Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 8, No. 3, September 2020, pp. 465~475
ISSN: 2089-3272, DOI: 10.11591/ijeei.v8i3.2078

Obstacle Evasion Algorithm Using Convolutional Neural Networks and Kinect-V1

Paula Catalina Useche Murillo, Javier O. Pinzón-Arenas, Robinson Jiménez-Moreno
Universidad Militar Nueva Granada, Colombia

Article Info

Article history:
Received Jan 15, 2020
Revised Jun 11, 2020
Accepted Jul 14, 2020

Keyword:
Object Recognition,
Convolutional Neural Networks,
Kinect-V1,
RGB-D Information,
Obstacle Evasion,
Path Planning

ABSTRACT

The following paper presents the development of an algorithm for the evasion of static obstacles during the process of gripping the desired object, using an anthropomorphic robot, artificial intelligence, and machine vision systems. The algorithm has developed to detect a variable number of obstacles (between 1 and 15) and the grip desired element, using a robot with 3 degrees of freedom (DoF). A Kinect V1 was used to capture the RGB-D information of the environment and Convolutional Neural Networks for the detection and classification of each element. The capture of the three-dimensional information of the detected objects allows comparing the distance between the obstacles and the robot, to make decisions regarding the movement of the gripper to evade elements present in the path and hold the desired object without colliding. Obstacles of less than 18 cm in height were avoided, concerning the ground, with a probability of collision of 0% under specific environmental conditions, moving the robot since initial path in a straight line to the desired object, which is prone to changes according to the obstacles present in its. Function tests have been according to the manipulator's ability to evade possible obstacles of different heights located between the robot and the desired object.

Corresponding Author:
Paula Catalina Useche Murillo,
Universidad Militar Nueva Granada,
Av cra 11 No 101-80, Bogotá, (Colombia),
Email: {u3900235: u3900231;robinson.jimenez}@unimilitar.edu.co

1. INTRODUCTION

Robotics has allowed automating different types of industrial processes, such as painting, welding, inspecting products, assembling on production lines, among others, as described in [1], achieving a high level of precision, speed, and resistance to adverse external factors in each of their jobs. However, the working environment of the robot is not always constant, and static or dynamic obstacles may appear, with which it will be necessary to establish techniques to avoid any collision during the performance of its task. Given this premise, different types of applications have been made that seek to generate obstruction-free trajectories, both for mobile robots and robotic manipulators, such as those presented below.

In the application developed in [2], a path planning has been performed for mobile robots in a work environment with obstructions, where the Flood Fill method was used to generate the movement of the mobile, by filling the workspace with numbers sorted in ascending order, starting with 1 in the areas adjacent to the robot, until reaching the point of arrival, with a one-by-one increase in numbering. A discrete division of the workspace was generated to be able to use the method, and an improvement was made in the trajectory obtained to reduce the amount of displacement of the mobile. In other cases such as the one developed in [3], a process of obstacles evasion for navigation of a mobile robot is presented based on machine vision systems, the environment is captured by a camera and the obstacles are detected by image processing techniques, this navigation method is susceptible to lighting changes and dynamic objects presence, both aspects can be improved by RGB-D camera use and change the obstacle detection methods with intelligent algorithms.

On the other hand, in [4], the control of a picking manipulator was performed using IOT remote control technology, for monitoring, simulation and control, and a real-time control system by PLC, to generate obstacle evasion, using artificial potential fields to determine the degree of repulsion between the manipulator and the obstacle.

For the cases shown in [5-8], different obstacle avoidance techniques were used for manipulators of different degrees of freedom and against different types of obstructions using techniques such as calculating the distance between the robot and the obstacles using the Gilbert-Johnson-Keerthi algorithm, and discrete detection of the environment in real-time with a Kinect-V2, as it was done in [5]. However, the recognition and identification of the work area where a manipulator moves are necessary to generate collision-free movements, but the method presented in [5] is not oriented to recognized objects and the task of gripping this in the same area of work the same distance, from a table, can, for example, may be generated mistakes in the recognition based only in deep information. In [6] an efficient method for object recognition is used for human-computer interaction using a Kinect sensor, with a great result to discriminate object at the same distance, but the deep information is not used, the autonomous robot proposed by the authors is limited to task for obstacle evasion.

In [7-8], the process of evasion of obstacles was focused to redundant manipulators, where the first one developed a backward quadratic search algorithm (BQSA) which is responsible for detecting possible collisions in the robot, evaluating it from the end effector to the base, enclosing obstacles in ellipses, and storing the positions of the endpoints of each manipulator link. In the second case, a minimum jerk norm scheme was used (MJN) with restriction of obstacle avoidance and improvement by feedback control, where efficiency in obstacle avoidance was guaranteed using the variable-magnitude escape jerk theorem. In the latter case, a fuzzy adaptive control was used for the displacement of an industrial robot with uncertain dynamics, achieving the evasion of obstacles during the movement, using the calculation of the Euclidean distance between the robot and the surrounding elements, here techniques of artificial intelligence are used in the algorithms.

