Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 9, No. 1, March 2021, pp. 70~83
ISSN: 2089-3272, DOI: 10.11591/ijeei.v9i1.1846

Classification and Grip of Occluded Objects

Paula Catalina Useche Murillo[1], Robinson Jiménez-Moreno[2]
[1,2]Militar Nueva Granada University, Av cra 11 No 101-80, Bogotá, Colombia

Article Info

Article history:

Received Jan 13, 2020
Revised Mar 16, 2021
Accepted Mar 19, 2021

Keyword:

CNN
Fast R-CNN
DAG-CNN
Occluded objects
Robot Manipulator
tools recognition

ABSTRACT

The present paper exposes a system for detection, classification, and grip of occluded objects by machine vision, artificial intelligence, and an anthropomorphic robot, to generate a solution for the subjection of elements that present occlusions. The deep learning algorithm used is based on Convolutional Neural Networks (CNN), specifically Fast R-CNN (Fast Region-Based CNN) and DAG-CNN (Directed Acyclic Graph CNN) for pattern recognition, the three-dimensional information of the environment was collected through Kinect V1, and tests simulations by the tool VRML. A sequence of detection, classification, and grip was programmed to determine which elements present occlusions and which type of tool generates the occlusion. According to the user's requirements, the desired elements are delivered (occluded or not), and the unwanted elements are removed. It was possible to develop a program with 88.89% accuracy in gripping and delivering occluded objects using networks Fast R-CNN and DAG-CNN with achieving of 70.9% and 96.2% accuracy respectively, detecting elements without occlusions for the first net and classifying the objects into five tools (Scalpel, Scissor, Screwdriver, Spanner, and Pliers), with the second net. The grip of occluded objects requires accurate detection of the element located at the top of the pile of objects to remove it without affecting the rest of the environment. Additionally, the detection process requires that a part of the occluded tool be visible to determine the existence of occlusions in the stack.

Corresponding Author:

Paula Catalina Useche Murillo,
Militar Nueva Granada University.
Av cra 11 No 101-80, Bogotá, Colombia.
Email: u3900235@unimilitar.edu.co

1. INTRODUCTION

Object gripping systems using robotic manipulators have become essential for industrial automation processes [1] where different working conditions have been addressed in order to adapt industrial processes to unexpected environmental situations, such as the presence of people or objects in its trajectory [2]. However, occlusion cases have not been considered largely, where a manipulative robot should not avoid an obstacle but remove an element that prevents the direct grip of the desired object.

In some works, mobile robots have been programmed for the grip of partially occluded objects, where the robot moves to change its viewing angle and thus be able to observe and detect a grip point on the object of interest, which has a certain degree of occlusion, taking advantage of the use of RGB-D information for environmental recognition [3]. In other works, such as those of [4] and [5], different grip systems of partially occluded objects were developed using mobile robots, where the first one added a manipulator arm, so that it is possible to establish the grip with both the mobile displacement and the arm pose, which allows working in environments with little free space, while the second makes a prediction of the quality of future views of the mobile, based on the class and posture of the occlusions, and its three-dimensional modeling.

In other works, such as the one reported in [6], a manipulator was programmed to grip objects in the presence of occlusions, where ellipses were used to estimate the grip pose of the element that generates occlusion, and thereby allow the robot grab the object with a stable posture and remove it before continuing

with the other elements. In the case of [7] and [8] different techniques were used for the recognition of partially occluded objects, where [7] used Fourier transforms in order to extract relevant characteristics of the element that allow its recognition and tracking, while [8] relied on color information both RGB, HSV and YCbCr, for occlusion recognition.

On the other hand, [9] and [10] focused on the recognition and classification of occluded objects using artificial intelligence methods, without solving grip issues. In both, the quality of recognition of partially occluded objects (with up to 50% occlusion) was compared between techniques such as CNN, SVM (Support Vector Machines) and DBN (Deep Belief Networks), where DBN achieved better results in the classification of images with occlusions and without them, reaching 55% accuracy, while CNN achieved 51.5% and SVM 51.3%.

CNNs are neural networks that have convolutional filters in their structure, responsible for extracting and learning the most relevant characteristics of an image in order to generate a classification within a group of previously trained categories [11]. The network architecture is defined by the user, who can combine layers of different types (presented and explained in [12]) and vary their parameters in order to couple the network to the requirements of the application and obtain better accuracy percentages.

CNN has been used for various applications for recognition and classification of images and patterns, such as the estimation of grip positions on objects of various geometries, as indicated in [13], where RGB-D and transfer learning was used for the detection of the best grip posture, and in [14] a CNN was used to discriminate tools in assistive robotics, in order to classify 4 different types of tools, achieving 96% accuracy in the network, and 94% in the case of [13].

Additionally, DAG-CNN type networks have been developed that contain two or more lines of parallel convolutional layers, as explained in [15], thus allowing an increase in the characteristics extracted from the image, thanks to each linear succession of layers get different kind of information. There are also CNN networks that include in their structure an additional stage of detection, which extracts the elements of interest from an image in order to enter them into a CNN, and generate their classification. These networks are the R-CNN (Region-Based CNN), the Fast R-CNN and the Faster R-CNN.

The R-CNN uses a region proposal algorithm based on detection boxes [16] to find the elements, the Fast R-CNN uses a Region Proposal Network that trains together with the CNN and allows faster detection test time than the R-CNN [17], and the Faster R-CNN has a box regression layer, and uses Pooling and fully connected layers to generate the estimation of the detection boxes [18]. Some examples of the application of R-CNN are shown in [19] and [20], where the first one detects and classifies up to 20 categories of animals and objects, with a 74.8% accuracy, and the second uses the R-CNN to the detection and classification of surgical instrumentation during surgery processes, classifying up to 7 categories with 81.8% accuracy.

An example of application for Faster R-CNN and Fast R-CNN, respectively, are presented in [21] and [22], where the first one detects and classifies a group of 3 homemade tools in a virtual environment, in order to grab and order them in different cubicles, achieving a 99% accuracy network, while the second classifies the same 20 categories as [19] but with an accuracy percentage of 71.4%.

