International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 2 (2018) pp. 995-1000

Neural Network for Pattern Recognition and Application to a Differential Drive Robot Path Planning

[1]Juan Manuel Triana, [2]Dario Amaya and [3]Olga Ramos
Universidad Militar Nueva Granada, Bogotá, Colombia.

Abstract

The neural network paradigm has permitted computers to learn from observational data, providing solutions in image, voice or language recognition, based on the human biology and the way we learn. In this work, a *Backpropagation* neural network was developed with the purpose of identifying patterns, specifically numbers from 1 to 5 taken from a picture, and using them to make the path planning for a differential drive robot. For this purpose, different software's such as Solidworks for the mechanical design of the robot and Matlab for the neural network code and the Simscape multibody link plug-in, were used to simulate and get results. This work shows the learning process and the solutions given by the network and how the path planning works.

Keywords: Artificial Neural Network, back propagation, patterns, path planning.

INTRODUCTION

Artificial neural networks are a learning and processing of information paradigm based on human biology [1]. Therefore, it tries to provide solutions in different image or language recognition problems. It's about an interconnected system of neurons that help each other to produce an output impulse [2].

Robotics are involved in this concept as possible receivers of artificial intelligence. Path planning in differential drive robots are an example of this, taking the robots from an initial to a final state considering the velocity in each wheel and how this change cartesian and angular positions [3].

Neural networks can be applied to different fields, especially the ones involved with pattern recognition. Computer science worked the definition of a pattern considering its environment and how to train a network to learn from it. Data mining, web search, facial recognition, among others, are some of the recent applications developed in this field [4].

Pattern recognition embraces a wide range of disciplines that recognize neural networks as a possible solution. Progress has been made in Security field, as in [5], where pattern recognition approach was made using a feedforward neural network, or [6], where a backpropagation network was used on an image filtered using edge detection to segment the picture. In [7], image processing is used to make plate identification in India.

Moreover, another kind of works with optimization algorithms were made, as in [8], where a particle swarm optimization algorithm was developed to vary the network parameters and reach high performance in pattern recognition, giving better results than a traditional network.

In addition to character recognition using images, or written texts as in [9] or [10], there are also neural networks developed for facial recognition, currently applied in different countries. In [11], a Kohonen self-organizing map neural network was used for facial recognition, accomplishing an 85% range of identification with a non-supervised learning.

Along with pattern recognition, a lot of applications can be unleashed. Some studies with neural networks are applied directly to differential drive robots, as in [12], where a backpropagation network predicts the robot displacement based on its geometrical center.

Some of this robot use neural networks as a controller, as in [13], where each of the wheels of the differential drive robot has its own network, allowing the robot to have 4 different behaviors. Additionally, there are some studies about path planning for robots using neural networks, as in [14], where the network finds the shortest path and without obstacles.

This works aims to train a neural network, based on patterns of numbers from 1 to 5, using an image taken from a camera. Once the numbers are identified, an algorithm does the path planning for the differential drive robot finding the shortest path and printing the distances and angles the robot must move.

METHODS AND MATERIALS

As shown in Figure 1, a camera takes a picture of random numbers from 1 to 5 and a backpropagation neural network identifies the numbers. Once the network finishes, an algorithm does the path planning for the differential drive robot, going through the shortest path between the possible numbers.

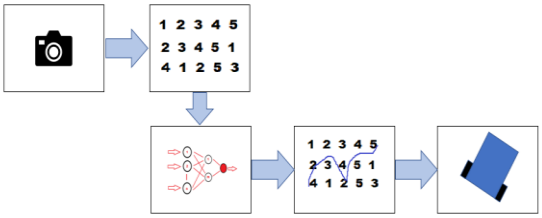

Figure 1. Outline scheme.

- ***Kinematic model of a differential drive robot***

The kinematics of the differential drive robot depends on the changes presented in their locomotives systems, in this case, it

refers to the engines and the wheels of the platform. A scheme that represent the mentioned model is shown in figure 2, where the platform has an (x, y) position and a displacement velocity in each wheel.

