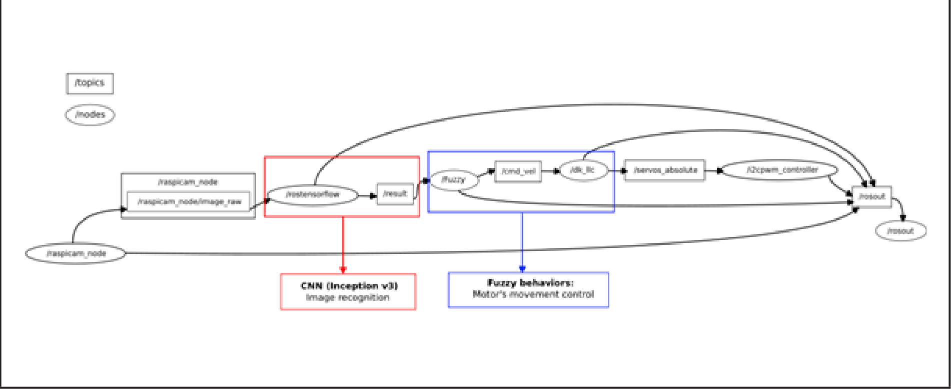
To represent the project's control architecture, ROS has a graphical interface tool known as rqt_graph, which allows you to visualize a ROS computing graph, which is equivalent to a ROS control architecture (Figure 1) (Mengoli, *et al*., 2021; ROS, 2024).

/raspicam_node. This node activates the Raspberry Pi camera and images are sent to the /rostensorflow node.

/rostensorflow. The trained CNN (Inception v3) processes the camera images and sends the results to the /Fuzzy node.

/Fuzzy. The node sends linear and/or angular movement commands to the motors, where they are received by the /dk_llc node.

Figure 1. Control architecture. Source: Authors' own creation.



3. Materials and methods

3.1. Robot Operating System (ROS)

It is an open-source robotics middleware. ROS provides a distributed programming environment, for both, real and simulated robots, as well as hardware abstraction, low-level control, message-passing between processes, software package management and tools and libraries for obtaining, building, writing, and running code across multiple computers. Nowadays, ROS has become the de-facto standard for robotics research (Itsuka, *et al*., 2022).

3.2. Donkey-car

It is an Ackermann-steering-type open source mobile robot designed to apply autonomous driving in small-scale cars, whose chassis is the 1/16th Exceed RC- magnet Truck, along with a single-board computer, the Raspberry Pi 3B, its 5MP camera and the PCA9685 board to control the servo motors though I2C communication.

Due to the hardware limitations within the embedded card, it was decided to use the operating system based on GNU/Linux, Ubuntu Mate 16.04.2 Armhf compatible with ROS1 Kinetic Kame.

3.3. Workstation

Lenovo Yoga 510 series with Intel Core i3-6100U CPU @ 2.30GHz x 4 processor, 7.7 GB of RAM and 114.5 GB of HDD. We used the workstation to train our own model by creating a virtual space with Anaconda with the following characteristics, TensorFlow 1.19.1 and Python 3.7

3.4. German Traffic Sign Recognition Benchmark (GTSRB)

It is a dataset that contains 43 classes of traffic signs, split into 39,209 training images and 12,630 test images. The images have varying light conditions and rich backgrounds (Stallkamp, *et al*., 2012). For the purpose of this project, only the classes mentioned in Table 2.2 are selected, which together contain 5339 elements.

3.5. ROS-Inception V3

The algorithm is based on CNN, compatible with ROS and can be configured through TensorFlow or PyTorch. To simplify the training time of the model, we will apply the transfer learning technique, which consists of reusing a model that has been previously trained with a large data set, and use it as a starting point for our own purpose (Transfer learning, 2024). For the purpose of this project, only the top layer will be retrained, leaving the others unchanged. Although it is not as good as training a system from scratch, it is quite effective for our purpose, and can be run from the PC, without the need for a GPU. Noteworthy, the images have been resized to 128x128x3.

4. Results and discussions

4.1 Experiments

For the validation experiments, two driving circuits were used, in which part of the traffic signals from Table 2 were incorporated. The first circuit shown in Figure 2 is a straight track of 10,25 meters. The second circuit in Figure 3 consists of a straight line of 4,25 meters, then a right turn and ends in a straight line of 3 meters.

Figure 2. Test track 1. Source: Authors' own creation.

