Autonomous Lane Tracking for an Ackermann-steering Robot

José Luis Pasarin, Jorge Dueñas, Jonathan Montoya, Alejandro Cibrian, Ulises Orozco-Rosas, Kenia Picos*

CETYS Universidad, Av. CETYS Universidad No. 4, El Lago, C.P. 22210, Tijuana B.C., Mexico
{joseluis.pasarin, jorge.duenas, jonathan.montoya, acibrian}@cetys.edu.mx, {ulises.orozco, kenia.picos}@cetys.mx

Resumen

Este artículo presenta la implementación de un vehículo autónomo con configuración de dirección Ackermann utilizando un sistema de control PID para seguimiento de carril mediante procesamiento digital de imágenes. El robot de dirección Ackermann modifica su ángulo de rotación frontal en función de los datos de la cámara, en los que se aplican filtros HSV y de contorno para definir los límites de la pista y su centro. Se aplica un controlador PID para asegurar la incorporación del vehículo al carril en cada punto del carril, utilizando la función de control para calcular la respuesta al error y modificar la orientación actual en consecuencia, siendo el error la desviación del centro del carril. CoppeliaSim se utiliza para crear el entorno virtual del vehículo, proporcionando una plataforma de simulación realista y eficiente para probar y evaluar el rendimiento del sistema de seguimiento de carril propuesto. Además, se utiliza la programación en Python para implementar todas las funciones en la simulación, incluido el procesamiento de imágenes, los cálculos del sistema de control y la comunicación con el entorno virtual en CoppeliaSim. Se presentan resultados experimentales para evaluar la factibilidad de la implementación propuesta.

Palabras clave— Robot de dirección Ackermann, Vehículos autónomos, Controlador PID, Filtro HSV, Filtro de contorno

Abstract

This paper presents the implementation of an Ackermann-steering configuration autonomous vehicle using a PID control system for lane tracking through digital image processing. The Ackermann-steering robot modifies its front rotation angle based on camera data, on which HSV and contour filters are applied to define the limits of the track and its center. A PID controller is applied to ensure the vehicle's incorporation into the lane at every point on the road, using the control function to calculate the error response and modify the current orientation accordingly, with the error being the deviation from the lane's center. CoppeliaSim is used to create the virtual environment of the vehicle, providing a realistic and efficient simulation platform for testing, and evaluating the performance of the proposed lane tracking system. Additionally, Python programming is used to implement all the functions in the simulation, including image processing, control system calculations, and communication with the virtual environment in CoppeliaSim. Experimental results are presented to evaluate the feasibility of the proposed implementation.

Keywords— Ackermann-steering robot, Autonomous vehicles, PID controller, HSV filter, Contour filter

1. INTRODUCTION

For the past few years, Control Engineering has become quite relevant due to the great benefits that can be obtained after applying its principles in the development of projects. This is because it allows the development of devices that can obtain feedback from your system, thus reducing the errors obtained in the output [1].

An example of the varied applications in which the principles of Control Engineering are used is in the development of autonomous vehicles, in which, as indicated by its name, they are able to follow a trajectory autonomously to reach a destination [2], [3]. However, for the elaboration of these vehicles, it is necessary to have mastery of the tools that allow the development of the necessary technology for its operation, both hardware, and software [4].

In this work, a solution for this problem is sought by the implementation of a proportional-integral-derivative (PID) controller that receives information through a vision sensor and the output provides the rim steering of the system. To this end, the following objectives are proposed:

- Create an environment and system model virtually using the simulation software CoppeliaSim.
- Design a mobile robot with Ackermann configuration that is capable of performing lane following on a track autonomously using a perspective vision sensor via CoppeliaSim.
- Implement an algorithm that controls the direction of the robot, allowing the robot to move within the trajectory of a track created in the simulation environment. This algorithm consists of two phases, the first being an image processing phase, and the second a PID-type direction controller with feedback.

To test the virtual environment, algorithms must be integrated through the Python language that are implemented in the robot simulation platform called CoppeliaSim, in which the program will allow testing and correcting the automated system of the mobile robot [5].

2. FUNDAMENTALS

In this section, we present a general description of the concepts and algorithms employed in this work.

