Control of a Mobile Robot for Figure Recognition Applying Lambda Tuning and Machine Vision Techniques

Mario González Moreno[1], Jonathan Pérez Arellano[2], José Federico Ramírez Cruz[3]

[1]Tecnológico Nacional de México/ Instituto Tecnológico de Apizaco, Apizaco, Tlaxcala, México
m193713671[at]apizaco.tecnm.mx

[2]Tecnológico Nacional de México/ Instituto Tecnológico de Apizaco, Apizaco, Tlaxcala, México
m19371371[at]apizaco.tecnm.mx

[3]Tecnológico Nacional de México/ Instituto Tecnológico de Apizaco, Apizaco, Tlaxcala, México
federico.rc[at]apizaco.tecnm.mx

Abstract: ***The complexity of the environment in which a mobile robot moves are a fundamental factor for different navigation failures, even invalidating the objective. This paper approaches the design and implementation of a system for the control of a mobile robot provided by computer vision techniques, which is able to verify the direction and tracking of a target point in a specific environment. This system integrates the Lambda tuning method to analyze and design proportional integral derivative (PID) controllers and ensures robustness, stability and non-oscillating response. The images used were subjected to pre-processing using segmentation, thresholding and morphological transformation algorithms in order to improve their quality and simplify the search of figures. Consequently, the lambda method resulted in the resolution of oscillation problems allowing the user to acquire control parameters easily and with high performance. Python and Arduino perform the simulations and experimental tests of the system, verifying that the tool complies with reaching the detection of the desired figure.***

Keywords: PID control, Lambda tuning, artificial vision

1. Introduction

Nowadays, it is very common to talk about robots, since they are programmable machines with a certain degree of intelligence, they are composed of electromagnetic systems and artificial machinery, capable of performing certain operations automatically depending on the decisions that have the structure of its program, especially heavy, repetitive, dangerous tasks.

At the moment, mobile robots are considered a field of advanced technology to deal with very complex problems. Its products consist of applications in the fields of control, programming, artificial intelligence, perception and instrumentation, and serve as a basis for progress in various fields of industry, providing innovative technological solutions to develop better robots and expand the scope of available applications.

The autonomous motion of each machine is achieved through the implementation of control techniques, such as PID. This is a control mechanism whereby the speed and flow of other process variables can generally be adjusted through a feedback loop (IEEC, 2021). The PID controller calculates the difference between our actual variable and the desired variable, having as main objective to ensure the correct operation of the automation system. Also note that no automatic program exists to adjust the controller to ensure its robustness and optimal response, so the industry has begun to take advantage of the Lambda adjustment method to generate a great economic impact.

On the other hand, the constant development of autonomous vehicles has put more attention on computer vision systems, many of these common car models already include sensors and cameras to avoid accidents and even provide driving assistance. Specifically, in autonomous systems, object detection and tracking methods are crucial because they are used to find specific targets and obstacles or even identify people.

The main reason behind this project is the necessity to combine robotics and computer vision to create a more effective autonomous system that can perform specific tasks in their environment.

This article is structured as follows: Section 2 describes the background of the research and work previously carried out. In section 3, the knowledge methodology is presented, in charge of developing, defining and systematizing the set of techniques, methods and procedures used. As well as the development procedure where the architecture of the quadrature encoder and the detection of the robot's direction are verified. Likewise, artificial vision techniques for object detection are described. Section 4 presents the experimental results of the system, thus concluding, with section 5 showing the conclusions and future work.

2. Literature review

(Allagui et al., 2019)proposes a robust control strategy to address the problem of autonomous navigation of mobile robots. This strategy uses the concept of a fractional order derivative integral proportional controller FOPID (Tajjudin et al., 2019).

(Geetapriya et al., 2019) focuses on trajectory planning algorithms and proposes a new algorithm for deep search; modified so that the robot can navigate in a maze-like environment, without crashing into obstacles and planning its way through the maze.

(Gatesichapakorn et al., 2019) presents an implementation of an autonomous mobile robot with the Robot Operating System (ROS). The system uses 2D LiDAR and RGB-D camera with ROS 2D navigation stack. The contribution of this article is that two ROS navigation stack system configurations are proposed. The first system is implemented on Raspberry Pi 3 using only 2D LiDAR. The second system is implemented on Intel NUC using 2D LiDAR and RGB-D camera.

(Nilwong & Capi, 2020) features a map-based navigation system for mobile robots. The proposed navigation system can guide the robot's using maps in the form of 2D images. It also includes the trajectory planning segment and the navigation segment where a star search algorithm is used to plan them. In the navigation segment, Q learning(Pandey & Pandey, 2010) is applied to train the robot to follow the planned routes on the map.

(Sergey, 2020)considers a control system for mobile robots based on an ultrasound navigation system. Nema 17 stepper motors were used as hardware in the experiments, and the BNO080 smart sensor, which combines the accelerometer, gyroscope and magnetometer, as the relative rotation sensor of the mobile platform. The probabilistic roadmap method was used to find the optimal trajectory, this procedure proved its effectiveness in the field of seismic signal analysis.

3. Design and development procedure

3.1 Methodology

In Fig. 1 presents the process divides into five stages for the final objective, which is the identification of the object. Each stage represents a series of steps that are broken down throughout this section.