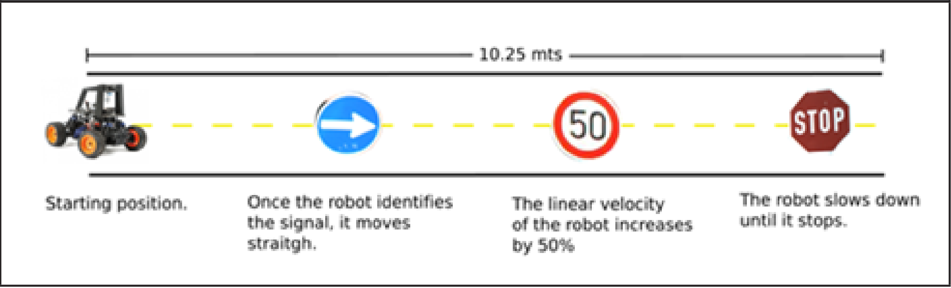
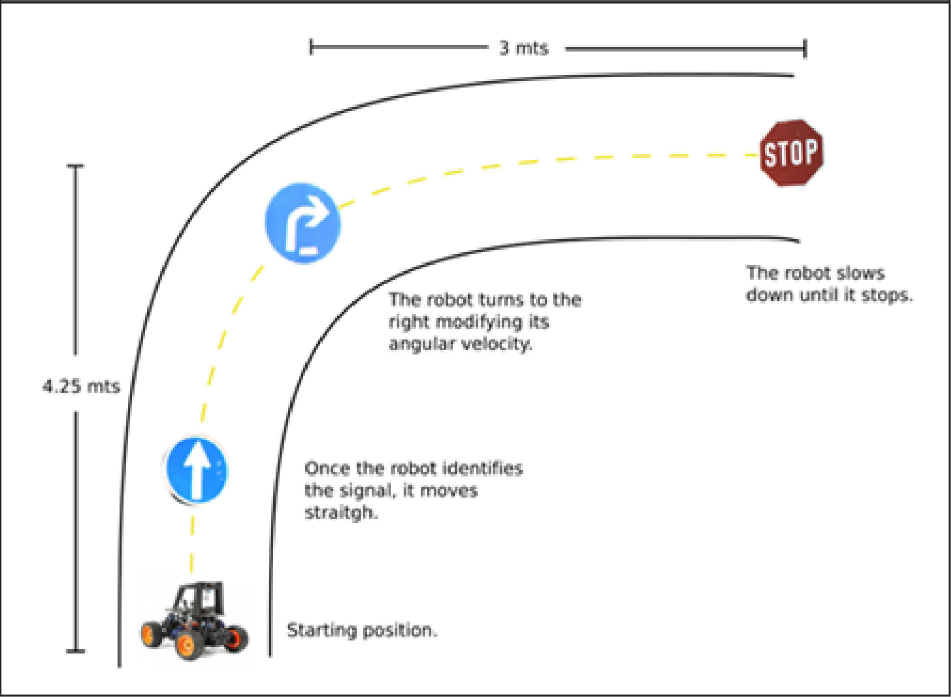

10.25 mts
50
STOP
Starting position.
Once the robot identifies the signal, it moves straitgh.
The linear velocity of the robot increases by 50%
The robot slows down until it stops.

Figure 3. Test track 2. Source: Authors' own creation.


3 mts
4.25 mts
STOP
The robot slows down until it stops.
The robot turns to the right modifying its angular velocity.
Once the robot identifies the signal, it moves straitgh.
Starting position.

4.2 Inception

We used the transfer learning technique in order to simplify the training time and get our own model, the concept consists in reusing a model that has been previously trained to only re-train the upper layer, leaving the rest unchanged. For the purpose of this project we used Inception V3 to try to identify the classes shown in Figure 4, and from GTSRB dataset a total of 5339 elements corresponding to the classes were used. The script includes a series of modifiable fields that made it possible to achieve an acceptable model. It was decided to keep all distortions disabled and to use the default arguments that may impact the training process, i,e.:

how_many_training_steps = 4000 learning_rate = 0,01

testing_percentage = 10% validation_percentage = 10%

Thanks to the above-mentioned, the accuracy the model reached after the training was of 96.2% as shown in Figure 5.

Figure 4. Classes. Source: Authors' own creation.