* *Corresponding author: ulises.orozco@cetys.mx*

2.1 Autonomous vehicle

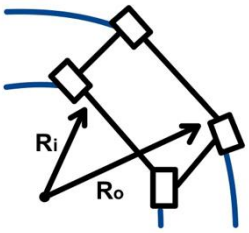
An autonomous vehicle is capable of moving without the need for human intervention [2]. That is to say, it does not need human action to intervene in the driving process as usual. To get to this point, it is necessary to implement hardware and software that allows the vehicle to detect its environment [6].

2.2 Ackermann configuration

In addition to hardware and software, other aspects are taken into account in autonomous vehicles, such as their kinematic model. This refers to how the wheels of the vehicle are configured so that it is possible to generate its movement [7].

One of the most used configurations in automobiles is the Ackermann type, which is based on the fact that the front wheels are the ones that control the direction, as seen in Figure 1. In this way, when making a turn, the inner wheel describes a greater angle than that of the outer wheel to avoid slipping.

Figure 1. Internal (R_i) and external (R_o) radios are described by the wheels of the vehicle with Ackermann configuration



Mathematically, the relationship between the angles of the inner and outer wheels can be expressed by Equation 1.

$$cot(\alpha_o) - cot(\alpha_i) = \frac{R+b}{l} - \frac{R-b}{l} = \frac{2b}{l} \quad (1)$$

In which α_o is the angle of the external wheel, α_i the internal wheel angle, R is the radius of instantaneous curvature of the robot path, b is the distance between the rear wheels, and l is the distance from the rear axle to the center of the front wheels.

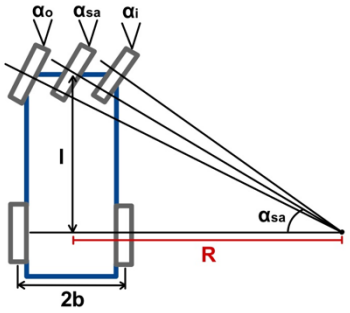
From a kinematic point of view, the Ackermann configuration is similar to the tricycle configuration, obtaining that the two front wheels of the Ackermann configuration are equivalent to the front wheel of the tricycle configuration, see Figure 2.

Mathematically, the relationship between the inner and outer wheel angles of the Ackermann configuration and the wheel angle of the tricycle configuration can be expressed by Equation 2 and 3.

$$cot(\alpha_{sa}) = cot(\alpha_i) + \frac{b}{l} \quad (2)$$

The mathematical relationship between the angle of the internal wheel of the Ackermann configuration and the angle of the wheel of the tricycle configuration can be expressed by Equation 2.

Figure 2. Kinematic model of the Ackermann configuration equivalent to the tricycle configuration



The mathematical relationship between the angle of the external wheel of the Ackermann configuration and the angle of the wheel of the tricycle configuration can be expressed by Equation 3.

$$cot(\alpha_{sa}) = cot(\alpha_o) - \frac{b}{l} \quad (3)$$

In which α_{sa} is the wheel angle of the tricycle configuration. In the same way, within the elements that make up an autonomous vehicle, the sensors are fundamentals.

2.3 Vision sensors

By definition, a sensor is a device capable of detecting the presence and measurement of a physical or chemical quantity [8]. These types of devices are often used in industry to transform instrumentation variables (measured properties) into electrical variables so that it is possible to work with them. Various types of sensors are capable of measuring variables such as temperature, force, speed, light intensity, and humidity, among other variables.

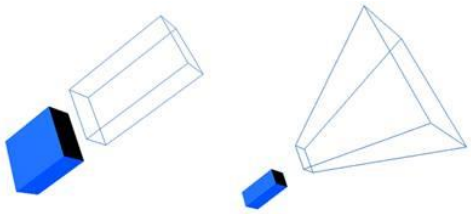
In the case of an autonomous vehicle, vision sensors are used since they are essential for controlling the direction and movement of this type of vehicle. A vision sensor is an element that is inside an electronic camera that is capable of detecting and capturing the information of the image observed by it [9]. Thus, it is responsible for converting light waves or electromagnetic radiation into electrical signals using a matrix of photodiodes or phototransistors, which are light-sensitive electronic components [8]. In this way, an autonomous vehicle can be able to observe its environment and thereby make a decision about its speed and displacement.

The categories of vision sensors can be divided into two types: the orthogonal projection sensor and the perspective projection sensor, see Figure 3. The first of them has a rectangular field of view, which is why they are widely used in short-range sensors such as infrared.

