Research Article

Open Access CrossMark

Acquisition and recognition of ultrasonic signatures using multi-layer neural network

Abstract

The echo captured by an ultrasonic sensor provides details of the geometric shape of objects, this ability common among dolphins and bats is known as echolocation. The signal detected by an ultrasound sensor contains, in addition to the distance from the object, characteristics such as the type of material and internal cracks. This article describes a method for acquiring ultrasonic signals and recognizing them as geometric shapes through the use of a neural network processed by the TensorFlow JS library. The results using basic geometries (rectangular, triangular and spherical parts) showed that TensorFlow JS recognizes their ultrasonic signatures, being an additional solution to those that use cameras for object recognition.

Keywords: echolocation, neural network, tensor flow, ultrasonic signature

Volume 6 Issue 3 - 2020

Maicol Peterson Gandolphi de Almeida, Pedro Bertemes-Filho
State University of Santa Catarina, Brazil

Correspondence: Pedro Bertemes-Filho, Universidade do Estado de Santa Catarina, Rua Paulo Malschitzki 200, Zona Industrial Norte, 89219-710, Joinville, Santa Catarina, Brazil, Tel +55 47 34817848, Email pedro.bertemes@udesc.br

Received: August 05, 2020 | **Published:** August 31, 2020

Introduction

Ultrasound systems used for object recognition and for the navigation of blind people through the environment are mostly composed of walking sticks that vibrate when approaching an object. In addition to vibrating systems, sound emission systems are also used to aid navigation for blind people,[1] these systems are composed of cameras combined with ultrasonic sensors, only cameras or only ultrasonic sensors.[2] A literature review by Almeida shows that from 2016 onwards there was a drop in the number of studies using ultrasound as the main means of object detection. The author explains that this drop is due to the fact that technologies using cameras for the recognition of objects and people are drawing the attention of the scientific sphere and more researches in this area are being developed.[2] Ifukube[3] performed tests for detecting an object by the generation of sounds and the use of ultrasonic sensors. He showed the importance of hearing low and high frequency sounds in order to improve the detection of the object's position. Simultaneously, studies such as Nabhani's[4] & Manju's[7] & Velappa's[6] used neural networks for object recognition, having Nabhani used cameras and Manju and Velappa used ultrasonic sensors. These authors showed that the use of neural networks is a viable method for identifying objects.[4,5] Neural networks are architectures used for machine learning, their main block is the neuron, a small functional unit. The neuron has an input and an output that is calculated through an activation function and a bias. The combination of several neurons forms the neural network with its activation functions, bias and weights. Figure 1 shows a neuron in the input layer with its input variables (X1, X2, ..., Xn), weights (P1, P2, ..., Pn), correction value (Bias B1), output intermediate value (V1), activation function (F) and output value (Y1).

The V1 value is calculated using Eq.1 and then the chosen activation function is applied, resulting in the Y1 output.[7]

$$V_1 = \Sigma(X_n P_n) + B_1 \quad n=1, 2, 3, \dots \text{ (Eq. 1)}$$

The input variables values in a neural network must be normalized for a better processing performance,[8] Eq. 2 is used to normalize the input data in a neural network, suppose that it is desired to normalize values between 2cm and 400cm (the minimum and the maximum that an ultrasound sensor model HC-SR04 can measure, for example) and transform them into 0.0 and 1.0:

$$D_N = (D - 2{,}0) / (400{,}0 - 2{,}0) \text{ (Eq. 2)}$$

Where D is the real distance and DN the normalized distance.

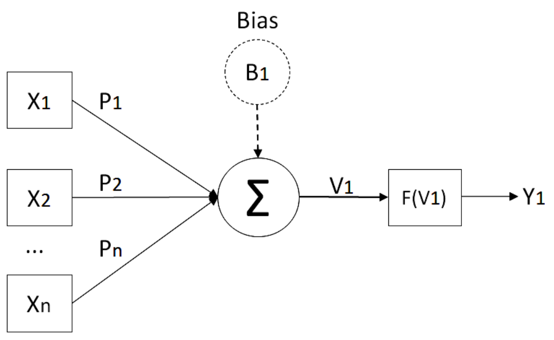

Figure 1 Neuron model in the input layer of a neural network.