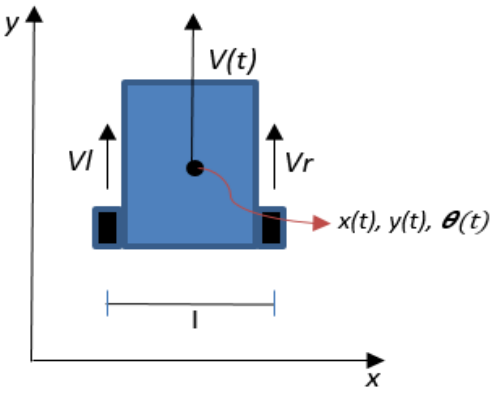

Figure 2. Differential drive robot and its reference system.

Based on the scheme, v_l y v_r represent the linear velocity in each of the locomotive system, which depends directly from the angular velocity and the wheel radius. Equations 1 and 2 model this behavior and 3 presents the average linear speed of the robot.

$$v_l = w_l\, r \qquad (1)$$

$$v_r = w_r\, r \qquad (2)$$

$$v = \frac{v_r + v_l}{2} = \frac{(w_r + w_l)r}{2} \qquad (3)$$

Knowing the robot is always moving in a cartesian reference system, at every moment Will have a coordinate *(x, y, θ)*. Equations 4-6 shows the velocity at each of this coordinate points based on the average linear speed and angular speed.

$$\dot{x} = v(t)\ \cos(\theta(t)) \qquad (4)$$

$$\dot{y} = v(t)\ \sin\big(\theta(t)\big) \qquad (5)$$

$$\dot{\theta} = w(t) \qquad (6)$$

With equations 4-6, and defining a Δt, position of the robot can be foud using its reference cartesian system, equation 7-9 model this behavior.

$$x(t) = x(t_0) + \int_{\Delta t} v(t)\cos\big(\theta(t)\big)\, dt \qquad (7)$$

$$y(t) = y(t_0) + \int_{\Delta t} v(t)\sin\big(\theta(t)\big)\, dt \qquad (8)$$

$$\theta(t) = \theta(t_0) + \int_{\Delta t} w(t)dt \qquad (9)$$

- ***Differential Drive Robot:***

To build the robot it's necessary a couple of motors, encoders, wheels, caster wheel and a platform. In figure 3 it is shown the design made using Solidworks®.

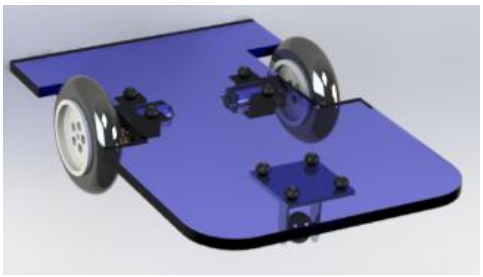

Figure 3. Differential Drive Robot design (CAD)

The measures of the robot are *180x120mm*, wheels radius is *60mm* and the distance between wheels is *110mm* long.

Using Matlab plug-in Simscape Multibody Link an XML file is created with the properties of the CAD file created in Solidworks. Figure 4 is a model of each of the components of the assembly in Simulink, where blocks can be created as solid or rigid transformations. Solid blocks provide the body geometry, inertia and colors while the rigid ones have the connections between bodies. There are also constraints defined, in this case, cylindrical ones.