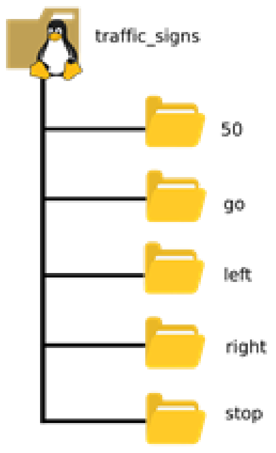
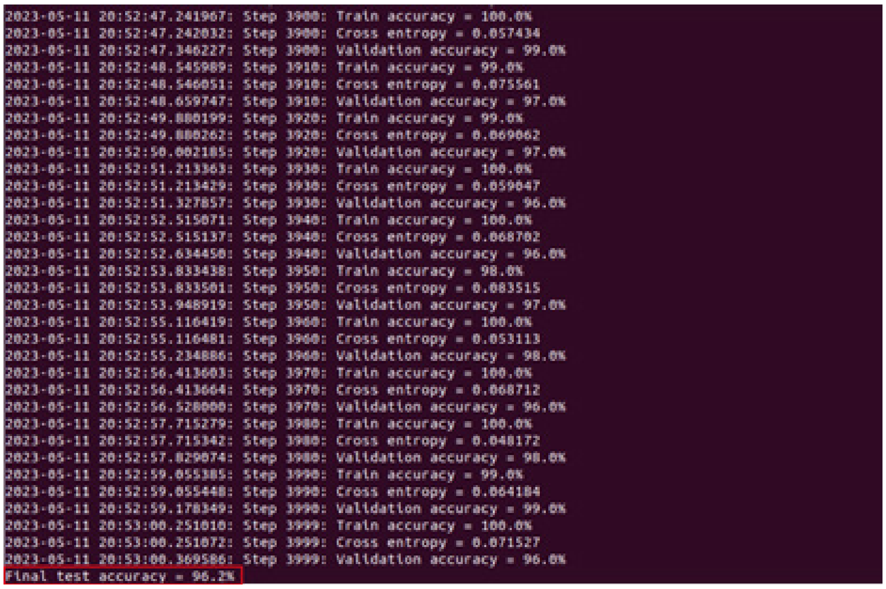


Figure 5. Training results. Source: Authors' own creation.



```
2023-05-11 20:52:47.241967: Step 3900: Train accuracy = 100.0%
2023-05-11 20:52:47.242032: Step 3900: Cross entropy = 0.057434
2023-05-11 20:52:47.346227: Step 3900: Validation accuracy = 99.0%
2023-05-11 20:52:48.545989: Step 3910: Train accuracy = 99.0%
2023-05-11 20:52:48.546051: Step 3910: Cross entropy = 0.075561
2023-05-11 20:52:48.659747: Step 3910: Validation accuracy = 97.0%
2023-05-11 20:52:49.880199: Step 3920: Train accuracy = 99.0%
2023-05-11 20:52:49.880262: Step 3920: Cross entropy = 0.069062
2023-05-11 20:52:50.002185: Step 3920: Validation accuracy = 97.0%
2023-05-11 20:52:51.213363: Step 3930: Train accuracy = 100.0%
2023-05-11 20:52:51.213429: Step 3930: Cross entropy = 0.059047
2023-05-11 20:52:51.327857: Step 3930: Validation accuracy = 96.0%
2023-05-11 20:52:52.515071: Step 3940: Train accuracy = 100.0%
2023-05-11 20:52:52.515137: Step 3940: Cross entropy = 0.068702
2023-05-11 20:52:52.634450: Step 3940: Validation accuracy = 96.0%
2023-05-11 20:52:53.833438: Step 3950: Train accuracy = 98.0%
2023-05-11 20:52:53.833501: Step 3950: Cross entropy = 0.083515
2023-05-11 20:52:53.948919: Step 3950: Validation accuracy = 97.0%
2023-05-11 20:52:55.116419: Step 3960: Train accuracy = 100.0%
2023-05-11 20:52:55.116481: Step 3960: Cross entropy = 0.053113
2023-05-11 20:52:55.234886: Step 3960: Validation accuracy = 98.0%
2023-05-11 20:52:56.413603: Step 3970: Train accuracy = 100.0%
2023-05-11 20:52:56.413664: Step 3970: Cross entropy = 0.068712
2023-05-11 20:52:56.528000: Step 3970: Validation accuracy = 96.0%
2023-05-11 20:52:57.715279: Step 3980: Train accuracy = 100.0%
2023-05-11 20:52:57.715342: Step 3980: Cross entropy = 0.048172
2023-05-11 20:52:57.829074: Step 3980: Validation accuracy = 98.0%
2023-05-11 20:52:59.055385: Step 3990: Train accuracy = 99.0%
2023-05-11 20:52:59.055448: Step 3990: Cross entropy = 0.064184
2023-05-11 20:52:59.178349: Step 3990: Validation accuracy = 99.0%
2023-05-11 20:53:00.251010: Step 3999: Train accuracy = 100.0%
2023-05-11 20:53:00.251072: Step 3999: Cross entropy = 0.071527
2023-05-11 20:53:00.369586: Step 3999: Validation accuracy = 96.0%
Final test accuracy = 96.2%
```