The F activating function of a neural network depends on the problem to be solved and also on the layer in which the neuron is at. It is possible to create new activation functions, but there are already widely used functions for several artificial intelligence problems that are effective, among them are the sigmoid function, ReLU (Rectified Linear Unit), Leaky ReLU and Softmax.[7,8] The sigmoid function and its combinations are broadly used in classification networks, whereas the ReLU function is applied to the hidden layers of the network (neurons positioned between the input and the output layer). The Leaky ReLU function is generally used in cases where the ReLU function has not given good results.[8] Similar to the sigmoid function, Softmax is useful in classification problems of several classes, resulting in the probability that an input from the network belongs to one of these classes.[8] The mathematical model in Figure 1 is known as Perceptron, this model was developed in the 1950s and 1960s by the scientist Frank Rosenblatt and allows a clear understanding of how a neural network works in mathematical terms.[8,9]

Rosenblatt proposed a simple rule for calculating the output, multiplying the input values by their respective weights, adding these results and applying a rule for the output value: 0 if the sum is less than or equal to a threshold value or 1 if the sum is above the threshold value. By varying the weights and the threshold, different decision-making can be reached.[9] The bias value defines the neuron sensitivity for supplying 0 or 1, that is, the higher the Bias (B1) value, the easier it is for the Perceptron to emit a value of 1 and the lower the Bias is, the greater the difficulty is for giving off a value of 1.[9] When a Perceptron has at least one hidden neuron layer (neither input nor output) it is called a Multilayer Perceptron (MLP), this architecture has the advantage of being an universal function approximator, meaning that with the use of weights and well-distributed neurons, the desired calculation of any function is achieved.[8–10] Determining the weights and bias of a neural network is called network training; in the case of a MLP network the training is supervised, that is, when providing known values of input and output, weights are determined. An error backpropagation algorithm is used. This algorithm has two phases: feed forward and backpropagation. In the first phase the output of the network is calculated following the neuron flow, in this phase the weights are not changed. The second phase shown in a simplified way in Figure 2 uses the Y output of the network and compares it with the correct answer, resulting in an error variation (Δε). The W weights are readjusted in the reverse order (backpropagation) until the answer given by the network approaches the real value, ending the supervised training of the network.[8–10] In Figure 2 the Δε error is propagated proportionally for each weight P, Eq. 3 shows how to recalculate the new weight NPn, i from the hidden layer output, where n represents the nth neuron of the layer, i represents the nth exit of this neuron and ΣPO is the sum of all weights P of the hidden layer.[7]

$$^{N}Pn,i = \Delta\varepsilon \cdot (Pn,i / \Sigma P_{O}) \text{ (Eq. 3)}$$

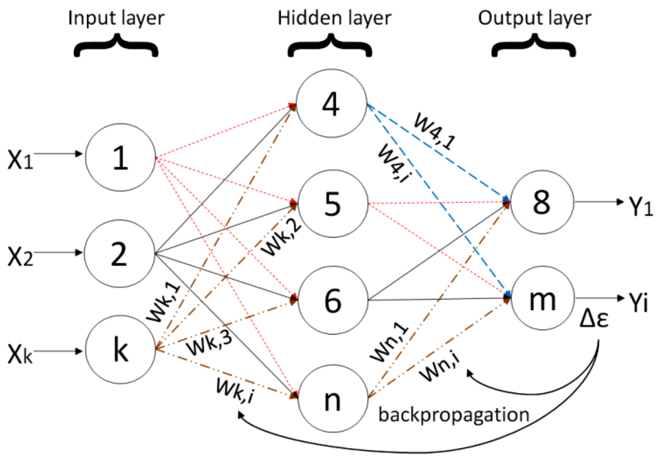

Figure 2 Example of a multilayer perceptron network using the Δε error for backpropagation to adjust the weights.

After training the network, new input variables are provided and based on the weights already defined, the network classifies the output according to the training standards used, so the greater the data for training the network, the greater the assertiveness in its classification. For the training and development of neural networks there are some open source libraries, such as the Brain.js, Synaptic.js and TensorFlow JS that run directly on the internet browser using the JavaScript language, according to the available examples on the development team website.[11] Among the programming environments for Arduino Atmega328p and Atmega168 microcontrollers, the Arduino IDE software[12] and the Johnny-Five platform are on the spotlight due to the ease for installing and being free of charge. Distance sensors are used in conjunction with microcontrollers to send data to the user's screen or other desired output. Through these programming interfaces it is possible to program microcontrollers through the browser[13] or natively by uploading the program into the Arduino board.[12]