For the following article, a system for detecting, classifying and grabbing occluded objects was designed using a manipulator robot, where a static viewing angle was established that makes a global capture of the environment and from it, detects each set of elements using a Fast R-CNN, and classifies them into 5 groups of homemade tools using a DAG-CNN. This work makes an important contribution in the process of grasping occluded objects, since an algorithm is proposed to search, grab and deliver a desired object, independent of the occlusions that may occur, thus complementing the state of art presented until the moment, where the works are limited to the classification of the occluded object, or to the estimation of the grip position when there are occlusions, but not to the complete process of grip and delivery. No research was found to use convolutional networks for object detection with occlusions in robotic applications in-state of the art. However [26] exposes a case with human tracking considering complete occlusion, but comparing of CNN and Histogram of Oriented Gradients (HOG) with Support Vector Machine (SVM) called HOG-SVM, not in the same way of this work.

The present article is divided into 4 main sections: in the first section, a brief introduction is made to occlusion detection systems in mobile robots, methods of classification of occluded objects and estimates of grip poses in manipulators for grasping of occluded objects. In the second section, the operation of the algorithm and each of the steps that compose it are explained. In the third section, the results of the application of the algorithm against different working conditions are shown and an analysis of them is carried out, to finally establish a series of conclusions, in the fourth section.

2. RESEARCH METHOD

The basic operation of the detection, classification and gripping program of occluded objects is detailed below, which uses artificial intelligence techniques such as Convolutional Neural Networks (CNN), and three-dimensional information captured by Kinect V1 [23].

Section II was divided into 7 subsections, where the first briefly explains the behavior of the algorithm and its operating flow, the second raises the working conditions necessary for the correct execution of the application, the third describes the variable initialization process prior to the beginning of the application, the fourth explains the initial capture of the environment and the first detection of tools with the Fast R-CNN where it is defined if there are elements or not, that allow starting the program, the fifth details the classification process of tools using DAG-CNN, the sixth indicates the method of grip and delivery of tools according to their denomination (desired object or not), while the seventh describes the process of searching for new tools in order to determine if the process has ended or if there are elements to evaluate.

2.1. Basic algorithm operation

The developed algorithm is in charge of capturing an RGB-D image of the work environment in order to detect and classify the tools present in it, and according to their locations, it generates a grip algorithm that allows removing unwanted elements, relocating them, and grab those expected by the user for delivery.

The sequence of operation of the algorithm is presented in the flowchart of Figure 1, where each step of the program was marked with numbers from 0 to 4.

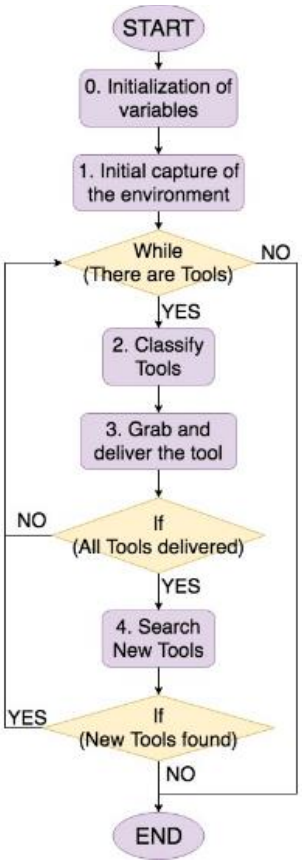


Figure 1. Flowchart of the algorithm. Source: self-made

In step 0, the variables used in the program are initialized, and the Kinect and the robot are linked. In step 1, the initial capture of the work environment is performed, where the tools are first detected with the Fast R-CNN, to determine if there are elements present or not. In step 2, all the tools detected are classified using the DAG-CNN. In step 3, the grip and delivery of tools is performed, placing them according to their classification and user requirements. In step 4, a new tool search is performed, once the algorithm has finished organizing all the elements detected in step 1, to verify that all objects in the environment have been removed.

The first contact the user has with the program is with the graphic interface shown in Figure 2, where one of the 5 trained tools (Scalpel, Scissor, Screwdriver, Pliers, Spanner) must be selected, for its grip and delivery, and then press START to start the application.

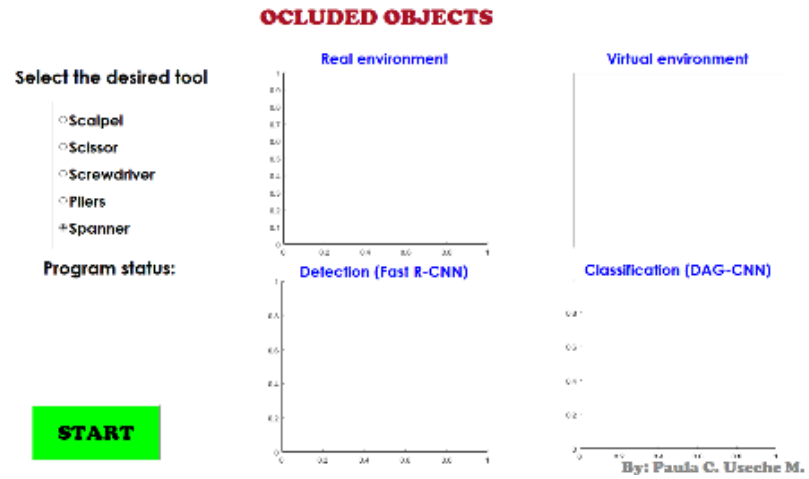


Figure 2. Graphical User Interface (GUI). Source: self-made

The graphical interface tells the user the status of the program (Program Status) and shows the image of the work environment (Real Environment), the simulation of the gripping process (Virtual Environment), the detection of the tools (Detection) and the classification of each of them (Classification). An example of the graphical interface during program execution is shown in Figure 3.