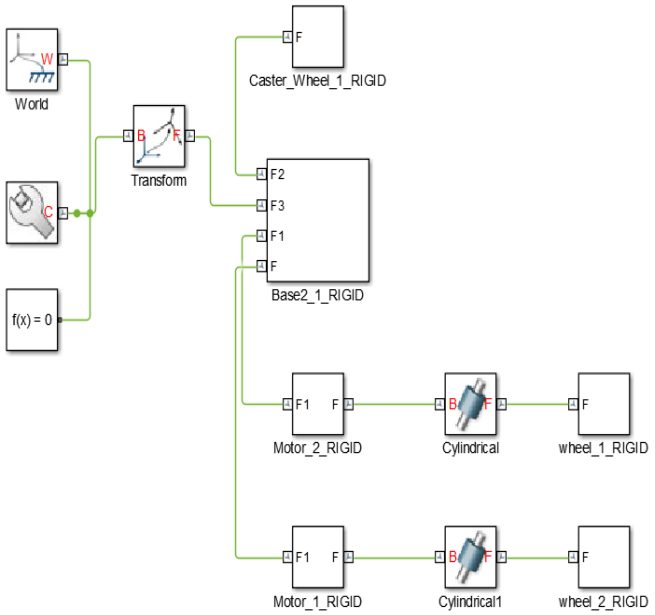

Figure 4. Simulink Model.

By executing the simulation, some behaviors can be observed depending on which actions the program receives. In this case, the wheel velocity can be changed to the system.

- ***Neural Network design:***

For the back propagation neural network design some aspects must be defined, for example its learning algorithm and its paradigm. In figure 5 is a basic scheme of a neural network, it has an input layer, a hidden layer and an output layer.

The learning and training process consists on refreshing weights in the network. In this work, an LMS (*Least Mean Square*) algorithm is used to train the network. This algorithm is based in gradient descent, reducing the difference between the real output and the desired one. The algorithm uses an activation function which takes weights and error values to make its calculations.

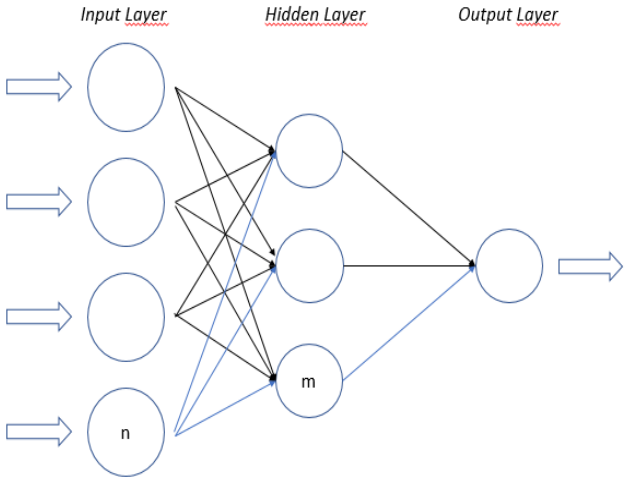

Figure 5. Artificial neural network with n input neurons, m hidden neurons and an output neuron.

Equation 10 shows how the new weights are calculated based on the previous weights, the learning factor (α), the error and the inputs.

$$w(k+1) = w(k) + \alpha * e(k) * \frac{x(k)}{|x(k)|} \qquad (10)$$

In turn, equation 10 uses an error defined as the difference between the desired output and the actual one, as is shown in equation 13. To clarify, getting the actual output is necessary to use equation 11, where each of the weights is multiplied with the respective input, and using the result, and equation 12, which uses the result of 11 to calculate the output.

$$s = \sum W_n * X_n \qquad (11)$$

$$Y = tanh(s) \qquad (12)$$

$$e(k) = Y_d - Y \qquad (13)$$

Besides the weight calculation, the algorithm generates a term error, the one in charge to stop the training when the net reaches the minimum error desired. Equation 14 shows how the mean squared error is calculated.

$$E = \frac{1}{2} \sum (e_i)^2 \qquad (14)$$

- ***Neural Network:***

On the basis that the external observable data is a picture taken from a camera with 640x480 resolution, and that the matrix used for the training is 35x100 sized, each pattern has a 7x5 dimension in pixels. The 35 rows created correspond to the input and the 100 columns refer to the total of patterns entered. As it was told, the patterns correspond to numbers from 1 to 5, so there are 20 examples for each number. The output matrix has a 5x100 dimension. The neural network proposed for this work is a backpropagation net.

Initially, an interface is created using Matlab guide with two operation modes, the first operation mode enables the user to create patterns and save them while the second one uploads the matrixes saved on a ". *mat*" file.

One of the tools the interface have is the one to export the results to an Excel file. The figure 6 shows some of the patterns saved from the Matlab interface and let the user observe the results.