4.3 Donkey-Car

Circuit I showed the better performance when the robot identified the start and stop signal. Even though the circuit was successfully completed, sometimes a delay in the FPS was presented, causing that the fuzzy commands did not played their part on time, which was expected due to the limited computational capacity of the Raspberry Pi and data transmission through WiFi. There was a similar behavior during circuit II, in this case, the knot of the issue was presented when the car was trying to modify its angular movement. In both circuits the Donkey-Car was able to identify the traffic signals.

4.4 Final analysis

The research presented in this document underscores the importance of combining Deep Learning techniques and fuzzy behaviors to achieve reactive autonomous navigation in a mobile robot. By analyzing the evolution of artificial intelligence over the past decades, a significant shift towards more specialized and applied approaches is evident, such as the use of Convolutional Neural Networks (CNNs) for object detection in dynamic environments.

The implementation of a transfer learning model using Inception V3 has proven effective in identifying specific classes in a dataset of traffic signals. The model's accuracy of 96.2% after training highlights the efficacy of this strategy in adapting pre-trained models to specific tasks, thereby reducing the time and resources required for training.

Integrating fuzzy behaviors based on traffic signal recognition into the robot's control architecture provides an additional layer of adaptability and responsiveness to changing situations in the environment. This combination of deep learning techniques and fuzzy logic enables the robot to make informed and safe decisions in real-time, enhancing its capability for autonomous navigation in complex scenarios.

5. Conclusions

Due to the transfer learning technique and using the Inception V3 network as a base, a satisfactory result was obtained in the training of a neural network for the identification of traffic signals. Especially, this technique reduced considerably the training time and allowed training without the Nee of a GPU.

Due to the low computational capacity of the Raspberry Pi within the Donkey Car, during the tests tracks, sometimes there were delays in the FPS, therefore, a delay in the movements of the robot.

In conclusion, this study has demonstrated that integrating Deep Learning and fuzzy behaviors in an autonomous navigation system can significantly enhance a mobile robot's ability to interact safely and efficiently with its environment. Leveraging pre-trained models alongside transfer learning techniques has accelerated the training process and improved object detection accuracy.

Furthermore, incorporating fuzzy behaviors based on traffic signal recognition has proven to be an effective strategy for adapting the robot's behavior to specific traffic situations, ensuring smoother and safer navigation. These findings underscore the importance of combining different artificial intelligence approaches to develop robust and adaptable autonomous systems.

In the future, exploring the application of these techniques in more complex and dynamic environments, as well as optimizing key parameters to further enhance system performance, could be valuable. This research lays the groundwork for future advancements in the field of autonomous robotics and artificial intelligence, opening up new possibilities for the creation of intelligent and secure autonomous systems.

6. References

Afif, M., Ayachi, R., Said, Y., Pissaloux, E., & Atri, M. (2020). Indoor image recognition and classification via deep convolutional neural network. In *Proceedings of the 8th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT'18)*, vol. 1, pp. 364-371. Cham, Switzerland: Springer International Publishing.

Bachute, M., & Subhedar, J. (2021). Autonomous driving architectures: Insights of machine learning and deep learning algorithms. *Machine Learning with Applications*, vol. 6, 100164. Available at: https://doi.org/10.1016/j.mlwa.2021.100164.

Bengio, Y. (2016). Machines who learn. *Scientific American Magazine*, vol. 314(6), pp. 46–51. Available at: https://doi.org/10.1038/scientificamerican0616-46.

Bjelonic, M. (2024). Yolo v2 for ROS: Real-time object detection for ROS. Available online: https://github.com/leggedrobotics/darknet_ros/tree/feature/ros_separation (accessed on 17 May 2024).