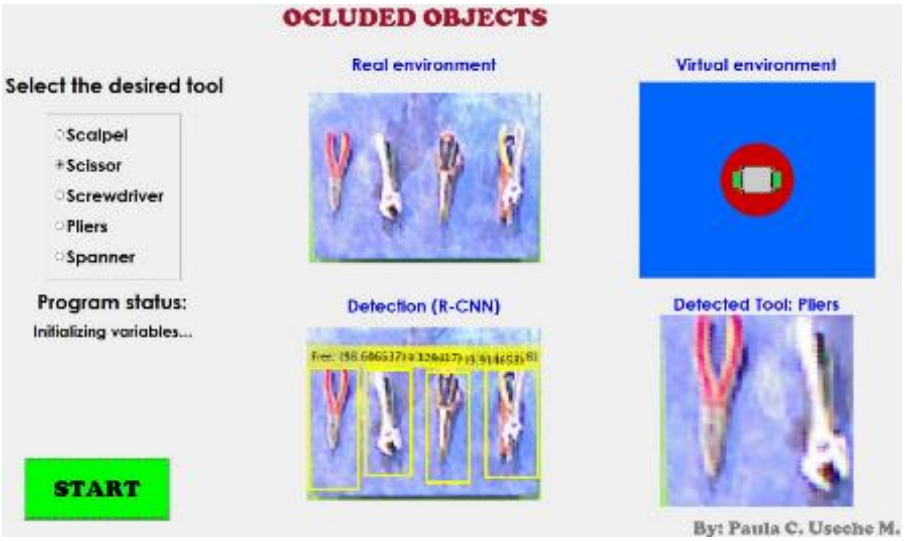


Figure 3. Running application. Source: self-made

Once the user starts the application, the algorithm is executed automatically, making decisions about what to do with the occlusions, and stopping only when all the tools have been removed from the work area, thus ensuring the grip and delivery of all items classified as the desired tool. Next, the working conditions required for the execution of the algorithm are presented, and each of the steps shown in the flowchart of Figure. 1 are explained.

2.2. Methods and materials

In previous research exposes in [28], the robotic arm found some obstacles in his path, where the occlusions let evidenced a problem in the object recognition. The same environment is employed, where Kinect V1 was used to capture the working environment and ubicated at a distance of 91cm from the ground. An academic anthropomorphic robot was used for the tools' grip, a flat work surface 6cm high was used as a working area (table), and a total of 5 homemade tools were selected, where aspects such as the color and shape of each of them were varied. Finally, five different types of elements of each tool were obtained, as shown in Figure 4.

In Figure 5 the working environment is shown, where the tools to be classified are located on the table, to avoid forcing the motors too much when taking the robot to a very ground level location. The working range of the robot covers the entire table, and on the opposite side of it, the gripped tools are deposited.

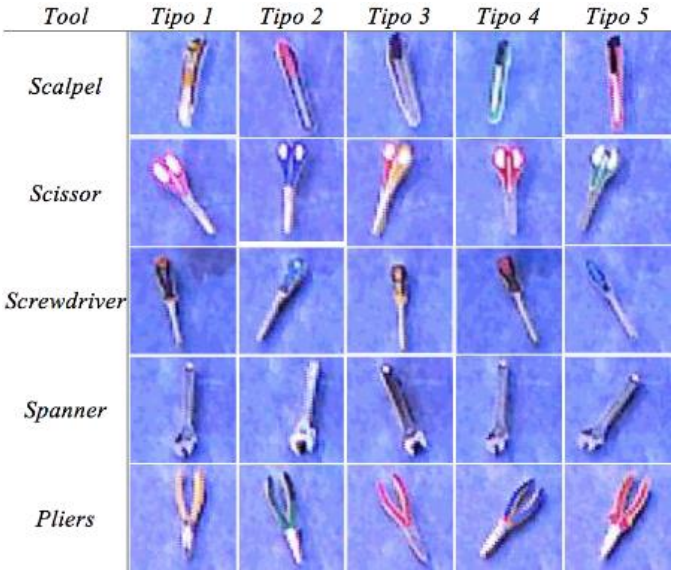


Figure 4. Work tools. Source: self-made

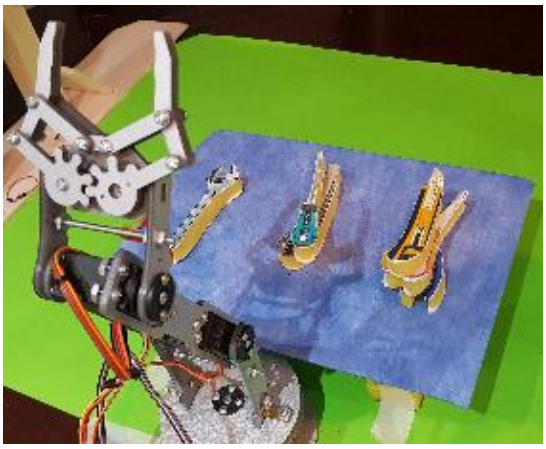


Figure 5. Work environment. Source: self-made

Two networks were trained for the development of the application, a Fast R-CNN for the detection of all the tools of the environment, and a DAG-CNN for its classification. The decision to use two networks instead of one, was because the DAG-CNN has shown a better behavior in the classification of categories very similar to each other, with main characteristics of different dimensions and shapes, thanks to the fact that it has two or more convolution branches that allow you to extract a greater amount of information than a CNN [15], which is used in the Fast R-CNN.

On the other hand, the Fast R-CNN oversees the process of detection and classification of the tools in Free and Occlusion categories, which allow to know when there are stacked elements (Occlusion) and when not (Free). In [29], a similar case based on distance information and faster R-CNN is showed, illustrating how this network improves object detection using a region of interest ROI. That condition allows knowing object location in complement to the fine-tune discrimination of the DAG-CNN.

According to the results generated by the Fast R-CNN, the program classifies with the DAG-CNN the tools defined as Free and starts the process of gripping and delivery of elements, then classifies those of the Occlusion category, and repeats the process. At the end, apply the Fast R-CNN again and repeat the whole process if another tool is found, otherwise, the program ends.