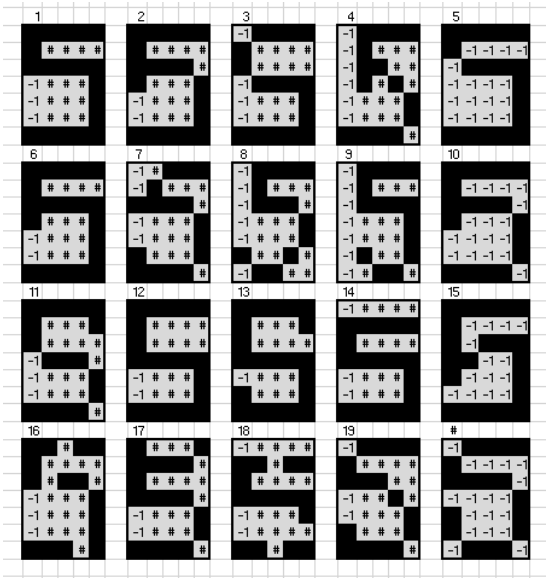

Figure 6. Excel saved numbers.

For the network training using the saved patterns, a minimal error is defined and the total neurons to be used in the hidden layer.

RESULTS ANALYSIS

While testing the neural network from the interface using the saved patterns, it is evident that if the defined minimal error is too small, the network has troubles to get out from the training loop, that's why the minimum error used was 5%, letting the net to obtain its weights.

Also, the number of neurons used in the hidden layer is directly related to the iterations needed for the net to decrease its error. For the case shown in figure 7, a 5% error was used along with 35 neurons in the hidden layer, printing a chart of the error versus the iterations required to stop its training.

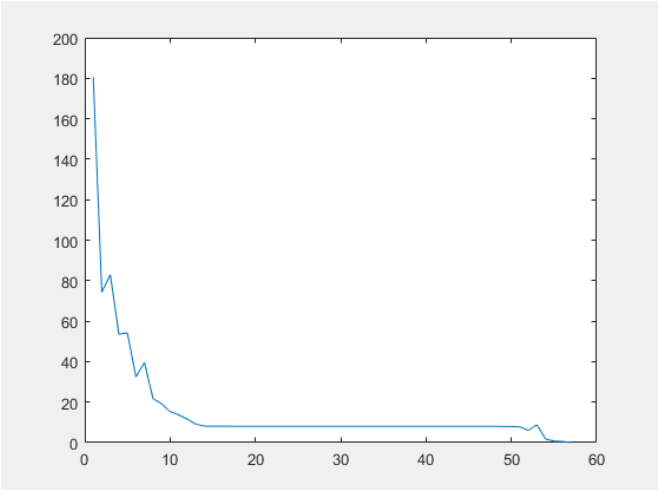

Figure 7. Error plot.

After the net is trained, some tests are run on the interface to verify its functionality giving at the input numbers from 1 to 5. The program gives an output vector with a percentage of the similar numbers identified.

For example, when a number 1 figure was entered to the net, the output vector gives a coincidence of 99.98% with the number one, while testing another number, like 4, some similarities with other numbers were found, like number 5. Some of the pixels in this numbers are located on similar positions and the net finds a coincidence.

As mentioned before, the number recognition starts with a picture taken from a camera, on which the path planning will work. Figure 8 shows the image captured.

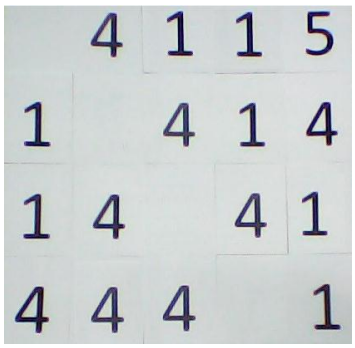

Figure 8. Captured Image.

With the picture captured, the image is uploaded to Matlab to make the filter and image processing. The first step is turn the image into a binary one with the command *im2bw.* This command does not have a mathematical model, is just based on a logical process, where a limit id defined, and the values below are replaced with a 0 and the ones above are replaced with a 1. The figure 9 shows the result of this filter using a pre-determined limit value of 0.5.