For this part, the perspective projection sensor is more similar to the human eye or a regular camera, since its field of

vision is trapezoidal, which is why they are used for camera sensors.

Figure 3. Projection of the orthogonal sensor (left) and the sensor in perspective (right)

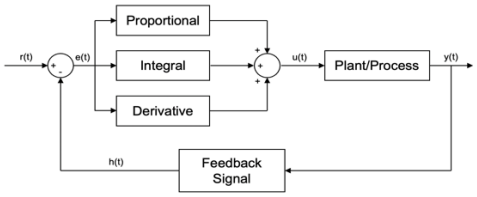


2.4 PID controller

In systems engineering, a control system is a set of devices that are responsible for ordering and regulating the behavior of a system so that it provides the desired response and eliminates errors [4].

One of the most used controllers in the industry due to its simplicity and reliability is the PID controller, this is a feedback control algorithm used in closed-loop systems, which, as its name says, is proportional, integral, and derivative, see Figure 4. This algorithm is based on three different parameters; its proportional value that depends on the current error, the integral that depends on the past error, and the derivative that predicts the future error [10]. These parameters are added, thus obtaining a result that is used to adjust the process to be controlled.

Figure 4. PID controller block diagram



The PID controller offers a fast response as a whole because it has a significant error signal compensation, however, it tends to oscillate, and due to this, it is important to adjust its parameters [11]. Equation 4 describes the PID controller output function.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de}{dt} \quad (4)$$

3. METHODOLOGY

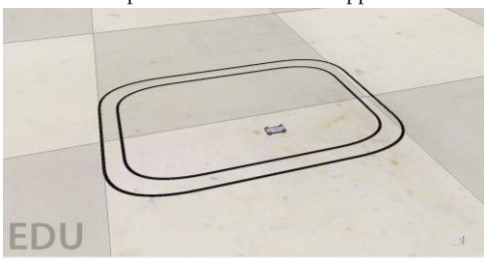
For the implementation of the proposed work, the Python programming language was employed. The program development process is detailed as follows.

3.1 Environment simulation implementation

For the environment simulation implementation, a CoppeliaSim project was created, in which inside the scene of the software a track made up of two lanes was designed.

The track was rectangular with circular edges, with two long straight lines and two shorter straight lines, for this, within the software two paths were created with the established measurements, in addition to adding black color and thickness to the generated lines, to facilitate the image processing that our robot would have, see Figure 5.

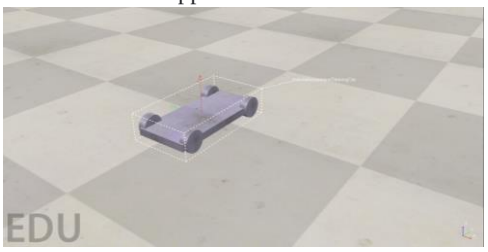
Figure 5. Track implementation in the CoppeliaSim simulator



3.2 Mobile robot implementation

Subsequently, the configuration of the mobile robot was carried out so that it could have manual control within the track. To do this, what was done was to use a model provided by CoppeliaSim, this being a vehicle with an Ackermann configuration, see Figure 6. This model was chosen because it already had the 3D design and the necessary programming for its manual operation, from rear-wheel drive to front-angle steering.

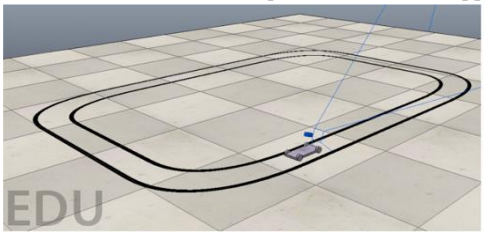
Figure 6. A vehicle with Ackermann configuration implemented in the CoppeliaSim simulator



3.3 Vision sensor in mobile robot implementation

Following this, the implementation of the vision sensor was carried out, which consists of a camera that is capable of processing the image obtained for lane tracking, see Figure 7.

Figure 7. A vehicle with camera implemented in the CoppeliaSim



In this work, we chose to use the perspective projection vision sensor because its field of vision (trapezoidal) proved to be the most suitable for achieving lane tracking, this vision being the one that most closely resembles the vision of a human eye driver.