Therefore, it is essential to use machine vision methods, such as RGB-D cameras [19] and artificial intelligence techniques [20], to extract information from the environment and deliver it to the obstacle evasion algorithm for decision making, taking in consideration deep information and efficient recognition methods, necessary for the robust task of obstacle evasion.

Some of the techniques used for the recognition of elements in a work environment are the Convolutional Neural Networks (CNN), which are neural networks designed for the classification of patterns or objects in images, through the use of convolution layers, as explained in [9]. From these networks, the Region-Based Convolutional Neural Network (R-CNN) were designed which use an Edge Boxes Algorithm [10] for the detection of the elements to be classified, which extracts from the image to be evaluated multiple detection boxes, and classifies them using a CNN, as explained in [11].

In [12], for example, an CNN was used for assistance robot application, where the anthropometric robot can recognize a group of tools and the hand of a user, delivering one of them in the hand. The application no use RGB-D cameras and no can be able to avoid obstacles, need a free path between objects and the user to operate.

Below, it is presented an application developed for the evasion of obstacles for an anthropomorphic robot, whose purpose is to take the desired object and to move it to a delivery area stipulated by the user, without colliding with any type of element during the movement. To achieve this, an innovative path planning method is used that combines 3D information obtained through Kinect-V1 and R-CNN for recognition of the work environment. The development of this application allows expanding the field of work and the autonomy of the manipulators, by granting them the ability to recognize the surrounding elements, and automatically establish changes on their original trajectory to avoid collisions.

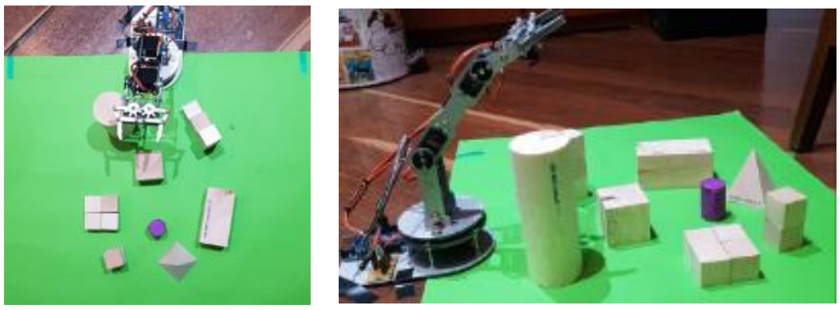
The present article is divided into 4 main sections, including this introduction. In the second section, the research method, the environment used, and the network architecture are explained, with the results obtained in the training of the R-CNN used for the algorithm. In the third section, the results of the application of the algorithm against different working conditions are presented and analysis on them is done, to finally raise a series of conclusions, in the fourth section.

2. RESEARCH METHOD

The algorithm developed is responsible for detecting the location of the desired object, the current position of the robot, and the presence of any obstacle type in the work environment, through a Kinect-V1 and artificial intelligence techniques such as R-CNN.

In the execution of the program, a Kinect-V1 [13] was used to capture the RGB-D information and the software Matlab [14] for the execution of the algorithm. For the manipulation of surrounding elements, an anthropomorphic robot is used, controlled by an Arduino programming card [15]. The objects have forms like parallelepipeds, cylinders, and pyramids, and a purple cylinder for the target object.

An example of the work environment where all the possible types of obstacles, the purple cylinder (desired object) and the robot, are shown in Figure 1. Figure 1a is the top view of the environment, and Figure 1b, the side view.

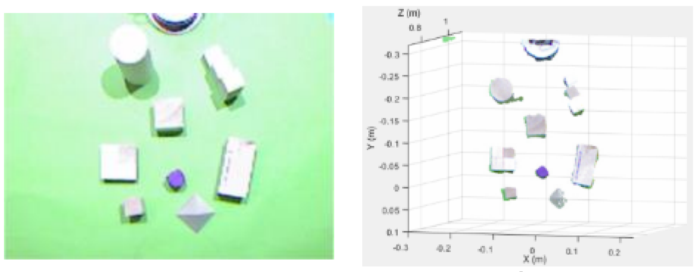


a. b.

Figure 1. Example of the work environment where a) top view and b) side view. Source: self-made

Figure 1 shows a green surface used as a working environment (ground) and the Kinect-V1 was located at a distance of 98 cm from the ground, to ensure that all the elements were within the range of vision of the Kinect (between 80cm and 4m, according to [16]), and that it was possible to recognize each element independently. Given that, a greater or lesser distance, the depth detection presented failures and losses of information, and generated confusion between objects identification.

An example of the capture of the work environment by the Kinect is shown in Figure 2. Figure 2a is the RGB capture and Figure 2b, the depth image after the elimination of the floor.



a. b.

Figure 2. Work environment captured by the Kinect-V1 where a) is the RGB image and b) the depth image. Source: self-made

As it can be seen, the Kinect-V1 captures the upper section of the obstacles for the case of the parallelepipeds and the cylinder, because the rest of the body is not visible from the current position of the Kinect, while for the pyramid it is possible to capture its entire body, due to the inclination of its faces.