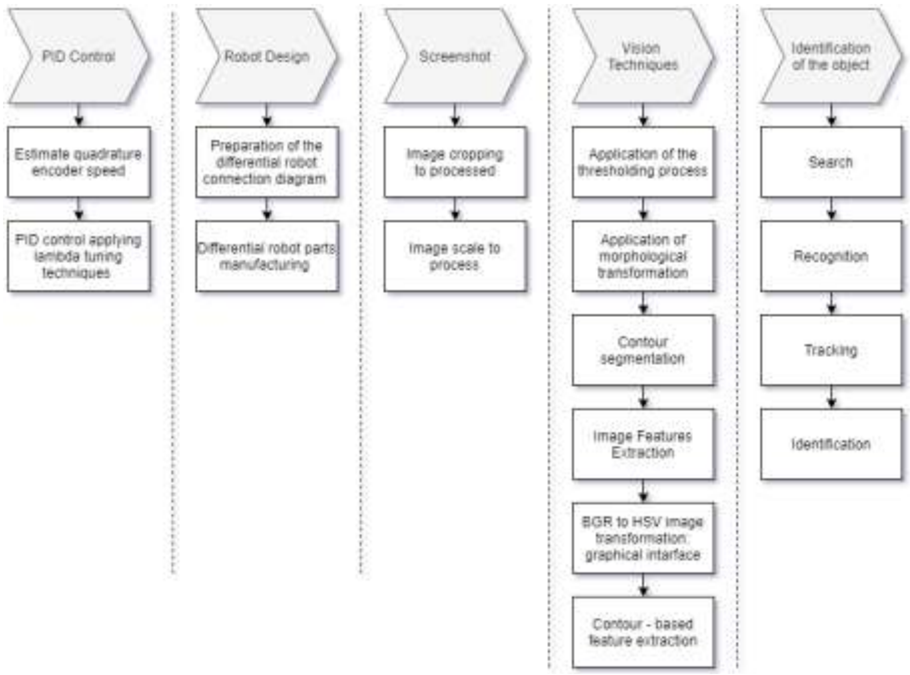

PID Control
Estimate quadrature encoder speed
PID control applying lambda tuning techniques
Robot Design
Preparation of the differential robot connection diagram
Differential robot parts manufacturing
Screenshot
Image cropping to processed
Image scale to process
Vision Techniques
Application of the thresholding process
Application of morphological transformation
Contour segmentation
Image Features Extraction
BGR to HSV image transformation graphical interface
Contour - based feature extraction
Identification of the object
Search
Recognition
Tracking
Identification

Figure 1: Methodology represented with a block diagram

In the first instance, the image is captured using the native camera of a cell phone, for the case an Android cell phone with 48 megapixels of resolution was used. Subsequently, the digital image processing was carried out, understanding this as the storage, transmission and representation of the information of the images by means of a computer. Later, an image segmentation algorithm was used, which is the process of dividing a digital image into several parts or objects to facilitate analysis.

Then, the identification of the object is made as detailed in section 5.7, it is compared with the vertices of the object to be searched by means of its invariant moments, obtaining a numerical value of such comparison. A signal is placed on the original image, using the coordinates of its location and the number obtained from the vertex comparison. The number represents the inequality between both figures.

Finally, the robot control is elaborated, which is formed by complex electronic systems (hardware implementation: programming to Chihai Motor CHR-GM25-370 motors of 140RPM at 12V, L298N controller and Arduino UNO microcontroller) that control their actions by means of a computer (software implementation: Arduino IDE, Python), through which the program describing the action to be performed by each element is introduced.

3.2 Robot control system

This section describes the implementation of materials used for the development of the project. Programming using the IDE Arduino development environment is applied to Chihai Motor CHR GM25-370 motors from 140RPM at 12V, L298N controller and Arduino Nano microcontroller.

3.2.1. PID control

Quadrature Encoder: It is an incremental rotary type Encoder that has the ability to indicate both the position and the speed and direction of movement of the motor shaft. In Fig.2, shown that the Encoder has two Hall effect sensors (Akin & Bhardwaj, 2010), which are attributed to the rotating disk mounted on the rear motor shaft generating two digital pulse signals with a phase difference of 90 degrees in quadrature. These output signals are commonly referred to as A and B. The sequence of activation and deactivation of these two sensors allows us to know the direction and amount of displacement that occurs in the Encoder. Tab. 1. shows the list of parameters of the geared motor used in the system.