2.3. Step 0: Initialization of Variables

The first step of the algorithm is to initialize the variables that will be used throughout the program, which are found in Table I, where its function, name and initialization value are mentioned.In addition to the variables mentioned in Table I, the graphical interface, the Kinect and the robot are initialized, and the networks are loaded.

Table 1. Variable Initialization

Variable	Function	Value
dimDAG	Dimensions of the input image for the DAG-CNN (pixels)	[70 70]
MaxZ	Maximum height to determine the presence of tools. Distance taken from the Kinect (mm)	897
PO	Final tool delivery positions (meters)	Dimensions: PO (10,3,2)
herrOD	User desired tool (char)	Input: Scalpel, Scissor, Screwdriver, Spanner, Pliers
xyzL	Kinect working range limits (meters) [X; Y; Z]	[-0.3 0.23; -0.32 0.1; 0.7 1]
dimI	Dimensions of the work area image (pixels)	[128 174]
ihd	Counter of the number of desired objects grasped	1
iho	Counter of the number of unwanted objects removed	1
tipoHerr	Type of tool	- Desired object (1) -Unwanted object (2)

The variable PO stores two matrices, each with 10 rows and 3 columns, where the first matrix saves the final positions for the desired objects, and the second saves the final positions for the unwanted elements, allowing a maximum of 10 objects per matrix, since the physical dimensions of the work environment do not allow more than 10 elements to be organized in the delivery area.

2.4. Step 1: Initial Capture of the Environment

In this section, the first capture of the work environment is made, the tools are detected and classified as Free or Occlusion, and the initial coordinates X, Y, Z of the elements are captured. The first step is to capture the RGB-D information of the environment with the Kinect, and then enter the RGB image to the Fast R-CNN, in order to detect the presence of any type of tool.

The Fast R-CNN detects and classifies objects in the Free and Occlusion categories, where the first demarcates those objects that do not present occlusions, while the second demarcates those where there are one or more tools covering some element. This type of network has a structure like the one shown in Fig. 6, where a stage of extraction of ROIs, the anchor's of the ROI's, layers of Pooling, Fully connected, and a box regressor [25] are added to the CNN for the detection process, generating outputs like the membership category and its confidence (softmax), the coordinates of the upper left corner of the detection box, and its width and height dimensions (bbox regressor).

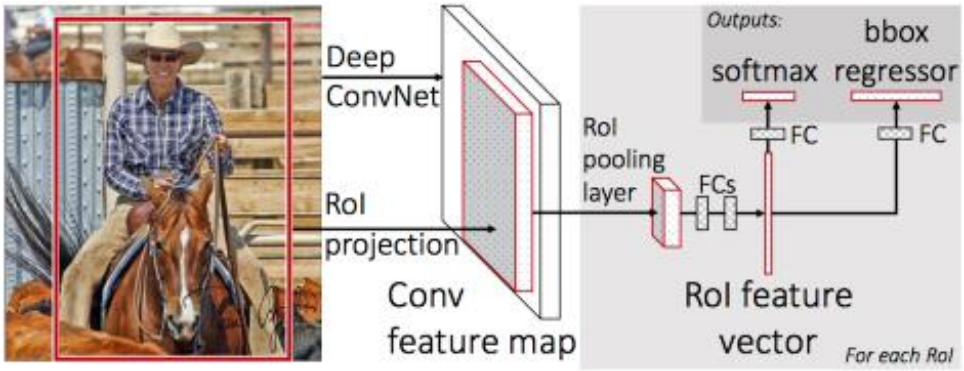


Figure 6. Fast R-CNN flowchart. Source: Girshick, "Fast r-cnn" [25]

The architecture for the CNN of the network is presented in Table 2, where rectangular filters were used to extract equivalent information from both dimensions of the image, avoiding deformations in the feature extraction process.

As shown in Table 2 and as indicated in [19], CNN has convolutional layers (CONV), Rectified Linear Units (RELU), MaxPool (POOL), Average Pool, Batch Normalization (BATCH), Fully Connected (FC), DropOut (DROP), Softmax (SOFT), among others.

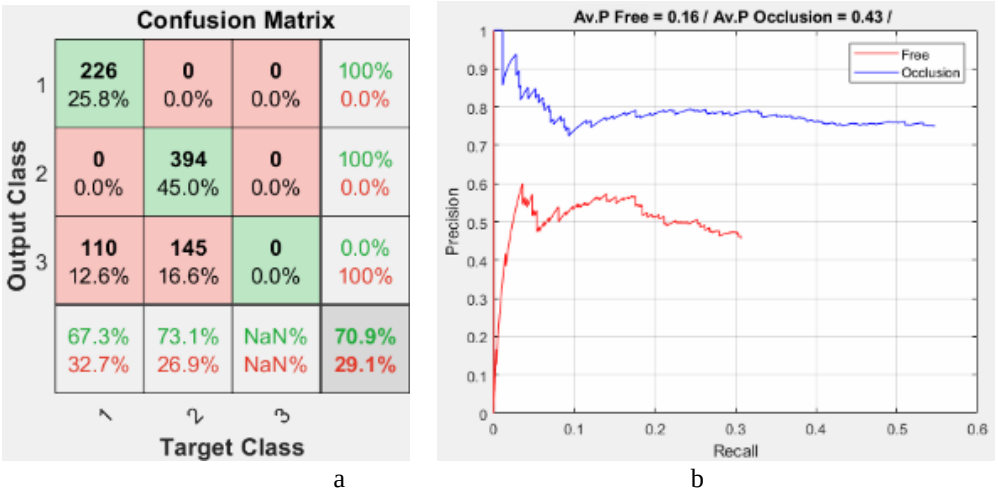
The network was trained for 300 epochs, with 2800 training and 350 test images, a 28 MiniBatchSize, an input image of 128x174 pixels, and detection frames of 55x28 pixels. The images in the database were taken as shown in Figure 7, where the elements that are alone were demarcated as Free and the others as Occlusion. A Data Augmentation algorithm [27] was used to increase the database, varying aspects such as color, light intensity and adding noise to the images.