The height Z of each obstacle, taken respect to the Kinect, which coincides with the height of the elements located in Figure 1 and Figure 2.Where the schematic of the robot and its direct and inverse kinematics coincide with those of the robot worked in [17] and the coordinates of the workspace are shown in Figure 3, the center of the robot is the point [0 0 0].

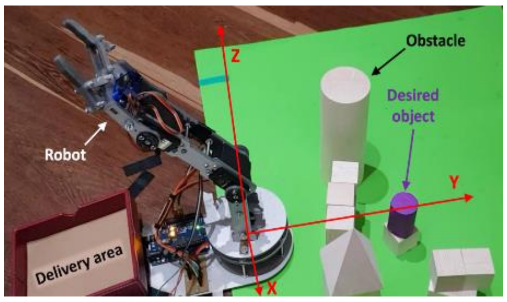


Figure 3. Workspace Coordinates. Source: self-made

Two R-CNN type networks were trained, one for the detection and classification of obstacles and the purple cylinder in the work environment (RCNNoc), and another for the detection of the robot in each frame of the application (RCNNr), using a database like the one shown in Figure 4. Here, the differences in size and shapes of each of the elements can be seen. The decision to use two R-CNN instead of one is justified by the tests performed in Figure XX of section III.

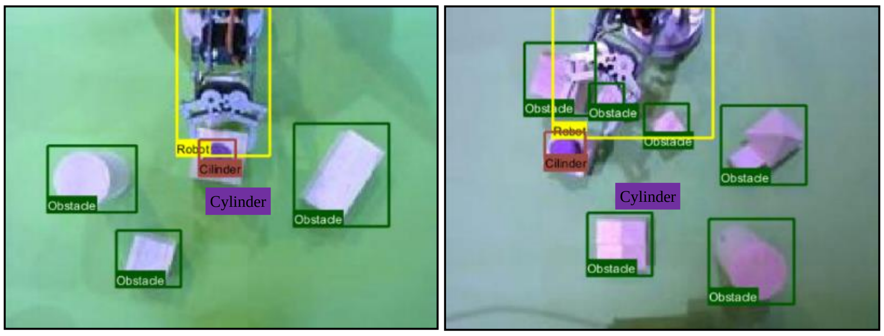


Figure 4. Database for RCNNoc and RCNNr. Source: self-made

The detection boxes for the RCNNoc and RCNNr networks were 30x30 and 45x45 pixels, for an input image of 240x340 and 160x214 pixels, respectively. This change of dimension in the detection boxes and the input image was due to the difference in size between the elements, because the robot manages to occupy a large part of the image in almost all its labels, while the obstacles and the cylinder can become as small as a 10x10 pixel bounding box for an input image of 160x214 pixels.

To define the size of the input image of the RCNNr, an average of the dimensions of the robot's detection boxes was taken for an image of 240x340 pixels and that factor that allowed to take the detection boxes to a size between 32x32 pixels and 64x64 pixels was sought in order to accelerate the detection process during the execution of the algorithm. 45x45 pixels images were defined, and, from this value, the input image was resized to reach 160x214 pixels.

Additionally, the use of small-sized images allows the network to recognize the elements in less time, but by specifying such small boxes, of 10 or 15 pixels on the side, the classification quality decreases, as well as having very large detection boxes, of 120x64 pixels for example (detection box for the robot in an image of 240x340 pixels). This is the reason is why two R-CNNs with input images of different dimensions were used.

Also, this change allowed to reach networks with a percentage of accuracy greater than 70%, while when trying to obtain a single network for all the elements, the percentages ranged between 25% and 30%, and in most cases, the purple cylinder was not detected.

The R-CNN has an image preprocessing stage that allows the regions of interest to be extracted from the input image, using an Algorithm Edge Boxes, to then individually enter them into a CNN in charge of doing the classification, as shown in Figure 5. It generates as a result the membership category, the coordinates of the upper left corner of the detection box, and its width and height dimensions.

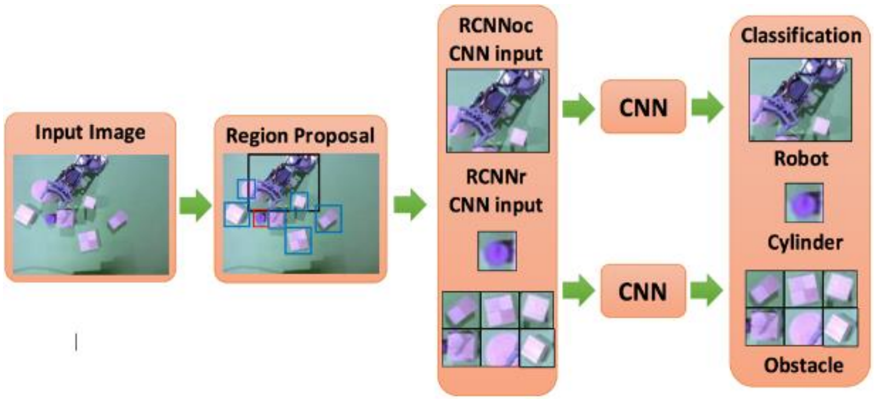


Figure 5. R-CNN flowchart. Source: self-made

The CNN architecture of the RCNNoc and RCNNr networks are presented in Tables 1 and 2, respectively. Small and square filters has been used to extract as much information from the image. For both networks, a Mini-Batch Size of 27 images, 2700 training images, 300 test images, and 200 training epochs were used, which were established by observing the accuracy of the network, which ceased to increase between 150 and 200 epochs.