Material and methods

A HC-SR04 SONAR module with a measurement ranging from 2cm to 400cm and a resolution of 3 mm was used. This module operates with a 5V voltage, a 40 Hz frequency and a 15mA working current. Together with this module, the Arduino Nano module was used, operating at a 5V voltage, a maximum current of 40 mA, a 16 MHz crystal oscillator, with 14 digital pins and 8 analog pins. The programming for reading the ultrasonic sensor is done on the Johnny-Five platform due to the ease of integrating it with the browser, therefore all code execution was centralized in a single environment, in this case in the Google Chrome browser. The TensorFlow JS library together with the JavaScript language were chosen for training the neural network and three different geometric shapes were tested: cubic, triangular and spherical. The inputs for training the network are made with 20 theoretical values of normalized distances between 0 and 1. Table 1 show the input and output values for the training of the three chosen geometries (values in parentheses are in centimeters). The output [1,0,0] represents the cubic object, [0,1,0] the triangular object and [0,0,1] the spherical one. After training the network, the sensor is connected to the Arduino module and it receives the distance information from the object's surface. The measurement was performed by moving the sensor from a 50cm distance from the object and in a single direction in order to acquire the ultrasonic signature through 20 measurements. The measurements were programmed to be made every 100ms.

Table 1 Values for training the multilayer perceptron neural network

Input	Cube	Triangle	Sphere
1	0,246 (100)	0,246 (100)	0,246 (100)
2	0,246 (100)	0,246 (100)	0,246 (100)
3	0,246 (100)	0,246 (100)	0,246 (100)
4	0,246 (100)	0,246 (100)	0,228 (93)
5	0,246 (100)	0,246 (100)	0,211 (86)
6	0,045 (20)	0,195 (80)	0,193 (79)
7	0,045 (20)	0,145 (60)	0,183 (75)
8	0,045 (20)	0,095 (40)	0,165 (68)
9	0,045 (20)	0,045 (20)	0,160 (66)
10	0,045 (20)	0,095 (40)	0,155 (64)
11	0,045 (20)	0,145 (60)	0,158 (65)
12	0,045 (20)	0,195 (80)	0,163 (67)
13	0,045 (20)	0,246 (100)	0,168 (69)
14	0,045 (20)	0,246 (100)	0,173 (71)
15	0,045 (20)	0,246 (100)	0,183 (75)
16	0,246 (100)	0,246 (100)	0,201 (82)
17	0,246 (100)	0,246 (100)	0,213 (87)
18	0,246 (100)	0,246 (100)	0,233 (95)
19	0,246 (100)	0,246 (100)	0,246 (100)
20	0,246 (100)	0,246 (100)	0,246 (100)
Output	[1,0,0]	[0,1,0]	[0,0,1]

Results and discussions

The values shown in Table 1 are represented in the graphs of Figure 3A–3C these values are theoretical and refer to 20 values of an ultrasonic sensor moving over a cube (a), triangle (b) and sphere (c). The network training was performed with 20 neurons in the hidden layer, 0.5 learning rate and 500 iterations. The activation function used in the hidden layer was ReLU and Softmax in the output one. Figure 4 shows the error rate graph during classification by the neural network. The error rate after 500 epochs reached 0.008. After training the network, practical tests using the ultrasonic sensor were performed using the Johnny-Five library and the JavaScript language. Through this library, communication was established between the Arduino and the browser for inputting the normalized data into the neural network. The first object tested was an object in the form of a rectangular block of dimensions 0.8 x 7.0 x 14.0cm. The purpose of this test was to verify whether the network's output values tend to classify this object as a cubic shape. The output values for this test classified the object as cubic, the second classification was spherical. The second object tested was an equilateral triangle with a 15.0cm side. For this test the network classified the object as triangular and in the second classification the probability was that of a spherical object. The third object tested was a 20cm diameter soccer ball. This time the net classified the ball first as spherical and the second probability was the triangular shape.

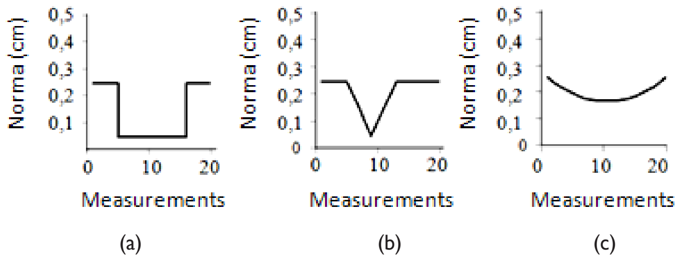

Figure 3 Representation of 20 theoretical measurements of the normalized distances of a sensor moving over a cubic (A), triangular (B) and spherical (C) object in 100ms intervals, resulting in a total time of 2s for each object.