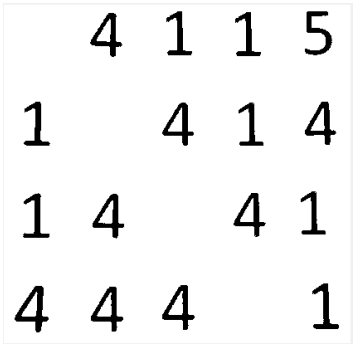

Figure 9. Binary Image.

After the image is turned to binary values, the command *regionprops* is used to measure the properties of each component in the image. To enclose the numbers is used the property Bounding Box, and based on this command, using Centroid, is possible to know the *x, y pixel* location of every component in the image. In figure 10 its shown how the command Works with every iteration that passes, finding the characters and boxing them in blue rectangles. Once the procedure is finished, a vector stores the centroid information of every component.

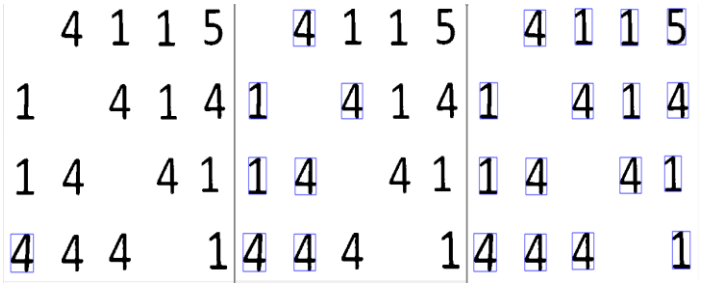

Figure 10. Object identification.

As the camera resolution is 640x480, the patterns need to be re-scaled to a 7x5 space with the purpose of entering this characters to the neural network and start their identification.

Because of the scale used, the identification results were below speciation's, so the net was trained again but this time considering some similar patterns with the ones captured from the image. The results of the second training are shown in figure 11, a plot of the error, where the iterations needed to stop the training were less compared to the first training.

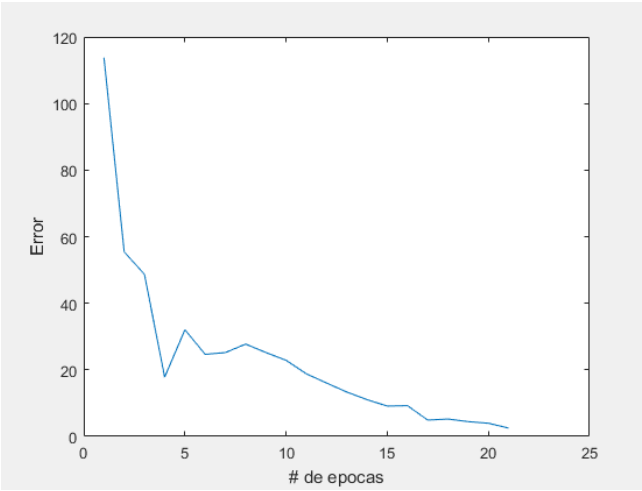

Figure 11. Second training, error plot.

After the second training, the net identifies the numbers from the picture and generates a vector where numbers 4 and 5 are possible positions for the path planning, so are replaced with a 1, and the non-possible positions, number 1, are replaced in the vector with a 0. After this, a comparison is made with the centroids vector, knowing the exact points where the robot can pass through.

The path planning algorithm evaluates the possible centroids and the distance between them, selecting the minimum distance presented. Figure 12 shows how the algorithm plans the route, passing through numbers '4' in the shortest path, and arriving to the final position, number '5'.