Table 1. CNN architecture for the RCNNoc network

LAYERS	Filter Size HxW	Stride	Number of Filters
CONV+BATCH+RELU	6x6	1	16
CONV+BATCH+RELU	5x5	1	64
CONV+BATCH+RELU	4x4	1	128
MAXPOOL	**2x2**	**2**	--
CONV+BATCH+RELU	3x3	1	256
CONV+BATCH+RELU	3x3	1	512
FC1+RELU+DROP	--	--	--
FC2+RELU+DROP	--	--	--
FC3+SOFT	--	--	--

Table 2. CNN architecture for the RCNNr network

LAYERS	Filter Size HxW	Stride	Number of Filters
CONV+BATCH+RELU	4x4	1	16
CONV+BATCH+RELU	3x3	1	64
MAXPOOL	**2x2**	**2**	--
CONV+BATCH+RELU	3x3	1	128
CONV+BATCH+RELU	3x3	1	256
MAXPOOL	**2x2**	**2**	--
CONV+BATCH+RELU	3x3	1	512
CONV+BATCH+RELU	3x3	1	1024
FC1+RELU+DROP	--	--	--
FC2+SOFT	--	--	--

For the RCNNoc, 3 consecutive convolution layers were used to extract aspects such as edges and corners, allowing to differentiate the obstacles of the cylinder. For the RCNNr, a deeper network was chosen, because the appearance of the manipulator varies considerably between the different positions that it can adopt within the work environment and requires greater feature extraction. On the other hand, the Mini-Batch Size was established small for the training database, to avoid saturating the information network at each epoch, allowing to learn the position and shape of the objects.

The RCNNr achieved 100% accuracy in the recognition of the manipulator, where labels of the robot were marked in 169 images of the 300 test, as shown in the confusion matrix of Figure 6, where 1 is the robot and 2 the background.

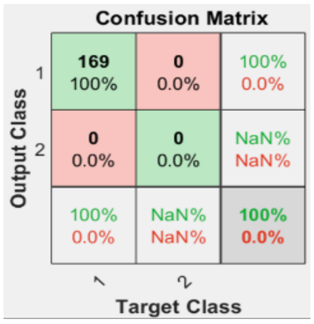


Figure 6. Confusion matrix of the RCNNr. Source: self-made

The classification carried out by the CNN of the network RCNNoc between the Obstacle and Cylinder categories, reached 100% accuracy, as no confusion was generated between the obstacles and the cylinder. However, the detection process done by the Edge Boxes Algorithm, presented a greater number of failures. The main reason for the "no detection" was due to differences in hue in the obstacles caused by the light, and the quantity, size, and orientation of them, leading to the network to fail to detect more than 8 obstacles per image, especially the pyramid, since the inclination of its faces increases the variation of hue according to the direction and intensity of the light.

The precision and levels of recall for both the RCNNoc and the RCNNr, where it is possible to check the low precision of the RCNNoc for the detection of the cylinder and obstacles (between 40% and 50% average precision) regarding the detection of the robot (92%). The recall results are the relationship between the level of overlap between the detection boxes generated by the network with respect to those defined in the database. The precision is the difference in location of the detection boxes of the network, with respect to the region of interest of the database. The detection boxes of the cylinder were more precise than the obstacle boxes, which reached only 42% accuracy, compared to 47% of the cylinder, which means that the pictures of the cylinder were closer to the labels of the test images than the boxes of the obstacles.

On the other hand, precision was obtained in the detection of the robot, with a value of 92% in the generation of the detection boxes. Additionally, the dimensions of the cylinder and the partial or total obstructions generated by the robot on it reduced the number of labels of the desired object in the test images, leading to only 294 cylinder labels in the 300 images, while there were 1764 obstacle labels in the test images.

Figure 7 shows the confusion matrix of the RCNNoc where 1 represents the obstacles and 2 is the cylinder. As can be seen, the “Obstacle” category presented a better detection than “Cylinder” (74.5% versus 54.1%), this because the presence of the robot in the work environment generated difficulties for the detection of the cylinder, especially when placed on it, and that there were a greater number of obstacles per image, in the training phase, than purple cylinders.