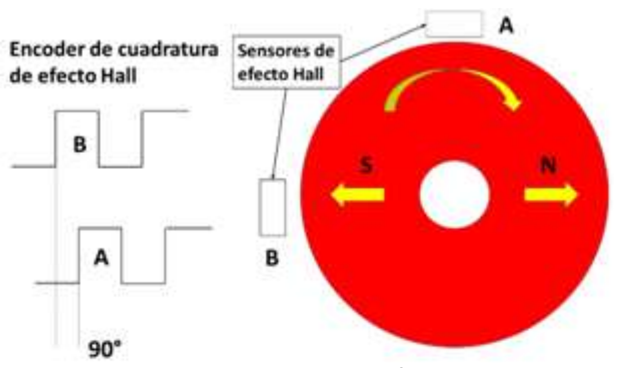


Figure 2: Internal diagram of an encoder

Table 1: Gearmotor parameters

Gearmotor parameter list	
Voltage	DC 12.0 V
No-load speed	140 RPM 0.1 A
Maximum efficiency	LOAD 1.7 kg.cm/110 rpm/2.1 W/0.33 A
Maximum power	LOAD 4.3 kg.cm/70 rpm/0.75 A
Stall	STALL TORQUE 8.5 kg.cm, STALL CURRENT 1.4 A
Retarder Reduction ratio	1: 45
Holzer Resolution	Motor Holzer 11 * ratio 45=195 PPR

Quadruple precision resolution: The resolution of an encoder is the measure of the number of pulses per revolution (PPR), that is, the number of counts produced. For the Chihai Motor CHR-GM25-370 from 140RPM at 12V, the Quadrature Incremental Encoder generates 11 counts per revolution of the motor shaft, which corresponds to 495 counts per revolution of the output shaft of the Holzer resolution gearbox (v. M. buyanov, 1967).

For robotic applications, quadruple precision is used, which allows the direction of rotation of the motor shaft to be detected. Therefore, the resolution of the Quadruple Pressure Encoder (R) is different from the Holzer resolution calculation method.

$$R = mH \times s \times r \tag{1}$$

where: R – encoder resolution,
mH – number of counts per motor shaft resolution: 11 counts (Holzer motor),
s – number of states generated by the AB channels: 4,
r – reduction ratio of the gearbox (ratio): 45,

Substituting,

$$R = 11 \times 4 \times 45 = 1980 \tag{2}$$

Position in degrees: To measure the relative position of the motor shaft.

$$P = \frac{n \times 360}{R} \tag{3}$$

where: P – position in degrees,
n – number of counts generated,
R – encoder resolution: 1980 counts per revolution,

Angular velocity in RPM: To measure the number of revolutions per minute.

$$N = \frac{n \times 60}{t \times R} \tag{4}$$

where: N – rotational speed in revolutions per minute (RPM),
n – number of counts generated in a given time t,
t – pulse generation time in seconds (s),
R – encoder resolution: 1980 counts per revolution,

Angular velocity in radians per second (rad/s): Find the angular velocity in rad / s with the revolutions per minute. One revolution equals 2π radians and 1-minute equals 60 seconds. Therefore, the angular velocity is calculated as below:

$$\omega = \frac{2\pi \times N}{60} \tag{5}$$

where: ω – rotational speed in radians per second (rad/s),
N – rotational speed in revolutions per minute (RPM),

Linear speed in meters per second (m/s): To calculate the linear speed in meters per second, the angular velocity is multiplied by the radius of the rim assembled to the output shaft of the gearbox.

$$u = r_{wheel} \times \omega \tag{6}$$

where: u – linear velocity in meters per second (m/s),
ω – rotation speed in radians per second (rad/s),
r_{wheel} – wheel radius in meters m.

3.2.2PID control Lambda tuning

The Lambda method, which originated in the paper industry and continues to be popular in that industry today, is essentially a synthesis method; that is, the controller is specifically for the process (Olsen & Bialkowski, 2002). When founded by Dahlim in 1968 (Dahlin, 1968), lambda adjustment provides a new method for coordinating paper mill loop adjustment to achieve process stability and a uniform product. This adjustment technique is relatively new to other industries.
To obtain the PID design, we consider the serial or interactive PID transfer function:

$$C'(s) = K'_c \left(\frac{(1+sT'_i)(1+sT'_d)}{sT'_i} \right) \tag{1}$$

where: $K'c$ – proportional gain,

T'i – integral time,
T'd – derivative time,

The plant model corresponding to a first order system with delay is given as:

$$G(s) = \frac{K}{1+sT} e^{-sL} \tag{2}$$

where: *K* – process gain,
T – open-loop time constant,
L – delay,

The exponential expression (2), can be calculated by means of the first order Páde approximation (Álamo, 2007), that is to say:

$$e^{-sL} = \frac{1-{}^{L}/_{2}s}{1+{}^{L}/_{2}s} \tag{3}$$

Then, substituting the approximate value (3) in the plant model (2), the transfer function becomes:

$$G(s) = \left(\frac{K}{1+st}\right)\left(\frac{1-{}^{L}/_{2}s}{1+{}^{L}/_{2}s}\right) \tag{4}$$

The lambda tuning method consists of canceling the process poles with the controller zeros (Huang et al., 2002). Therefore, we consider the integral time as $T_i = T$ and the derivative time as $T_d = L/2$ then, the loop transfer function has the following form:

$$G_1(s) = G(s)C'(s) = \frac{KK'{}_{C}(1-s\,{}^{L}/_{2})}{sT} \tag{5}$$

Then, the characteristic equation of the closed-loop system is given by:

$$s\left(T - KK'_c\,{}^{L}/_{2}\right) + KK'_c = 0 \tag{6}$$

The closed-loop pole required is $s = -1/\lambda$, giving the following tuning rules:

$$K'_c = \frac{T}{K({}^{L}/_{2}+\lambda)} T'_i = TT'_d = {}^{L}/_{2} \tag{7}$$

It is observed in (7), a parameter that regulates the response speed of the method. Increasing λ decreases the response speed and decreasing λ has the opposite effect. For a strong controller $\lambda = 3T$ and for an immediate tuning $\lambda = T$.