Table 2. Architecture of the Fast R-CNN

LAYERS BRANCH 1	Filter Size HxW	Stride	Number of Filters
CONV+BATCH+RELU	6x4	1	16
CONV+BATCH+RELU	5x3	1	128
MAXPOOL	**2x3**	**2**	--
CONV+BATCH+RELU	4x3	1	256
CONV+BATCH+RELU	4x3	1	512
FC1+RELU+DROP	--	--	--
FC2+RELU+DROP	--	--	--

Figure 7. Fast R-CNN Database. Source: self-made

The MiniBatchSize was chosen small to make the network learn few information in each epoch and assimilate it without saturating it, in order to improve its learning, since by increasing this parameter, in each epoch a lot of information is presented, and that increases the cost of learning. The confusion matrix of the Fast R-CNN and the Recall Vs Precision graph are shown in Fig. 8, where the network reached 70.9% accuracy, with a precision in the detection boxes of 16% and 43% for Free and Occlusion, respectively. 1 is for Free, 2 for Occlusion and 3 for Background.



a b

Figure 8. Fast R-CNN a) Confusion matrix and b) Recall Vs Precision. Source: self-made

Although the network only reached 70% accuracy, it was used in the application since it was observed that by repeating the detection process in Step 4, the network captures the tools that were not capture in the first take, which allows the process to be fully developed, without inconvenience. An example of the detection

and classification of the Fast R-CNN is shown in Figure 9 for a case of 3 tool stacks where the boxes manage to completely cover the elements, correctly classifying the tools with and without occlusions.

Figure 9. Example of application of the Fast R-CNN. Source: self-made

According to the detection results, the coordinates [x y w h] are stored in the variable Objetos, where [x y] are the coordinates of the upper left corner of the detection box generated by the Fast R-CNN and [w h] the width and height of the box, respectively. Each row of Objetos contains the location of each element, and they are stored according to their classification, leaving in the first rows those tools classified as Free. To differentiate the Free and Occlusion categories, a number 1 for Free and a 2 for Occlusion were marked in the first column of Objetos.

Subsequently, the average height in Z was extracted from each detection box using the depth information captured by the Kinect. The depth data corresponding to the table height were eliminated, and the average of the remaining data was obtained. Because the height changes detected by the Kinect between one tool and the other are very subtle, predefined height ranges were established, where those values that are within a certain range, are assigned a specific Z height that will be used in the application to move the robot to the grip point. This reduces the influence generated by the 3D information of other tools present in the detection box on the actual height of the tool to be grasped, during the calculation of the average.

If no tool is found, the program displays the message "The desired object was not found" and ends. Otherwise, the algorithm proceeds to calculate the specific position of the tool found, using the activations of the DAG-CNN, where the tool is extracted from the background, by marking with white the most relevant information of the image (the object) and leaving the less relevant information (the background) in black as shown in Figure 10. In this way, the centroid of the largest white sector of the box is obtained and it is determined as the final position of the tool, and the results are stored in xh, yh variables.

Figure 10. Original image (left) and activations of the DAG-CNN (right). Source: self-made

As a result of this subsection, the variable Objetos is obtained as shown in (1) and the final grip positions of each of the tools (Pf) as shown in (2), where z is the calculated average height for each Detection box, Clasif is the classification of boxes as Free (1) or Occlusion (2), and n the total number of detection boxes.

$$Objetos = \begin{bmatrix} Clasif_1 & x_1 & y_1 & w_1 & h_1 & z_1 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ Clasif_n & x_n & y_n & w_n & h_n & z_n \end{bmatrix} \quad (1)$$

$$Pf = \begin{bmatrix} x_{h1} & y_{h1} & z_1 \\ \dots & \dots & \dots \\ x_{hn} & y_{hn} & z_n \end{bmatrix} \quad (2)$$

2.5. Step 2: Classification of the Tools

In Step 2, the detected elements are classified in order to determine what type of tool is: Scalpel, Scissor, Screwdriver, Spanner, Scissor. Depending on the result and the object selected by the user, the program determines if it should be removed because it is an occlusion, or grasped for delivery, if it is the desired object.

The tool classification process is executed inside the While of Figure 1, and its purpose is to extract each tool from the detection boxes and classify them with the DAG-CNN. To do this, the coordinates [x y] of the detection box corresponding to the iteration i of the While (Objects (i,2:3) i from 1 to n) are taken and from there, a box of dimensions dimDAG is extracted to enter it to the network, which gives as a result the category of the tool evaluated.

Then, the classification of the DAG-CNN is compared with the tool desired by the user (herrOD), and if they are equal, 1 is assigned to tipoHerr, but assigned 2. Additionally, after generating the first classification, the firstTool variable is set to 1, that indicate that changes are going to be made in the work environment and, therefore, RGB-D information must be captured again at each iteration. Once firstTool is equal to 1, the RGB-D capture of the environment is repeated before classifying any detection box, in order to update the information of the work area, especially for occlusion cases, where the tool that was located at the top of the object stack, was removed in the previous iteration.

After generating the classification and determining if the tool corresponds to the object desired by the user or not, proceed to execute Step 3 of the diagram in Figure 1. As previously mentioned, the DAG-CNN has a parallel branch structure, where each one is composed of convolution layers, and all come together at a certain point to deliver a classification as a result. In Figure 11, the structure used for the DAG-CNN of the application is shown, where there is a division of 2 branches that receive the image to be classified as input, and are joined at the end in a Fully Connected (FC) layer.

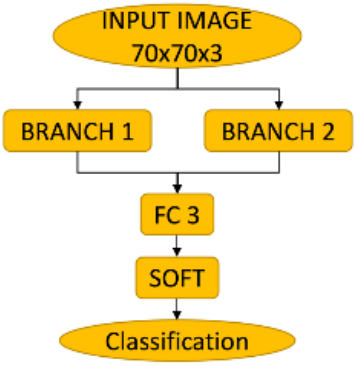


Figure 11. Structure of the DAG-CNN. Source: self-made

Table 3 shows the architecture of the DAG-CNN used to classify the 5 types of tools chosen for the program, where input images of 70x70 pixels were used, and square filters for the equivalent extraction of both horizontal information and vertical, where one branch uses filters larger than the other, to extract both large and small characteristics.