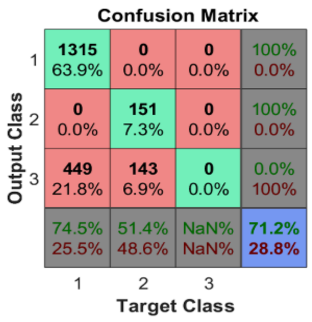
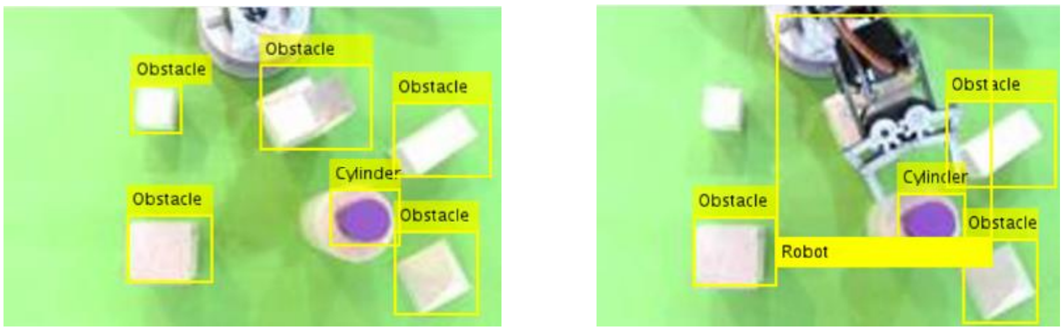


Figure 7. RCNNoc confusion matrix

It was observed that, when detecting and classifying objects only in the images where the robot is not present, an improvement in the detection percentage of the cylinder was achieved, up to 85% detection, as was experimentally verified by taking successive images of the environment and applying the RCNNoc. This allowed to reach 80.5% accuracy of the RCNNoc under this condition, which is why it was decided to use this network in the application, considering that only the cylinder would be detected at the beginning of the program, where 85% accuracy would prevail over the 51% in cylinder detection.

Furthermore, as shown in section III, it was possible to increase the percentage of accuracy of the RCNNoc by more than 10% with respect to the confusion matrix of Figure 9, varying the dimensions of the input image up to 2.5 times its original size, which led to reaching percentages of up to 84.8% accuracy of the network, for cases, both with and without the presence of the robot in the environment.

An example of detection and classification of the elements in the work environment is shown in Figure 8, where the detection is compared before (a) and after (b) the robot appears, using both networks at the same time. As can be seen, the presence of the robot causes the obstacle detection to be altered, which is why the RCNNoc was applied only in the initialization of the program and the RCNNr during execution, using it to update the robot's position in each iteration.



a. b.
Figure 8. Application of RCNNoc and RCNNr networks a) before and b) after the robot has appeared.
Source: self-made

3. RESULTS AND ANALYSIS

The initial tests of the algorithm are presented below, in which only the three-dimensional information captured by the Kinect and the detection of the robot made by the network were considered, ignoring theoretical information such as the dimensions of the robot and its ability to overcome obstacles that are too high.

Tables 3 and 4 show a record of the distance between the top of the robot and the surface of the obstacle, where the difference between them, according to the calculations made by the algorithm, was less than 45mm. Also, they are compared with the real distance between the robot and the obstacle at that moment, where the DisA column indicates the distance calculated by the program, DisR the real distance, DisError shows the absolute difference between DisA and DisR, K the value of the iterator k in which it was detected select<=45mm, and Collision indicates if in the execution of the entire algorithm, the robot collided with an obstacle.

All distances were taken in millimeters, and each of the tests was performed on the same work environment, for each of the bounding boxes, varying the height and position of the obstacles between one box and the other, to observe the variations in the execution of the program against different obstruction conditions.

Table 3. Distances between the robot and the obstacles, case 1

Test	DisA	DisR	DisError	K	Collision
1	21.2	35	13.8	39	No
2	28.6	35	6.4	39	No
3	32.2	50	17.8	37	No
4	26	35	9	40	No
5	27.5	45	17.5	39	No

The obstacle with a confidence of 90.86% being the element that the robot had to overcome to take the cylinder. Table 4 repeats the data collection in Table 3 for a second scenario, shown in Figure 9 where the obstacle with a confidence of 91.36% is the element that the robot must overcome to take the cylinder.

Table 4. Distances between the robot and the obstacles, case 2.

Test	DisA	DisR	DisError	K	Collision
1	38	40	2	36	Si
2	42.6	30	12.6	34	No
3	38.4	33	5.4	34	No
4	39.8	45	5.2	33	Si
5	41	40	1	35	No

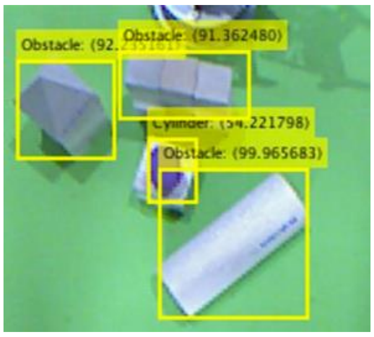


Figure 9. Working environment of Table 4. Source: self-made

In the test cases 1 and 4 of Table 4, a collision occurred, given that the inclination of the gripper reached such point (close to 90º with respect to the ground) that the motor in charge of opening and closing the gripper, located at the bottom of it, reached to touch the obstacle as the robot approached the desired object. Although the height of the manipulator had been detected relatively far from the obstacle (more than 30 mm), it was not possible to consider the presence of the motor.

The difference in the results of Collision of Tables 3 and 4 was mainly due to the height of the obstacle located between the robot and the desired object, where, in case 1, the height of said element was 9 cm, while which in case 2, it was 12 cm. Considering the range of vision of the Kinect, there is an instant of the application in which it is not possible to extract three-dimensional information from the robot (when it is over 18 cm high, relative to the horizontal), because it is outside the working range.