In this project, the PID controller in non-interacting or standard (ISA) form(Åström & Hägglund, 1995) is used. Therefore, (7) can be expressed as a standard form with the following equation:

$$K_c = K'_c \frac{T'_i+T'_d}{T'_i} T_i = T'_i + T'_d T_d = \frac{T'_i T'_d}{T'_i+T'_d} \tag{8}$$

For the PID design, the controller with the transfer function is considered:

$$C_1(s) = K_c\left(\frac{1+sT_i}{sT_i}\right) \tag{9}$$

A time integral $T_i = T$, if the time delay is approximated by Taylor series e-sl. The characteristic equation of the closed-loop system is:

$$s(T - KK_cL) + KK_c = 0 \tag{10}$$

Giving the following tuning rules for both standard and serial controllers:

$$K_c = \frac{T}{K(L+\lambda)} T_i = T \tag{11}$$

3.2.3 Robot design

Differential Robot: connections: In Fig. 3., the connection diagram is presented where it can be observed that the Arduino Nano microcontroller, which controls the entire robot. It integrates an HC-05 Bluetooth module for wireless communication, a driver for controlling the motors with Chihai Motor CHR-GM25-370 quadrature encoder from 140RPM at 12V in this case it is the L298N module, and a 1600mah 11.1v Lipo Battery 3s 20c Turnigy Robotics Drone Rc F who is responsible for giving power to the entire circuit, finally a general switch to turn the entire system on and off.

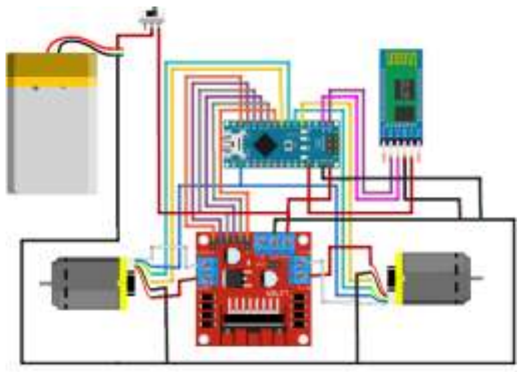


Figure 3: Mobile robot circuit

Differential robot: structure: Fig.4. shows the robot that has been manufactured with 3D printed plastic castings: motor coupling, chassis, motor support, tension pulley support, chassis support. The implemented motion mechanism is of tricycle type with differential system. The movement of the robot is performed by two servomotors with quadrature encoder, which can control the movement of the robot.

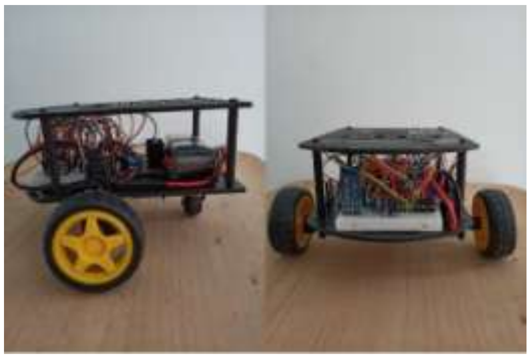


Figure 4: Mobile robot model

3.3 Artificial vision techniques

For the development of machine vision techniques (Constante et al., 2016), the Python programming language and modules such as NumPy, which is essential for handling matrices or arrays, were used, and OpenCV was also used for the vision stage, which is essential for the execution of the algorithms.

3.3.1 Image capture

Image capture allows the creation of an image data set where the training for object recognition or detection is performed. Fig. 5 shows the graphical interface developed in Python language that is used to start the image processing.

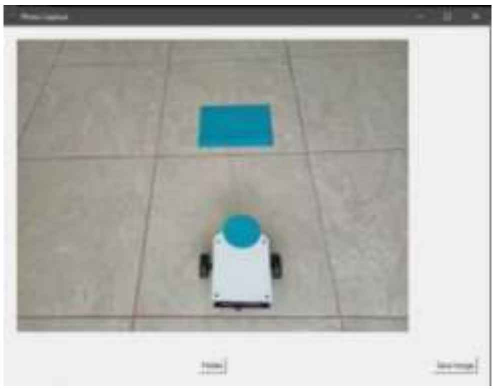


Figure 5: Graphical interface for imagen capture

3.3.2 Image processing and capture: thresholding method

Fig. 6. shows the applied thresholding method, which is the process by which an optimal threshold is sought that allows the objects in the background to be distinguished from the foreground objects in an image. This threshold is the value at which the histogram of an image is divided in two; OpenCV uses cv. threshold to apply the threshold. The suggested syntax is as below:

Syntax: ret, thresh= cv. threshold (img,127,255, cv. THRESH_BINARY)

Arguments: img: image to be processed (grayscale image).
127: threshold value.
255: maximum value.
cv.THRESH_BINARY: threshold type