Blacklock, P. (1986). Standards for programming practices: An alvey project investigates quality certification. *Data Processing*, vol. 28(10), pp. 522–528. Available at: https://doi.org/10.1016/0011-684X(86)90069-9.

Dahirou, Z., & Zheng, M. (2021). Motion detection and object detection: Yolo (You Only Look Once). In *2021 7th Annual International Conference on Network and Information Systems for Computers (ICNISC)*, pp. 250-257. New York, USA: IEEE.

DonkeyCar. (2024). How to build a Donkey. Available online: http://docs.donkeycar.com/guide/build_hardware/ (accessed on 17 May 2024).

Itsuka, T., Song, M., & Kawamura, A. (2022). Development of ROS2-TMS: New software platform for informationally structured environment. *Robomech J.*, vol. 9(1). Available at: https://doi.org/10.1186/s40648-021-00216-2.

Kahraman, C., Deveci, M., Boltürk, E., & Türk, S. (2020). Fuzzy controlled humanoid robots: A literature review. *Robotics and Autonomous Systems*, vol. 134, p. 103643. Available at: https://doi.org/10.1016/j.robot.2020.103643.

Lighthill, J. (1973). Artificial intelligence: A general survey. *The Lighthill Report*. Available at: http://dx.doi.org/10.1016/0004-3702(74)90016-2.

Lin, H., Han, Y., Cai, W., & Jin, B. (2022). Traffic signal optimization based on fuzzy control and differential evolution algorithm. *IEEE Transactions on Intelligent Transportation Systems*, vol. 1(4). Available at: https://doi.org/10.59890/ijetr.v1i4.1138.

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (1955). A proposal for the Dartmouth summer research project on artificial intelligence. *AI Magazine*, vol. 27(4), p. 12. Available at: https://doi.org/10.1609/aimag.v27i4.1904.

Mengoli, D., Tazzari, R., & Marconi, L. (2020). Autonomous robotic platform for precision orchard management: Architecture and software perspective. In *2020 IEEE International Workshop on Metrology for Agriculture and Forestry, MetroAgriFor*, pp. 303-308. New York, USA: IEEE.

Newell, A., Simon, H. A., & Shaw, J. C. (1958). Report on a general problem-solving program. Pittsburgh, Pennsylvania: Carnegie Institute of Technology, pp. 1-27. Available at: http://dx.doi.org/10.1016/0004-3702(74)90016-2.

OTL. (2024). ROS inception v3. *GitHub, Inc*. Available online: https://github.com/OTL/rostensorflow (accessed on 17 May 2024).

Qian, J., Zhang, L., Huang, Q., Liu, X., Xing, X., & Li, X. (2024). A self-driving solution for resource-constrained autonomous vehicles in parked areas. *High-Confidence Computing*, vol. 4(1), 100182. Available at: https://doi.org/10.1016/j.hcc.2023.100182.

Redmon, J., Santosh, D., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp. 779-788. Available at: https://doi.org/10.1109/CVPR.2016.91.

ROS. (2024). Ros rqt_graph. *Open Robotics*. Available online: http://wiki.ros.org/rqt_graph (accessed on 17 May 2024).

Sharifani, K., & Amini, M. (2023). Machine learning and deep learning: A review of methods and applications. *World Information Technology and Engineering Journal*, vol. 10(07), pp. 3897-3904. Available at: https://doi.org/10.4028/www.scientific.net/JERA.24.124.

Soori, M., Arezoo, B., & Dastres, R. (2023). Artificial intelligence, machine learning and deep learning in advanced robotics, a review. *Cognitive Robotics*, vol. 3, pp. 54-70. Available at: https://doi.org/10.1016/j.cogr.2023.04.001.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, vol. 32, pp. 323-332. Available at: https://doi.org/10.1016/j.neunet.2012.02.016.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp. 2818–2826. Available at: https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.308.

Transfer learning. (2024). Transfer learning. Available online: https://paperswithcode.com/task/transfer-learning (accessed on 17 May 2024).

Treleaven, P., & Lima, I. (1982). Japan's fifth generation computer systems. *Computer*, vol. 15(08), pp. 79–88. Available at: https://doi.org/10.1109/MC.1982.1654113.

Vinolia, A., Kanya, N., & Rajavarman, V. N. (2023). Machine learning and deep learning based intrusion detection in cloud environment: A review. In *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, IEEE, pp. 952-960. Available at: 10.1109/ICSSIT55814.2023.10060868.