For this, a sensor provided by CoppeliaSim was integrated, in which, once staged, the sensor only had to be

paired and located next to the mobile robot, declaring its location X, Y, and Z, as well as the angle of inclination that this would have to achieve a better projection of the track, see Figure 8.

Figure 8. Vision sensor projection in perspective

With this sensor, it is possible to obtain a two-dimensional image with a great range both near and far, observing both black lanes at all times.

3.4 Implementation of image processing algorithm.

To track the circuit through the rails, the image obtained by the vision sensor had to be processed. For this, it was necessary to connect the CoppeliaSim software with Python, which was the programming language used to perform the image processing. One of the tools used to achieve this was OpenCV, which is a Python library that provides tools to solve computer vision problems.

In this way, the first thing that was done was the connection of the Python IDE with CoppeliaSim through Code 1.

Code 1. Python connection with CoppeliaSim

```
####Conexión con CoppeliaSim###

#Finalizar todas las conexiones con el simulador
sim.simxFinish(-1)

#Iniciar una nueva conexión en el puerto 19999
(dirección por defecto)
clientID = sim.simxStart('127.0.0.1', 19999, True,
True, 5000, 5)

#Comprobar que se haya podido conectar con CoppeliaSim
if clientID != -1
    #Si clientID es diferente a -1, la conexión es
exitosa
    print('CONEXION ESTABLECIDA')
else:
    #En caso contrario, no se ha podido conectar
    sys.exit('ERROR AL CONECTARSE')
```

Subsequently, the data from the vision sensor and the motors of CoppeliaSim were extracted so that they could now be controlled by the algorithm developed in Python, see Code 2.

Code 2. Extraction of vision sensor data and motors from CoppeliaSim to be connected with Python

```
####Conexión de motores y sensor de visión###

#Guardar el handle del sensor de visión
error_cam, cámara = sim.simxGetObjectHandle(clientID,
'Vision_sensor', sim.simx_opmode_oneshot_wait)
error_izq, motor_izquierdo =
sim.simxGetObjectHandle(clientID,
'nakedCar_motorLeft', sim.simx_opmode_oneshot_wait)
error_der, motor_derecho =
sim.simxGetObjectHandle(clientID,
'nakedCar_motorRight', sim.simx_opmode_oneshot_wait)
error_eje_izq, eje_izq =
sim.simxGetObjectHandle(clientID,
'nakedCar_steeringLeft', sim.simx_opmode_oneshot_wait)
error_eje_der, eje_der =
sim.simxGetObjectHandle(clientID,
'nakedCar_steeringRight',
sim.simx_opmode_oneshot_wait)
If error_cam or (error_izq or error_der or
error_eje_izq or error_eje_der): sys.exit('ERROR AL
CONECTARSE')

#Capturar un frame para activar el sensor
_, resolution, image =
sim.simxGetVisionSensorImage(clientID, camara, 0,
sim.simx_opmode_streaming)
time.sleep(1)
```

Following this, the image obtained by the vision sensor was processed, using the Python OpenCV, see Code 3.

Code 3. Image processing using the Python OpenCV library

```
#Capturar un frame de la cámara del robot y guardar la
imagen y su resolución
_, resolución, image =
sim.simxGetVisionSensorImage(clientID, cámara, 0,
sim.simx_opmode_buffer)
img = np.array(image, dtype=np.unit8) #Convertir la
imagen en un array de numpy
img.resize([resolution[0], resolution[1], 3] #Cambiar
sus dimensiones
img = np.rot90(img, 2) #Rotarla 90 grados para
enderezarla
img = np.fliplr(img)

#Aplicar escala de grises, canny, blur y líneas
img2 = np.copy(img)
imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(imgGray, (7,7), 0)
bordes = cv2.Canny(blur, 50, 200)
ref_linea = cv2.HoughLinesP(bordes, 2, np.pi/180, 20,
np.array([]), minLineLenght=40, lineasplus =
corregir(img2, ref_linea)
recta = Mostrar_lineas(img2, lineasplus)
imagenlines = cv2.addWeighted(img2, 0.8, recta, 1, 1)
```

3.3 PID controller implementation

Finally, the PID controller was implemented in Python, which is used to control the steering of the vehicle, this allows the robot to stay within the lane, see Code 4 for reference.