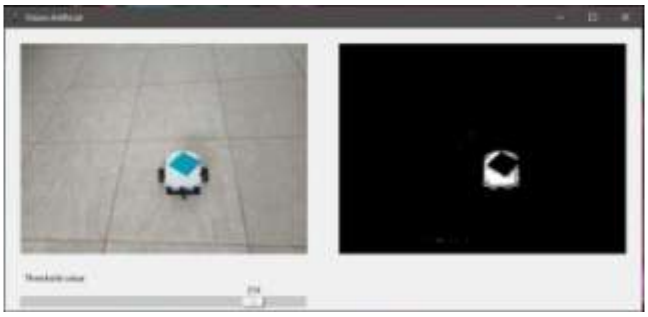


Figure 6: Original imagen applying thresholding method

3.3.3 Morphological transformation

These are simple operations based on the shape of the image and are usually applied to binary images. Two inputs are required, the original image and the kernel, which determines the nature of the operation. Two morphological conversions are used in the development of the application:

a) Morphological transformation opening: The opening is the result of erosion followed by dilation, which is useful for eliminating noise.
b) Morphological transformations closure: The closed operation is the reciprocal of the open operation. It first expands and then corrodes.

Fig. 7. shows the effect of the morphological transformation, the upper left part shows the original image to be processed, the upper right part shows the thresholding process, the lower left part shows the opening transformation of the image, and finally, the lower right part shows the closing transformation of the original image. The suggested syntax is as below:

Syntax: opening= cv2.morphology (img, cv2.MORPH_OPEN, kernel)
Arguments: img: image to be processed (binary image).
cv2.MORPH_OPEN: type of transformation.
kernel: mask or core.

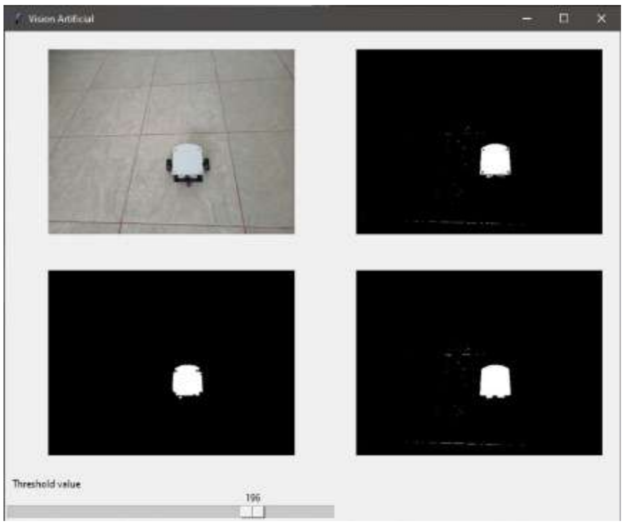


Figure 7: Morphological transformation

3.3.4 Segmentation of contours

In OpenCV, finding outlines is like finding a white object from a black background. The object to be found must be white and the background must be black. The suggested syntax is as below:

Syntax: contours, hierarchy = cv2.findContours (thresh, cv2.RETR_EXTERNAL, cv. CHAIN_APPROX_SIMPLE).
Arguments: thresh: image to be processed (binary image).
cv2.RETR_EXTERNAL: returns only the extreme outer contours.
cv.CHAIN_APPROX_SIMPLE: contour approximation method.

To draw the contours, the function cv2.drawContours is used. As can be seen in Fig. 8 shows the contours drawn in green color of the detected objects, on the left side we see that there are countless objects, everything that is white is detected as an object, which do not interest us for recognition, with the slider we regularize the amount of desired objects in this case the region of interest is only the robot, however there are still green regions, in the next section we eliminate these small objects detected by a size filter using the area of each object. The suggested syntax is as below:
Syntax: cv2.drawContours(img, contours, -1, (0,255,0), 3)

Arguments: img: image where the contours are to be shown.
(-1): useful when drawing individual contours, -1 draws all contours.
(0,255,0) : color (B, G, R).
(3) : contour thickness (-1 fills the contour).

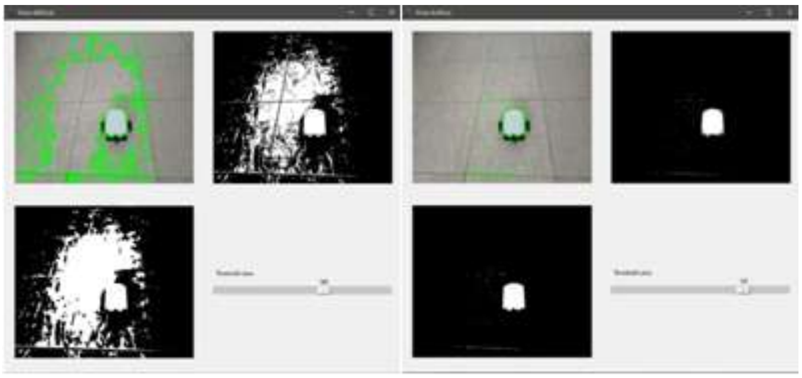


Figure 8: Segmentation of contours

3.3.5 Feature extraction

In this part, the extraction of features where a point of interest is detected and allow to perform the control is studied. Next, three more processes are listed to determine the region of interest.

a) Image moments: help to calculate some features, such as the center of mass of the object, the area of the object etc. The suggested syntax is as below:

Syntax: cnt= contours [0] M= cv2.moments(cnt)

Arguments: cnt: contour.