This limitation makes it difficult to evade very high obstacles since the Z points of the gripper begin to be recognized only when the robot is below 18 cm and, therefore, very close to the obstruction. Also, it is not possible to fully capture the cloud of points on the manipulator, unless its entire structure is below 18 cm, which affects the decision making of the algorithm and the quality of its operation, in addition to getting too close the gripper motor, to the obstacles.

As can be seen in Table 5, when the obstacle is of a lower height, DisR increases with respect to DisA, allowing to generate a safety margin of approximately 13 mm (DisError) that manages to consider the presence of the motor in the manipulator structure, thus generating a lower probability of collision with the obstacle and a better evasion of it.

Table 5. Average of tables 3 and 4

Case	DisA (mm)	DisR (mm)	DisError (mm)	K	P. Col.	Obstacle height
1	27.1	40	12.9	38.8	0%	9cm
2	39.96	37.6	5.24	34.4	40%	12cm

Additionally, the location of the desired object and the how the manipulator moves towards it must be taken into account, since, if the gripper generates a very steep inclination (close to 90º concerning to the horizontal), the joint motor 3 can get to be located on the tips of the gripper, and generate that the taking of three-dimensional information to be focused on the motor of the joint and not on the gripper, which can cause collisions, since it would not be possible to detect the true minimum height of the robot.

Faced with these difficulties, the obstacle avoidance algorithm have been executed with two theoretical considerations, where the first one stops the program by detecting, during the trajectory, the presence of an obstacle too high to be evaded by the robot, and the second supports the program with theoretical information about the robot's primary dimensions that help avoid collisions.

In the first case, a height limit was added to define the step of the robot over the obstacle, where the program stops if an obstacle appears above the set threshold. An example is shown in Figure 10, in which the robot must try to overcome an obstacle that is too high, and the algorithm, by detecting that it is not possible to pass over the object, stops the program and informs that it is not possible to evade the obstacle.

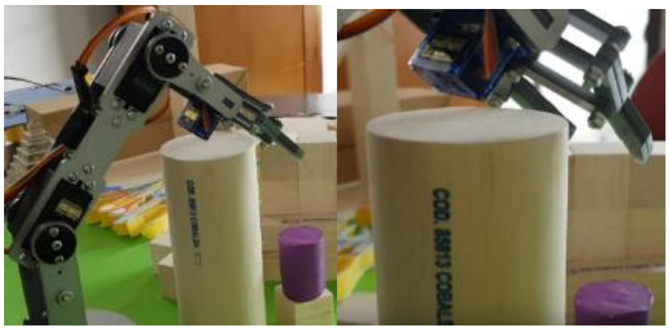


Figure 10. Example when the obstacle is too high. Source: self-made

In the second case, the absence of three-dimensional information during the displacement of the robot was solved, with the theoretical information of the dimensions of the robot, generating a comparison of height between the collected and theoretical three-dimensional information. An example of the distance between the robot and the obstacle is shown in Figure 11 when the conditional stops the program due to a high probability of collision, where it is possible to appreciate the proximity between the motor and the obstacle surface.

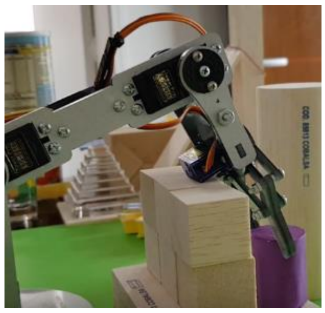


Figure 11. High theoretical probability of collision. Source: self-made

On the other hand, when comparing the behavior between the two networks trained for the application (RCNNoc and RCNNr), and a third trained in the first tests of the algorithm (RCNNeo) that detects and classifies both the obstacles and the desired object and the robot, it was possible to observe a similar behavior between both cases. However, it has differences such as: RCNNoc and RCNNr, as independent networks, allow better intersection detection than the RCNNeo, because the RCNNr reaches to cover the obstacles that the robot has to evade, intersecting with its detection box, while RCNNeo let the robot get too close to the objects without generating intersections between the robot's bounding box and the obstacles. It constantly causes that hayO equals 0, when the robot really encounters a high probability of collision.

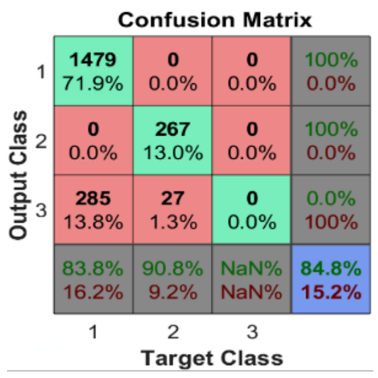
The CNN of the RCNNeo network has an architecture like the one shown in Table 6 (very similar to that of RCNNr), for an input image of 160x214 and detection boxes of 20x20 pixels.