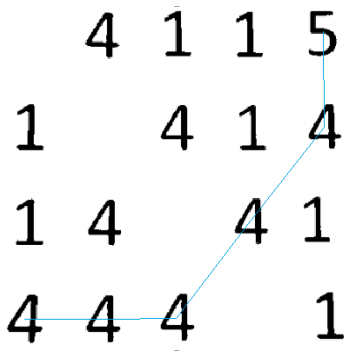

Figure 12. Path planning

The distance is calculated in using the pixel difference between the centroids, and, knowing the direction, the angles are calculated at each point. Considering the robot has already an orientation when arrives to the next point, the information send to the robot is the angle difference between the current orientation and the next one.

For the physical implementation, a conversion from pixels to distance in *cm* is found with experimental tests using the encoders.

CONCLUSIONES

In this article it's been discussed a pattern recognition approach to make the path planning for a differential drive robot. By using a backpropagation neural network, identifying numbers from 1 to 5 taken from a camera with a resolution of 640x480 was possible, showing that is feasible to use this as a possible solution for characters identification.

The balanced usage of neurons per layer and patterns to train leads to a proper learning for the network. Also, the tests show that the path planning algorithm works and is simple to implement.

As a future work, camera should have a better resolution and not only work with one image as an input, it could also work with video as an entrance.

ACKNOWLEDGEMENTS

The authors would like to offer their special gratitude to the Nueva Granada Military University for supporting this work as a result of scientific initiation at research group 'GAV', 2017.

REFERENCES

[1] M. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

[2] R. Salas, «Redes neuronales,» *Universidad de Valparaiso,* pp. 1-15, 2004.

[3] E. Mariscal-García, «Sistema de Información Científica Redalyc,» 2005. [En línea]. Available: http://www.redalyc.org/comocitar.oa?id=46110112. [Último acceso: 12 Diciembre 2017].

[4] K. Tai-hoon, «Pattern Recognition Using Artificial Neural Network:,» de *International Conference on Information Security and Assurance*, Miyazaki, 2010.

[5] R. Yarisi, E. Di Claudio y G. Lucarelli, «Experimental car plate recognition system by neural networks and image processing,» de *Signal Processing Conference (EUSIPCO 1998), 9th European*, Rhodes, 1998.

[6] N. H. Barnouti, M. A. Sahib Naser y S. S. Mahmood Al-Dabbagh, «Automatic Iraqi license plate recognition system using back propagation neural network (BPNN),» de *New Trends in Information & Communications Technology Applications (NTICT)*, Baghdad, 2017.

[7] A. Agarwal y S. Goswami, «An Efficient Algorithm for Automatic Car Plate Detection & Recognition,» de *2016 Second International Conference on Computational Intelligence & Communication Technology (CICT)*, Ghaziabad, 2016.

[8] M. I. F. Maruzuki, S. N. Ishak y S. Setumin, «Malaysia car plate recognition based on RBF Neural Network and

Particle Swarm Optimization,» de *Control System, Computing and Engineering (ICCSCE)*, Penang, 2012.

[9] M. Rajnoha, R. Burget y M. Kishore Dutta, «Handwriting comenia script recognition with convolutional neural network,» de *Telecommunications and Signal Processing (TSP), 2017 40th International Conference*, Barcelona, 2017.

[10] M. M. Abu Ghosh y A. Y. Maghari, «A Comparative Study on Handwriting Digit Recognition Using Neural Networks,» de *Promising Electronic Technologies (ICPET)*, Deir El-Balah, 2017.

[11] S. Ghorpade, J. Ghorpade y S. Mantri, «PATTERN RECOGNITION USING NEURAL NETWORKS,» *International Journal of Computer Science & Information Technology (IJCSIT),* vol. 2, nº 6, pp. 92-98, 2010.

[12] S. Jia, Q. Qiu y J. Li, «BP neural network based localization for a front-wheel drive and differential steering mobile robot,» de *Information and Automation*, Lijiang, 2015.

[13] J. C. Vega y P. Humaní, «Neural control to simulate 4 autonomous navigation behaviors in a differential-drive mobile robot,» de *URUCON*, Montevideo, 2017.

[14] L. Chaomin, S. X. Yang y M. Krishnan, «An effective vector-driven biologically-motivated neural network algorithm to real-time autonomous robot navigation,» de *Robotics and Automation (ICRA)*, Hong Kong, 2014.