M: returns a dictionary.

b) Centroid: given by of all calculated moments:

$$C_x = \frac{M_{10}}{M_{00}} \tag{12}$$

$$C_y = \frac{M_{01}}{M_{00}} \tag{13}$$

c) Area: given for the moment, M[‘m00’]. Now that the programming processes have been applied in the application to eliminate the noise, we continue with the sample Fig.9. In the left part of the image, there is a better result by detecting fewer white regions (objects), additionally we obtain the centroid, represented by the red dot as shown in the left part of the image.

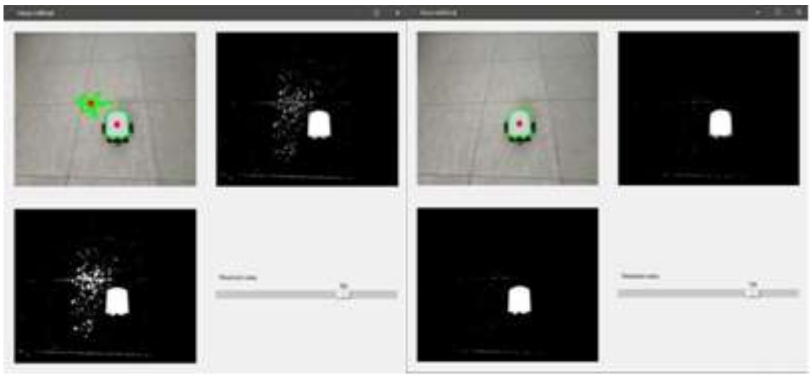


Figure 9: Feature extraction, locating the centroid of the desired object

3.3.6 BGR to HSV image transformation: graphical interface

When an image is read in OpenCv, by default it is read in BGR, so it needs to be transformed to the HSV (HUE, SATURATION, VALUE) color space. The suggested syntax is as below:

Syntax: imgHSV= cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

Arguments: img: BGR image to be processed

cv2.COLOR_BGR2HSV: BGR to HSV transformation method

It is necessary to know what values can take each of the HSV channels in OpenCv to obtain a better result when locating the desired color, in this case the color of interest is blue, as shown in Fig. 10:

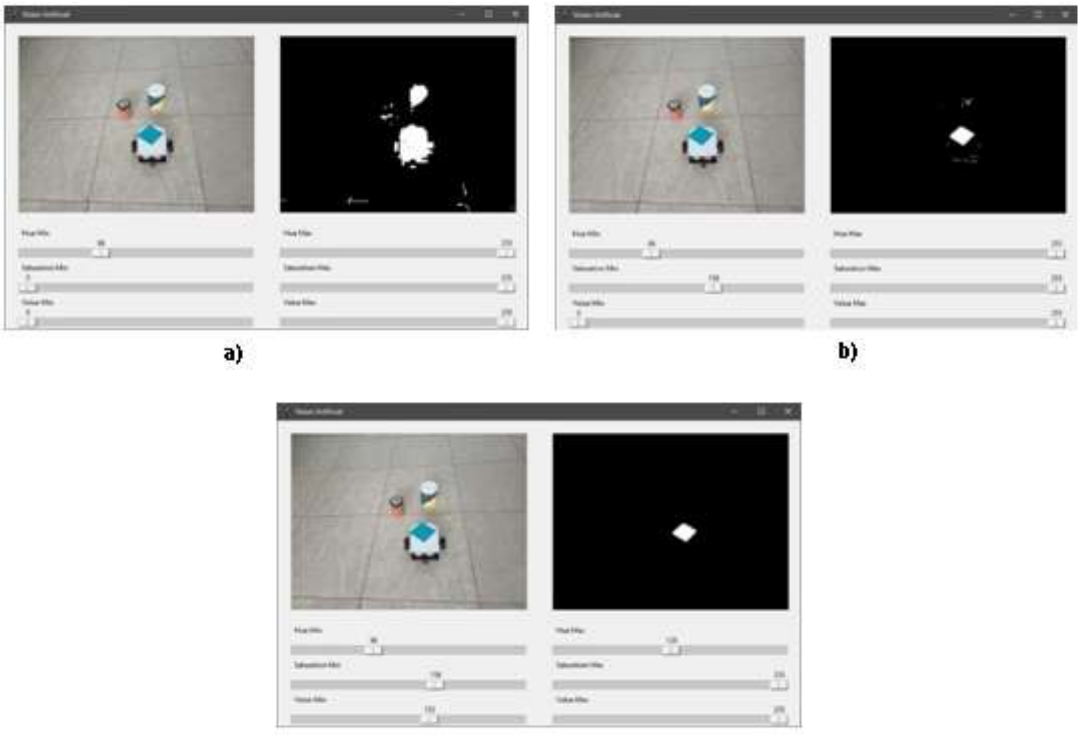

a) b) c)