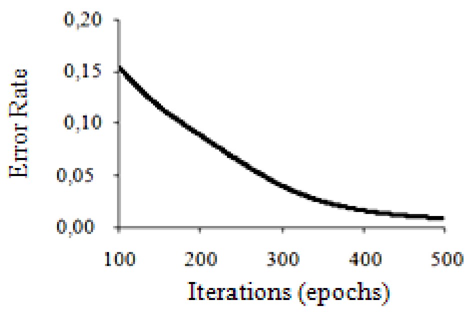

Figure 4 Error rate graph between 100 and 500 iterations.

All tests were performed at a maximum distance of 50 cm from the sensor and repeated 3 times. The program reads 20 distances and if at any time the object is removed from the front of the sensor or the distance is above 50 cm, then the program redoes the 20 readings from the moment it detects the object again. When triangular objects of small inclination were used in this work, the network have classified them as spherical objects. Similarly when using smaller spheres, in which some objects were classified as triangular objects. It has to be emphasized that the size of the objects is a drawback in the classification process. Therefore, it is recommended to train the neural network with various spherical and triangular objects of different dimensions and curvatures. The geometric shape can be better identified by using ultrasonic sensors for detecting an object, even not without using neural network.[3] On the other hand, neural networks can handle and process a huge amount of data with different object's shape and 'size in a real in a real time application. Though, it is expected to use different geometric shapes and to combine them to evaluate how the network will classify these combinations.

Conclusion

The use of neural networks to solve problems on classifying geometric shapes through ultrasonic signatures is a method that provides results for the 3 shapes that have been used (cubic, triangular and spherical). For shapes that vary slightly in slope, the values at the network output (network classification) tend to be close, in that case increasing the input values for training the network or increasing the number of network neurons are some solutions to solve this problem. According to our literature review, most researches in this filed identify objects by using cameras or other techniques, which are different to the one presented in this article. We foresee the use of ultrasonic signatures combined with neural networks and sound generation as a promising technique to be tested on blind people to ascertain its effectiveness in terms of object identification.

Acknowledgments

We thank the State University of Santa Catarina (UDESC) for the institutional support and the Foundation for Research and Innovation of Santa Catarina State (FAPESC) for the financial aid.

Conflicts of interest

The authors declare that there is no conflict of interest.

References

1. Pooja P, Gundewar, Emant K Abhyankar. A Review on an obstacle detection in navigation of visually impaired. *International Organization of Scientific Research Journal of Engineering*. 2013;3:01–06.
2. Almeida M. Recognition ultrasound systems for assisting blind people. XXII Congreso Argentino de Bioengenharia e II Jornada de Engenharia Clínica – SABI; 2020 Mar 4-6; Piriápolis, Uruguay. 2020.
3. Ifukube T, Sasaki T, Peng C. A blind mobility aid modeled after echolocation of bats. *IEEE Transactions on biomedical engineering*. 1991;38(5):461–465.
4. Nabhani F, Shaw T. Performance analysis and optimisation of shape recognition and classification using ANN. *Robotics and Computer-Integrated Manufacturing*. 2002:177–185.
5. Manju J, Meera A, Sahna A. Ultrasonic visual aid using ANN. *International Journal of Scientific & Engineering Research*. 2014;5.
6. Velappa G, Soh CY, Jefry N. Fuzzy and neural controllers for acute obstacle avoidanc in mobile robot navigation. *In: Proceedings of the IEEE International Conference on Advanced Intelligent Mechatronics*. 2009;7:1236–1241.
7. Nascimento J, Cairo L, Yoneyama T. Inteligência artificial em controle e automação. São Paulo. Edgard Blücher ; 2000.
8. Data Science Academy. Deep Learning Book. 2019.
9. Silva LNDC. Análise e síntese de estratégias de aprendizado para redes neurais artificiais. Master's dissertation, São Paulo. State Univetsity of Campinas; 1998.

10. Filho JBDF. Base Teórica para o processamento neura-adaptivo de sinais, Master's dissertation. São Paulo. State Univetsity of Campinas. 1994.
11. Abadi M, Barham P, Chen J. et.al. Tensorflow: A system for large-scale machine learning. *In 12th USENIX Symposium on Operating Systems Design and Implementation*. 2016;265–283.
12. Blum J. Exploring Arduino: tools and techniques for engineering wizardry. John Wiley & Sons; 2019.
13. Johnny-Five: The JavaScript Robotics & IoT Platform.