Table 3. Architecture of the DAG-CNN

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

The network was trained for 90 epochs, with a MiniBatchSize of 60; the database consists of 4500 images where 3000 images are used in the training and 1500 image in the test (300 per category), NVIDIA GeForce GTX 1060 6GB GPU (Graphics Processing Unit), using MATLAB® software [24] for training. The input image was set at 70x70 pixels because it is the maximum length that the tools reach when photographing them with the Kinect from their working height, and it was defined as a square image, to capture the tool even when it is very inclined, as shown in the first image of Figure 12.

Figure 12. Example of the training database of the DAG-CNN. Source: self-made

An example of the training database for the Pliers category is shown in Figure 12, where you can see the presence of other elements in the environment, in order to teach the network to classify work tools, even when elements of other categories are partially visible. The databases of the other tools are like that of Pliers.

The selection of the architecture for the CNN of both branches was based on the capacity of the DAG-CNN to extract information of various dimensions, where a branch with small filters and 4 convolutions (Branch 2) and another with slightly larger filters and 3 convolution layers (Branch 1) were defined, as observed in Table 3.

The confusion matrix of the DAG-CNN is shown in Figure 13, where 1 is Pliers, 2 Scalpel, 3 Scissor, 4 Screwdriver and 5 Spanner. Pliers was the category that best classified the network, and Screwdriver was the one that obtained the lowest accuracy, however, the lowest ranking percentage by category was 91%, and the highest was 100%, so it was decided to use this network for the application, expecting less than 10% error per category.

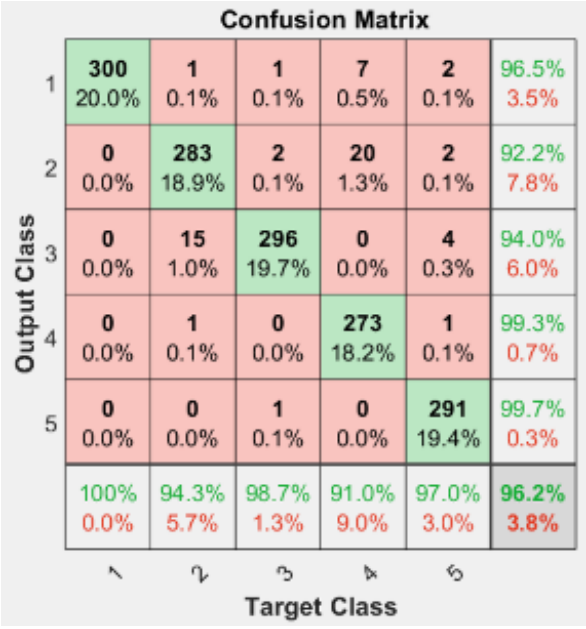


Figure 13. Confusion matrix for DAG-CNN. Source: self-made

An example of the classification of each trained tool is shown in Figure 14, under different working conditions, where it is possible to detect the presence of other elements inside the detection box. In Figure 14

the most difficult tool to classify was the Scalpel, which was confused with Pliers when one or more tools were presented in the detection box, while the others were correctly classified despite the number of elements present in the image.

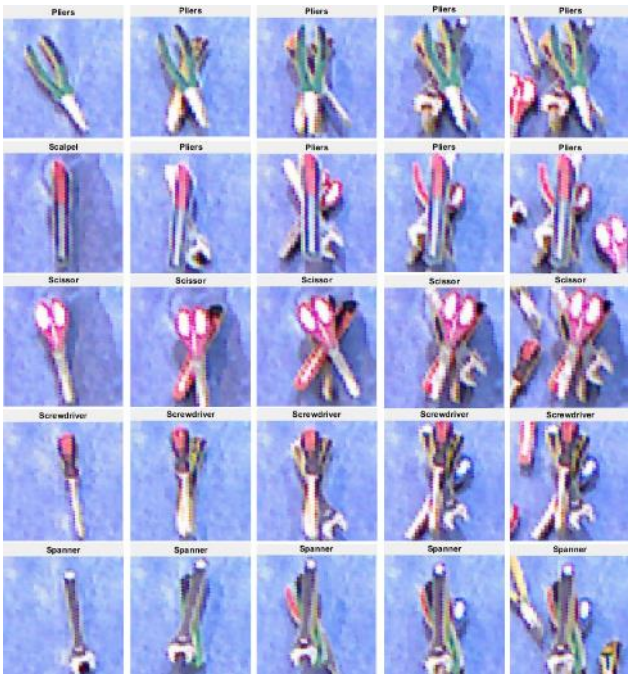


Figure 14. Examples of classification with the DAG-CNN in the physical work environment. Source: self-made

2.6. Step 3: Grip and Delivery of Tools

In Step 3, the process of gripping and delivering the tools is carried out virtually and physically, where the tools classified as Free are removed, and those that have an occlusion are evaluated until the object of interest is found. The process continues until no tools are left on the work area.

The gripping process for both the desired object and the other tools, has the same sequence of movement of the robot, with the only difference that the endpoints vary according to their classification, leaving unwanted objects towards the right side of the robot (considering the front part that is in contact with the work area) and the desired ones from the center to the left side, as shown in Figure 15.

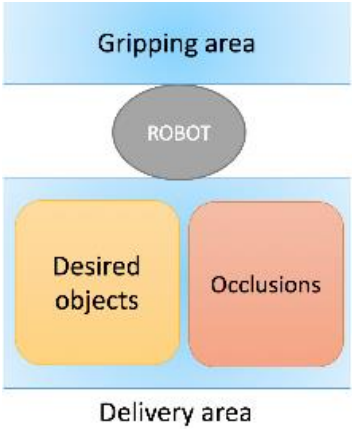


Figure 15. Delivery coordinates for the desired and unwanted tools by the user. Source: self-made

Based on tipoHerr value set in the previous step, the program determines if the element detected is an unwanted tool or the desired object. The value of tipoHerr determines the final position of the grasped object, selecting it from the arrangement PO as indicated in (3), where iOr is an indicator whose value equals ihd, if a desired object is grasped, or equals iho if it is grasped an unwanted tool, where ihd and iho are integer variables that count the number of tools delivered, either desired (ihd) or unwanted (iho).

$$PO(iOr, :, tipoHerr) \quad (3)$$

Once the grip and delivery points of the tools are determined, the virtual and physical robot are moved simultaneously and when finished, the manipulator is returned to its initial position, waiting to obtain the following grip coordinates.