Figure 10: Graphical interface, BGR to HSV

3.3.7 Contour based features

a) Contour features Perimeter: It is also called arc length. It can be found using the function cv2.arcLength().
Syntax: perimeter= cv2.arcLength(cnt, True)
Arguments: cnt: contour
True: specifies if the shape is a closed contour

b)Contour features Approximation: Approximates a contour shape to another shape with fewer vertices depending on the specified approach. The suggested syntax is as below:
Syntax: epsilon= 0.01*cv2.arcLength(cnt, True)
approx= cv2.aproxxPoly(cnt, epsilon, True)

Arguments: epsilon: maximum distance from the contour to the approximate contour

Based on the contours previously reviewed in this section, we work with features that allow the detection of shapes for a better recognition of objects, Fig. 11 shows the detection of the shape depending on the number of vertices that the object has, in this case objects with 4 vertices are detected.

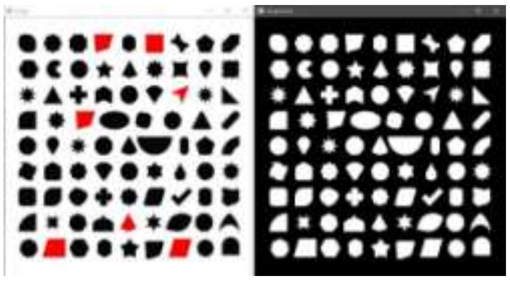


Figure 11: Object detection

4. Experimental Results

4.1 Open loop response

The tuning of a PID controller is shown here. In the graphs of Fig.12., shows the result of the real process variable represented in orange color. The variable estimated by the algorithm is shown in blue. The control variable represented in green color, approaches 40% at 3 seconds of revolution. The error approaches 0, as shown in the right graph of Fig.12. With these data have step response, which means to reach an optimal point of motor speed. Based on this have the open loop solution for the first order system with delay, then the Lambda tuning technique is applied.

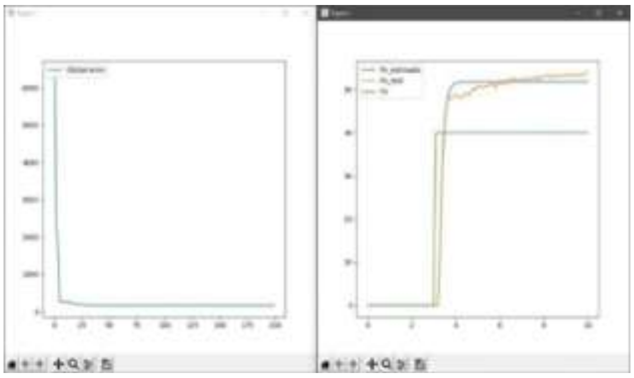


Figure 12: Open loop output model

4.2 Closed loop response, Lambda tuning

Fig. 13. visualizes the control action with setpoint applying the Lambda tuning method, the system fully responds to the step input and a variable Setpoint input, implementing the PID controller. It has the characteristic of stabilization time of 3 seconds without over impulse and good response of the control action.

4.3 Object tracking robot based on colors and shapes

The mobile robot moves over an environment, which covers an area of approximately 1.80 m wide by 1.20 m high; the scenario is complemented by a main camera of an Android cell phone located at a height of 1.20 m. In the area there is a flat blue rectangular figure, in the upper part of the robot there is a blue circular figure with a white background, which serves as a control and tracking algorithm to complete the objective of moving towards its goal.

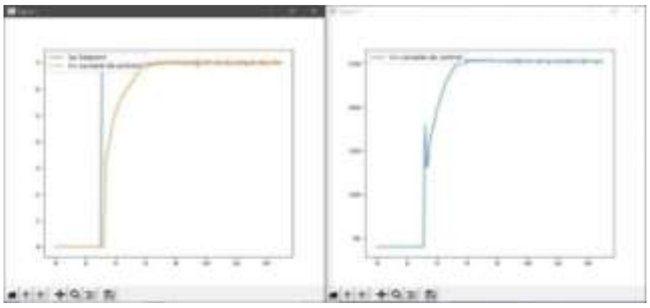
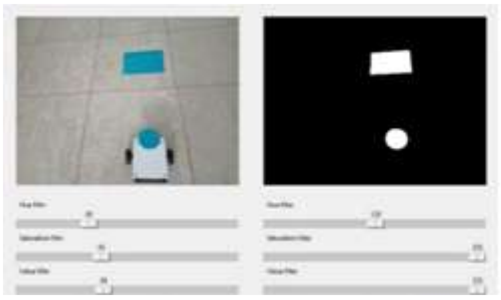
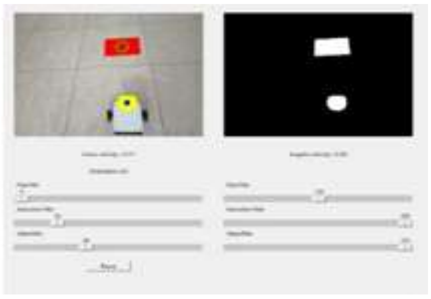


Figure 13: Setpoint step model, Lambda tuning

When recognizing the rectangle, automatically the vision system will paint it in red and the circle will paint it in yellow, the centroid of each figure is detected and the robot begins to advance until these figures have a meeting as shown in Fig. 14.