Code 4. PID controller implemented in Python

```
#Controlador PID
Kp = 0.2
Ki = 0.08
Kd = 0.3

error = posicion (img2, ref_lines) -160
print(error)
error_i =+ error
PID = Kp*error + Ki*(error_i) + Kd*(error-
error_anterior)
error_anterior = error
```

4. RESULTS

After implementing the advances obtained, it was possible to obtain the simulation of a mobile robot with an Ackermann configuration that is capable of autonomously following the lane on a track using a perspective vision sensor and a PID-type steering controller with feedback, thus fulfilling the objectives of the work. As mentioned above, the implemented PID controller allowed us to algorithmically manage the behavior of the vehicle. The gain values listed in Table 1 for the PID controller employed in this work were empirically obtained by a series of trial-and-error to obtain the best results.

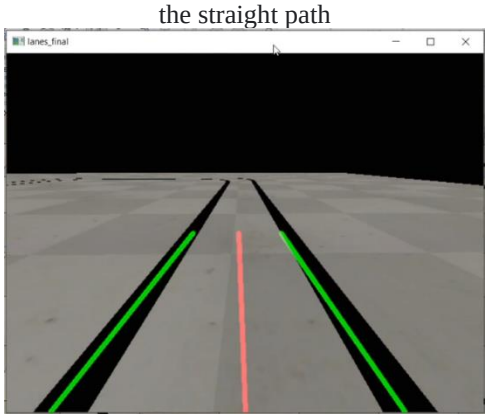
Table 1. Gain values for the PID controller

Gain	Value
K_p	0.20
K_i	0.08
K_d	0.30

During testing, it was observed that the rendering speed depended on the vehicle's speed, therefore it was decided to maintain a lower speed to ensure accurate processing and correction. With the set gain values for the PID controller, the robot could follow a straight path with high accuracy and a constant speed.

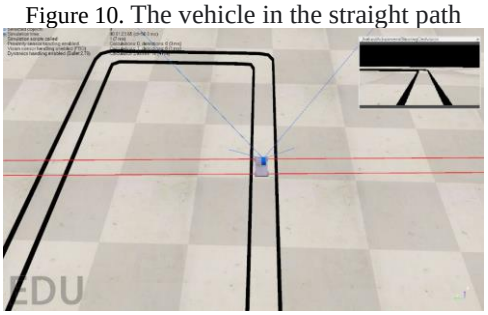
With the implementation of these gain values for the PID controller the results in the straight path were very precise with an appropriate speed, see Figure 9.

Figure 9. Simulation of the image processing of the sensor in the straight path



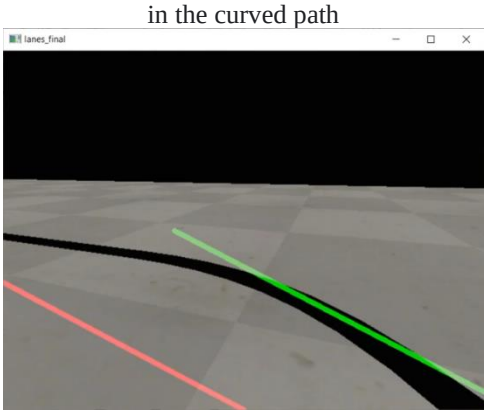
When following a straight path, the PID controller's implementation of the prescribed gain values has produced good results in terms of accuracy and speed. The controller produced movements with a high degree of accuracy while maintaining a constant speed. This improvement is evidence of the PID controller's efficiency. Figure 10 shows the vehicle following a straight path.

Figure 10. The vehicle in the straight path



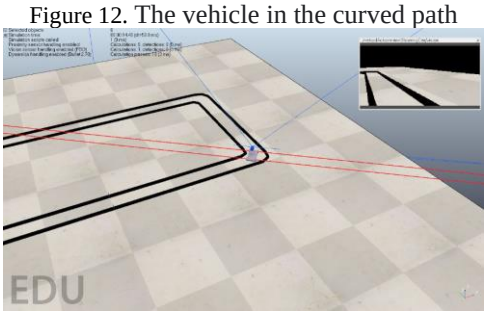
In the curved path, we also had good results, the reaction was fast and accurate thanks to these PID gain values. Despite losing the reference of a line, see Figure 11, it did not go off the rail or lose control. It is important to emphasize that it works in one way (a clockwise trip) or another (a counterclockwise trip).