2.7. Step 4: Search for New Tools

In Step 4, the tool search process performed in Step 1 is repeated and, in case of finding any element that had not been removed in the previous steps, the While cycle is restarted to repeat the process, otherwise it ends the program.

When the application detects that all detection boxes have been evaluated and considers that all tools have been removed with steps 2 and 3, Step 4 is executed, which seeks to confirm, using the Fast R-CNN and information three-dimensional, there are no tools left in the work environment.

In this step, a new capture of the RGB-D environment is performed and the Fast R-CNN is applied. If no tool is detected, the program ends, otherwise, the entire process of Step 1 is repeated and complemented with a three-dimensional analysis where, if the average height Z of any of the detection boxes is greater than one maximum expected (MaxZ), it is determined that in the table there are no tools and it is discarded as a grip point, otherwise, it is considered that there are still tools to evaluate and the While cycle is restarted.

In the same way, when elements classified as Occlusion are grasped, it is determined that the robot has finished removing all the elements only when the average height of the detection box is greater than MaxZ; otherwise, elements are still searched. In the cases of boxes classified as Free, a single tool is grasped, and it is considered that after removing it, there are no more elements, so the detection box is changed.

3. RESULTS AND ANALYSIS

The algorithm was tested under two different premises, one considering the quality of the detection of the elements, and another focused on the subjection of the tools, in order to test in the real application, the operation of the trained networks and the quality of the algorithm to remove occlusions.

In Table 4 was evaluated the quality of detection and grip of tools for different number of stacked objects, from 1 to 5, where different percentages of success and detection were defined, in which aspects such as:

Successes: Percentage of successful grips, where a tool, or tool stack was detected, and its grip coordinates were defined based on the three-dimensional information of the Kinect and the Fast R-CNN detection boxes. A successful grip is considered to be the one in which the robot takes the expected tool, and a failed grip is the one in which the coordinates obtained do not match with the location of the expected tool.

Detection: Percentage of detections, where the percentage of element stacks detected by the Fast R-CNN is evaluated with respect to the total groups of elements present in the work environment. It is considered as a successful detection when a detection box is defined on a tool stack, and a failed detection when no detection box is generated on the expected elements.

Reinforcement: Percentage of detections made in Step 4, where the percentage of stacks of elements detected by the Fast R-CNN is evaluated with respect to the total groups of elements present in the work environment after executing Step 4 once. It is considered as a successful detection when a detection box is defined on a tool stack, and a failed detection when no detection box is generated on the expected elements.

Table 4. Quality of Detection and Attachment of Tools

NUMBER OF TOOLS	Successes (%)	Detection (%)	Reinforcement (%)
1	100	71.43	
2	79.17	100	
3	85.19	78.57	90.48
4	88.89	64.29	
5	77.78	60.31	

The Fast R-CNN presented a great facility to detect stacks of two tools, and difficulties for stacks of greater height, especially more than 4 tools. This was because, at a higher height, the elements occupy more space in the image captured by the Kinect, which makes them tend to be perceived as background.

On the other hand, the algorithm manages to detect the exact grip coordinates for tools without occlusions without failing, but when adding height to the stack of elements, difficulties begin to appear. It was observed that the failures were mainly due to inconveniences in the calculation of the height of the tool, where the X, Y coordinates were right, but the height corresponded to the tool below the expected. This was due to

the fact that the presence of other elements, below the desired object, affects the calculation of the height, reducing its value.

The percentage of reinforcement demonstrates that after executing step 4 once, there is a high probability that all the tools in the environment are detected (greater than 90% accuracy), which implies that, in most cases, the program will be executed, maximum twice, to achieve its objective.

On the other hand, in Figure 16, an example of the application of DAG-CNN in the real environment is shown, during different functional tests, where it was observed that, despite the high percentage of accuracy obtained during the tests, the variance in the location of the detection boxes caused the quality of the classification to fall, especially for images of tools with cropped sections

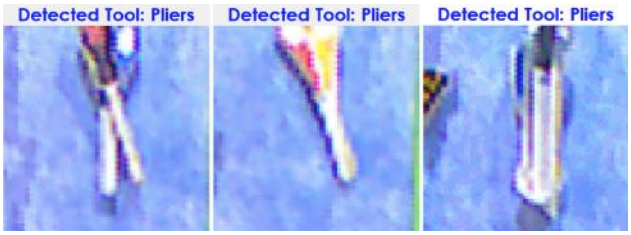


Figure 16. Classification of tools with failures due to detection frames. Source: self-made

4. CONCLUSION

The proposed system of detection, classification and grip of tools has the ability to run automatically against different working conditions, with a maximum of 5 stacked elements, but with greater efficiency for towers of up to 4 elements, located one just above the other.

The grip coordinates closely approximated the expected grip position, however, the presence of the occlusions altered the grip Z coordinate, leading the robot to grip an occluded tool, instead of grasp the one located above the stack.

The networks used managed to provide sufficient information of the state of the tools on the table, indicating if they present occlusions, and classifying them within the trained categories. Both networks are interrelated, which means that the DAG-CNN depends on the detection quality of the Fast R-CNN to ensure a good classification. For its part, the Fast R-CNN depends on the number of stacked elements, achieving better results when working with only two elements, maximum 3. The anchors defined in the RoI of the Faster RCNN let to the network discriminate the stacked objects successfully, the RoI location is employed for the desired tool classification through the DAG-CNN.

Regardless of the quality of the Fast R-CNN trained to detect which elements have occlusions and which do not, it should be taken into account that, since it is a two-dimensional analysis, it is complex to detect occlusions in cases where one tool completely covers the other, since, from this viewing angle, it is not possible to detect occlusion.