Table 6. Architecture of CNN for the RCNNeo network

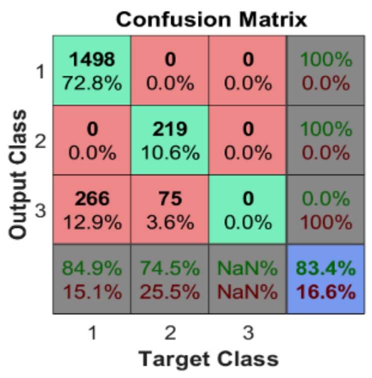
LAYERS	Filter Size HxW	Stride	Number of Filters
CONV+BATCH+RELU	3x3	1	16
CONV+BATCH+RELU	3x3	1	64
MAXPOOL	**2x2**	**2**	--
CONV+BATCH+RELU	3x3	1	128
CONV+BATCH+RELU	3x3	1	256
CONV+BATCH+RELU	3x3	1	512
FC1+RELU+DROP	--	--	--
FC2+SOFT	--	--	--

As can be seen, the dimensions of the detection box with respect to the minimum dimension of the obstacles and the cylinder in the image (10x10 pixels) make the detection boxes on these elements larger than them and covering more space of the necessary. In contrast, for the network RCNNoc, the detection boxes are fairer, which allows better recognition of the obstacles and the cylinder, and more precise extraction of its three-dimensional information during the execution of the application.

On the other hand, when testing the RCNNoc network with 2.5 and 1.5 times larger input images than training images, an improvement of more than 10% in the recognition of the elements was observed, as shown in the confusion matrix of the Figure 12a and Figure 12b. However, the network detection process, for the first case, takes up to 1.5 seconds evaluating a single image, while in the second case (1.5 times larger images) an average detection time of 762 ms was obtained, although with a 1.4% accuracy lower than that of Figure12a.



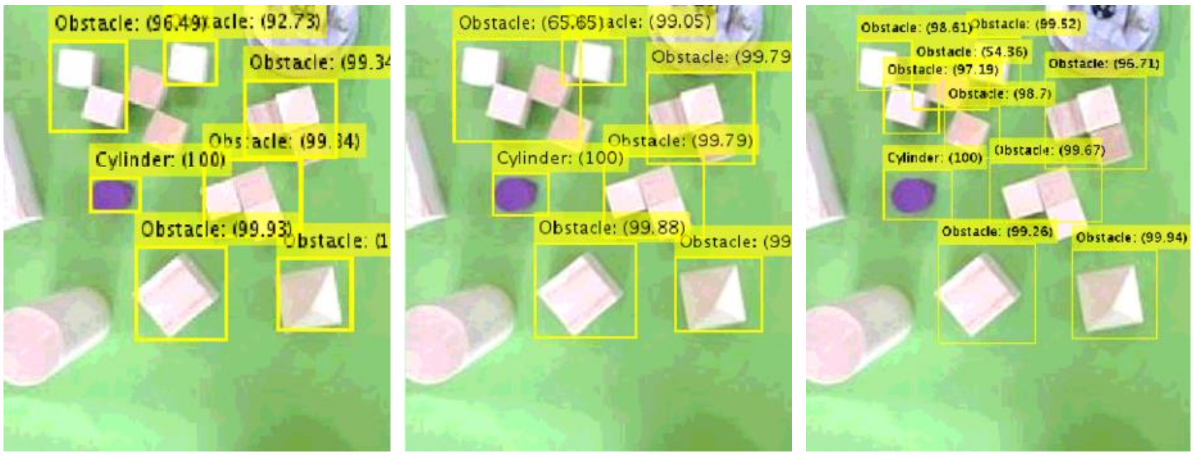
a.



b.

Figure 12. Confusion matrix for the RCNNoc network with a.) 2.5 larger input images. B.) 1.5 larger input images. Source: self-made

Although the difference in accuracy between the cases mentioned is small, it was preferred to use 2.5 times larger images for the detection of obstacles and the cylinder, since the confusion matrix of Figure 12a shows a 90.8% accuracy for recognition of cylinders, while the other only reach 74.5%, with differences of 1% in the recognition of obstacles. Moreover, when using larger images, it was observed that the RCNNoc network makes a more detailed capture of the obstacles, as shown in Figure 13, where detections for images 1, 1.5 and 2.5 times larger were compared, making it possible to detect independently each square obstacle in larger images, and not a whole bounding box for many of them (Figure 13c).



a. b. c.

Figure 13. Detection of elements for images a) 1 time, b) 1.5 times and c) 2.5 times bigger. Source: self-made

4. CONCLUSION

The use of the R-CNN allowed the detection of intersections between the robot and the elements of the environment, in addition to providing constant information about the current location of the manipulator and the three-dimensional information of the work area (with the help of the Kinect), to determine the evasion points of obstacles, without requiring previous information from the environment or the specific positions of each part of the robot. The combination of these two tools allowed the generation of an obstacle evasion algorithm, independent of the kinematics of the manipulator, and with the potential to operate under other working conditions, requiring only the training of the R-CNN for the specific application.

The use of independent R-CNNs helped to reduce the loss of information on the objects present in the environment since the RCNNoc detected the presence of obstacles and the purple cylinder to store their information in variables throughout the execution of the algorithm. It avoids the need to re-detect them in the following iterations, with the presence of occlusions such as the robot. Also, the previous detection carried out by the RCNNoc allowed generating detection boxes whose dimensions precisely contained the elements, ensuring an extraction of the cloud of points of each one of them, without interference from other objects.