(a)


(b)

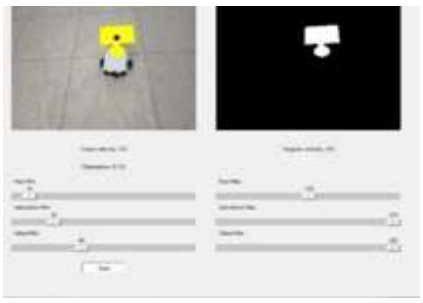

(c)

Figure 14: Color and shape object detection

5. Conclusion and Discussion

This paper presents the application of the lambda tuning method for a PID controller that controls the speed of a motor of a mobile robot. This method was implemented with an Arduino microcontroller. The execution of the lambda adjustment rules during the autoregulation process provided a response without overshoot or spikes, resulting in a uniform response. This method uses a parameter called closed-loop time constant whose function is to increase or decrease the system response.

Finally, the experimental results demonstrated the good performance of the proposed tool. On the other hand, in the area of artificial vision, it proposes to continue with the study and development of image processing techniques, applied to the solution of different problems in robotics, such as: global vision of the robot's workspace and extraction of relevant information. Some points to be developed in the future are: camera calibration and extension of the algorithms for cooperative robotics with detection of several robots.

References

[1] Akin, B., & Bhardwaj, M. (2010). Trapezoidal Control of BLDC Motors Using Hall Effect Sensors_sprabq6. *Control*, *July*, 1–33.

[2] Álamo, T. (2007). Diseño del Controlador PID. *Universidad de Sevilla, Departamento de Ingeniería de Sistemas y Automática*, 1–37.

[3] Allagui, N. Y., Abid, D. B., & Derbel, N. (2019). Autonomous navigation of mobile robot with combined fractional order PI and fuzzy logic controllers. *16th International Multi-Conference on Systems, Signals and Devices, SSD 2019*, 78–83. https://doi.org/10.1109/SSD.2019.8893176

[4] Åström, K. J., & Hägglund, T. (1995). *PID controllers: theory, design, and tuning* (Vol. 2).

[5] Constante, P., Chang, O., Pruna, E., & Escobar, I. (2016). Artificial Vision Techniques for Strawberry ' s

Industrial Classification. *Ieee Latin America Transactions*, *14*(6), 2576–2581.

[6] Dahlin, E. B. (1968). Designing and Tuning Digital Controllers. In *Instruments and Control Systems* (pp. 77–84).

[7] Gatesichapakorn, S., Takamatsu, J., & Ruchanurucks, M. (2019). ROS based Autonomous Mobile Robot Navigation using 2D LiDAR and RGB-D Camera. *2019 1st International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics, ICA-SYMP 2019*, 151–154. https://doi.org/10.1109/ICA-SYMP.2019.8645984

[8] Geetapriya, S., Pillai, N. R., Aswin, C. K., & Menon, M. (2019). Graph-based algorithm for mobile robot navigation in a known environment. *Proceedings of the International Conference on Trends in Electronics and Informatics, ICOEI 2019*, *Icoei*, 278–281. https://doi.org/10.1109/ICOEI.2019.8862775

[9] Huang, H. P., Roan, M. L., & Jeng, J. C. (2002). On-line adaptive tuning for PID controllers. *IEE Proceedings: Control Theory and Applications*, *149*(1), 60–67. https://doi.org/10.1049/ip-cta:20020099

[10] IEEC. (2021). *Sistemas de Control y Proceso Adaptativo. Reguladores y Comunicación*. 1–16. http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/Informacion_de_referencia_ISE8_1_1.pdf

[11] Nilwong, S., & Capi, G. (2020). Effects of Training Difficulties on Reinforcement Learning Based Outdoor Robot Navigation System. *17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2020*, 300–303. https://doi.org/10.1109/ECTI-CON49241.2020.9158278

[12] Olsen, T. (Emerson), & Bialkowski, B. (Emerson). (2002). Lambda Tuning as a Promising Controller Tuning Method for the Refinery. *AIChE Spring National Meeting*, *42*.

[13] Pandey, D., & Pandey, P. (2010). Approximate Q-learning: An introduction. *ICMLC 2010 - The 2nd International Conference on Machine Learning and Computing*, 317–320. https://doi.org/10.1109/ICMLC.2010.38

[14] Sergey, B. (2020). Ultrasonic navigation to control the movement of a mobile robot. *Moscow Workshop on Electronic and Networking Technologies, MWENT 2020 - Proceedings*, 17–20. https://doi.org/10.1109/MWENT47943.2020.9067482

[15] Tajjudin, M., Johari, S. N. H., Aziz, S. A., & Adnan, R. (2019). Minimum ISE Fractional-order PID (FOPID) Controller for Ball and Beam Mechanism. *ICSGRC 2019 - 2019 IEEE 10th Control and System Graduate Research Colloquium, Proceeding*, *August*, 152–155. https://doi.org/10.1109/ICSGRC.2019.8837071

[16] v. M. buyanov. (1967). 済無No Title No Title No Title. *Angewandte Chemie International Edition*, *6(11)*, *951–952*.