Figure 11. Simulation of the image processing of the sensor in the curved path



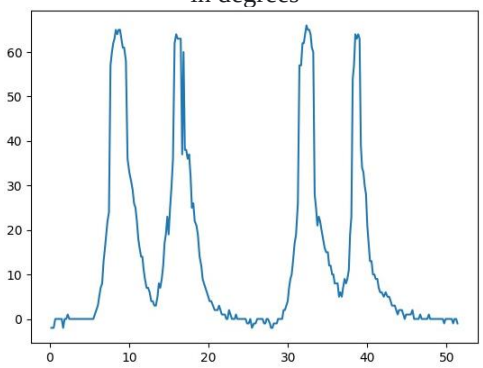
When traversing along the curved path, the PID controller has achieved a good level of precision and speed thanks to the implementation of the set gain values shown in Table 1. Figure 12 shows the vehicle following a curved path.

Figure 12. The vehicle in the curved path



Nevertheless, in Figure 13 it can be observed that there is a difference in the output error between the curved path and the straight path.

Figure 13. Error output graph, where the horizontal axis represents the time in seconds and the vertical axis the error in degrees



On average, the vehicle completed a full lap of the circuit in 20 seconds without deviating from the rail or losing control.

5. CONCLUSIONS

Throughout this work, it was possible to apply knowledge from the Control Engineering area, as well as from other areas, to process the image obtained and control the direction of the mobile robot to follow the lane on the oval track fulfilling the objectives of this work.

The knowledge and experience obtained throughout the development of this work made it possible to obtain a PID controller with an output capable of directing the movement of the robot both in straight sections and in curves in both directions.

Likewise, we hope that in the future we can apply the knowledge that has been obtained in this work in other kind of robot configurations like in applications self-driving cars, exploration vehicles, unmanned aerial vehicles, and autonomous underwater vehicles.

REFERENCES

[1] U. Orozco-Rosas, K. Picos and O. Montiel, "Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots," *IEEE Access*, vol. 7, no. 1, pp. 156787-156803, 2019.

[2] C. Bartneck, C. Lütge, A. Wagner and S. Welsh, An Introduction to Ethics in Robotics and AI, Christchurch, New Zealand: Springer, 2021.

[3] U. Orozco-Rosas, O. Montiel and R. Sepúlveda, "Parallel Evolutionary Artificial Potential Field for Path Planning—An Implementation on GPU," in *Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization*, Studies in Computational Intelligence, vol 601, Springer, 2015.

[4] A. Chouza, R. Hernandez, J. L. Jimenez and U. Orozco-Rosas, "Navegación autónoma de un vehículo aéreo por referencia visual," in *Revista Aristas: Investigación Básica y Aplicada*, Tijuana, Mexico, 2020.

[5] U. Orozco-Rosas, O. Montiel and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Applied Soft Computing*, vol. 77, pp. 236-251, 2019.

[6] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor and A. Cuesta-Infante, "Mobile robot path planning using a QAPF learning algorithm for known and unknown environments," *IEEE Access*, vol. 9, no. 1, pp. 101217-101238, 2022.

[7] R. Acharya and D. Jena, "Sampling based motion planning of Ackermann steering system using transformation," in *IEEMA Engineer Infinite Conference (eTechNxT)*, New Delhi, India, 2018.

[8] J. Kim, D. Han and B. Senouci, "Radar and Vision Sensor Fusion for Object Detection in Autonomous Vehicle Surroundings," in *Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, Prague, Czech Republic, 2018.

[9] U. Orozco-Rosas, K. Picos, O. Montiel and O. Castillo, "Environment Recognition for Path Generation in Autonomous Mobile Robots," in *Hybrid Intelligent Systems in Control, Pattern Recognition and Medicine*, Studies in Computational Intelligence, vol 827, Springer, 2020.

[10] T. Wang and C. Chang, "Hybrid Fuzzy PID Controller Design for a Mobile Robot," in *EEE International Conference on Applied System Invention (ICASI)*, Chiba, Japan, 2018.

[11] J. A. Rodriguez Olea, I. D. Leon Bon, U. Orozco-Rosas and K. Picos, "Vehículo autónomo de configuración Ackermann para seguimiento de carril mediante procesamiento digital de imagen," in *Revista Aristas: Investigación Básica y Aplicada*, Tijuana, Mexico, 2020.