On the other hand, the range of vision of the Kinect limited the physical workspace of the application, leading the algorithm to the detection, only, of objects less than 18 cm high, generating faults with obstacles very close to that height. However, the algorithm works properly away from the limits of vision of the Kinect, and within the working range of the robot, determined by the dimensions of its links and by its kinematics.

ACKNOWLEDGEMENTS

The authors are grateful to the Militar Nueva Granada University, which through its Vice chancellor for research finances the present project with code IMP-ING-2290 and titled " Prototipo de robot asistencial para labores de cirugía", from which the present work is derived.

REFERENCES

[1] V. Alekh, E. S. Rahul, R. R. Bhavani, Comparative Study of Potential Field and Sampling Algorithms for Manipulator Obstacle Avoidance, International Journal of Control Theory and Applications, vol. 9, p. 71-78, (2016)

[2] J. E. H. Benavides, D. E. E. Corredor, R. J. Moreno, R. D. Hernández, Flood Fill Algorithm Dividing Matrices for Robotic Path Planning, International Journal of Applied Engineering Research, vol. 13, no. 11, p. 8862-8870, (2018).

[3] Pachon-Suescun Cesar G., Enciso-Aragon Carlos J., Jimenez-Moreno Robinson, Robotic navigation algorithm with machine vision, International Journal of Electrical and Computer Engineering (IJECE), Vol 10, No 2: April 2020, p. 1308-1316.

[4] B. Li, M. Tian, S. Zheng, Y. Ling, Design and Research of Picking Manipulator Obstacle Avoidance System Based on IOT, International Journal of Online Engineering (iJOE), vol. 14, no. 3, p. 152-163, (2018).

[5] D. Han, H. Nie, J. Chen, M. Chen, Dynamic obstacle avoidance for manipulators using distance calculation and discrete detection, Robotics and Computer-Integrated Manufacturing, vol. 49, p. 98-104, (2018).

[6] Qiuxiang Chang, Zhenkai Xiong, Vision-aware target recognition toward autonomous robot by Kinect sensors, Signal Processing: Image Communication, Volume 84, 2020, 115810, ISSN 0923-5965.
[7] D. Chen, Y. Zhang, Minimum jerk norm scheme applied to obstacle avoidance of redundant robot arm with jerk bounded and feedback control, IET Control Theory & Applications, vol. 10, no. 15, p. 1896-1903, (2016).
[8] M. Benzaoui, H. Chekireb, M. Tadjine, A. Boulkroune, Trajectory tracking with obstacle avoidance of redundant manipulator based on fuzzy inference systems, Neurocomputing, vol. 196, p. 23-30, (2016).
[9] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, In Advances in neural information processing systems, pp. 1097-1105, (2012).
[10] C. L. Zitnick, P. Dollár, Edge boxes: Locating object proposals from edges, European Conference on Computer Vision, Springer, Cham, p. 391-405. https://doi.org/10.1007/978-3-319-10602-1_26 (2014)
[11] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587, (2014).
[12] Jiménez-Moreno Robinson, Pinzón-Arenas Javier Orlando, Pachón-Suescún César Giovany, Assistant robot through deep learning, International Journal of Electrical and Computer Engineering (IJECE), Vol 10, No 1: February 2020, p. 1053-1062.
[13] Z. Zhang, Microsoft kinect sensor and its effect, IEEE multimedia, vol. 19, no. 2, p. 4-10, (2012).
[14] Matlab, U. S. G. The mathworks. Inc., Natick, MA, 1992, (1760).
[15] Y. A. Badamasi, The working principle of an Arduino. En Electronics, computer and computation (icecco), 2014 11th international conference on, p. 1-4, (2014)
[16] E. Duque, Diferencias entre Kinect V1 y Kinect V2, Toward a brand-NUI World, published (5 February 2015), seen (5 September 2018), [Online], https://edwinnui.wordpress.com/2015/02/05/diferencias-entre-kinect-v1-y-kinect-v2-2/
[17] P. U. Murillo, R. J. Moreno, Individual Robotic Arms Manipulator Control Employing Electromyographic Signals Acquired by Myo Armbands, International Journal of Applied Engineering Research, vol. 11, no. 23, p. 11241-11249, (2016).
[18] W. Magnus, F. Oberhettinger, Formulas and theorems for the special functions of mathematical physics. Chelsea Pub. Co, (1949).
[19] Ganesh Roy, Aritra Bhuiya, Aditya Mukherjee, Subhasis Bhaumik, Kinect Camera Based Gait Data Recording and Analysis for Assistive Robotics-An Alternative to Goniometer Based Measurement Technique, Procedia Computer Science, Volume 133, 2018, Pages 763-771, ISSN 1877-0509.
[20] Dasom Seo, Euncheol Kang, Yu-mi Kim, Sun-Young Kim, Il-Seok Oh, Min-Gul Kim, SVM-based waist circumference estimation using Kinect, Computer Methods and Programs in Biomedicine, Volume 191, 2020, 105418, ISSN 0169-2607.