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 cirugia", from which the present work is derived.

REFERENCES

[1] 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).
[2] 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).
[3] S. K. Kim, M. Likhachev, “Planning for grasp selection of partially occluded objects”, In Robotics and Automation (ICRA), 2016 IEEE International Conference on, pp. 3971-3978, (2016).
[4] Y. C. Lin, S. T. Wei, S. A. Yang, L. C. Fu, “Planning on searching occluded target object with a mobile robot manipulator”, In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 3110-3115, (2015, May).
[5] T. Patten, M. Zillich, R. Fitch, M. Vincze, S. Sukkarieh, “Viewpoint evaluation for online 3-D active object classification”, IEEE Robotics and Automation Letters, vol. 1, no. 1, p. 73-81, (2016).

[6] J. H. Su, Z. Y. Liu, G. Yang, "Pose estimation of occluded objects with an improved template matching method", In First International Workshop on Pattern Recognition, vol. 10011, p. 1001115, International Society for Optics and Photonics, (2016).
[7] A. Ruchay, V. Kober, "A correlation-based algorithm for recognition and tracking of partially occluded objects", In Applications of Digital Image Processing XXXIX, vol. 9971, p. 99712, R. International Society for Optics and Photonics, (2016).
[8] S. Soleimanizadeh, D. Mohamad, T. Saba, A. Rehman, "Recognition of partially occluded objects based on the three different color spaces (RGB, YCbCr, HSV)", 3D Research, vol. 6, no. 3, p. 22, (2015).
[9] J. L. Chu, A. Krzyźak, "The recognition of partially occluded objects with support vector machines, convolutional neural networks and deep belief networks", Journal of Artificial Intelligence and Soft Computing Research, vol. 4, no. 1, p. 5-19, (2014).
[10] J. L. Chu, A. Krzyżak, "Application of support vector machines, convolutional neural networks and deep belief networks to recognition of partially occluded objects", In International Conference on Artificial Intelligence and Soft Computing, pp. 34-46, Springer, Cham, (2014).
[11] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks", In Advances in neural information processing systems, pp. 1097-1105, (2012)
[12] C. C. Aggarwal, "Neural networks and deep learning", Springer: Berlin, Germany, (2018).
[13] M. Schwarz, H. Schulz, S. Behnke, "RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features", In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pp. 1329-1335, (2015).
[14] R. J. Moreno, O. Avilés, D. M. Ovalle, "Red neuronal convolucional para discriminar herramientas en robótica asistencial", Visión electrónica, vol. 12, no. 2, (2018).
[15] S. Yang, D. Ramanan, "Multi-scale recognition with DAG-CNNs", En Computer Vision (ICCV), 2015 IEEE International Conference on. IEEE, p. 1215-1223, (2015).
[16] C. L. Zitnick, P. Dollár, "Edge boxes: Locating object proposals from edges", European Conference on Computer Vision, Springer, Cham, pp. 391-405, (2014). https://doi.org/10.1007/978-3-319-10602-1_26
[17] MATLAB, R-CNN, Fast R-CNN, and Faster R-CNN Basics, [Online] Available in: https://la.mathworks.com/help/vision/ug/faster-r-cnn-basics.html;jsessionid=9e182a3b05ce30852b1c5db52a7a (2018)
[18] R. Girshick, J. Donahue, T. Darrell and 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)
[19] M. Oquab, L. Bottou, I. Laptev, J. Sivic, "Is object localization for free?-weakly-supervised learning with convolutional neural networks", In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 685-694, (2015).
[20] A. Jin, S. Yeung, J. Jopling, J. Krause, D. Azagury, A. Milstein, L. Fei-Fei, "Tool Detection and Operative Skill Assessment in Surgical Videos Using Region-Based Convolutional Neural Networks", arXiv preprint arXiv:1802.08774, (2018).
[21] J. O. P. Arenas, R. Jiménez, P. C. U. Murillo, "Faster R-CNN for object location in a Virtual Environment for sorting task", International Journal of Online Engineering (iJOE), vol. 14, no. 07, p. 4-14, (2018).
[22] X. Wang, A. Shrivastava, A. Gupta, "A-fast-rcnn: Hard positive generation via adversary for object detection", In IEEE Conference on Computer Vision and Pattern Recognition, (2017).
[23] Z. Zhang, "Microsoft kinect sensor and its effect", IEEE multimedia, vol. 19, no. 2, p. 4-10, (2012).
[24] Matlab, U. S. G. The mathworks. Inc., Natick, MA, 1992, (1760).
[25] R. Girshick, "Fast r-cnn". In Proceedings of the IEEE international conference on computer vision, pp. 1440-1448, (2015).
[26] Muhammet Fatih Aslan, Akif Durdu, Kadir Sabanci, Meryem Afife Mutluer, CNN and HOG based comparison study for complete occlusion handling in human tracking, Measurement, Volume 158, 2020, 107704, ISSN 0263-2241, https://doi.org/10.1016/j.measurement.2020.107704.
[27] P. C. U. Murillo, J. O. P. Arenas, R. J. Moreno, "Implementation of a Data Augmentation Algorithm Validated by Means of the Accuracy of a Convolutional Neural Network", Journal of Engineering and Applied Sciences, vol. 12, no. 20, pp. 5323-5331, (2017).
[28] Paula Catalina Useche Murillo, Javier O. Pinzón-Arenas, Robinson Jiménez-Moreno. Obstacle Evasion Algorithm Using Convolutional Neural Networks and Kinect-V1. 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.
[29] Xiaobiao Dai, Junping Hu, Hongmei Zhang, Abubakar Shitu, Chunlei Luo, Ahmad Osman, Stefano Sfarra, Yuxia Duan, Multi-task faster R-CNN for nighttime pedestrian detection and distance estimation, Infrared Physics & Technology, Volume 115, 2021, 103694, ISSN 1350-4495, https://doi.org/10.1016/j.infrared.2021.103694.