

Imitation of the human upper limb by convolutional neural networks

Paula Useche Murillo¹, Robinson Jiménez-Moreno¹, Javier Eduardo Martínez Baquero²

¹Department of Mechatronic Engineering, Universidad Militar Nueva Granada, Bogotá, Colombia

²Faculty of Basic Sciences and Engineering, Universidad de los Llanos, Villavicencio, Colombia

Article Info

Article history:

Received Apr 3, 2024

Revised Jun 28, 2024

Accepted Jul 26, 2024

Keywords:

Convolutional neural networks
Fast R-CNN

Gestures recognition system
human motions

Motion capture data

Real-time imitation

ABSTRACT

The paper outlines the development of an algorithm focused on imitating movements of a human arm and replicating strokes generated by the user's hand within a working environment. The algorithm was crafted to discern the position of either the user's left or right arm, tracking each section (fingers, wrist, elbow, and shoulder) through a detection and tracking system. These movements are then replicated onto a virtual arm, simulating the actions of a cutting tool, generating strokes as it moves. Convolutional neural networks (CNNs) were employed to detect and classify each arm section, while geometric analysis determined the rotation angles of each joint, facilitating the virtual robot's motion. The stroke replication program achieved an 84.2% accuracy in stroke execution, gauged by the closure of the polygon, distance between initial and final drawing points, and generated noise, which was under 10%, with a 99% probability of drawing a closed polygon. A Fast region-based convolutional neural network (Fast R-CNN) network detected each arm section with 60.2% accuracy, producing detection boxes with precision ranging from 17% to 59%. Any recognition shortcomings were addressed through mathematical estimation of missing points and noise filters, resulting in a 90.4% imitation rate of human upper limb movement.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Robinson Jiménez-Moreno

Mechatronics Engineering Program, Faculty of Engineering, Universidad Militar Nueva Granada

Carrera 11 # 101-80, Bogotá D.C., Colombia

Email: robinson.jimenez@unimilitar.edu.co

1. INTRODUCTION

Research oriented to the detection and tracking of the human body [1] allows exploring applications associated with improving the living conditions of people, for example, with the monitoring of pain therapies through yoga as mentioned in [2]. Another approach to body tracking is in human machine interaction to avoid collisions between these [3]. Also by performing specific identifications such as hand and elbow, drone control can be performed as discussed in [4]. Aspect that at the specific level of pattern learning allows to generate trajectory tracking [5].

Nowadays, pattern learning and/or reinforcement learning techniques are used by deep learning algorithms to track, for example, convolutional neural networks (CNN) [6] applied to tracking mobile systems under the concept of internet of things [7] or satellite information [8]. For the topic of interest CNNs can be used to track groups of people [9] or a varied group of targets [10]. What is important in this case is to have one or several tracking targets and to train the CNN adequately with sufficient data as described in [11] and robust enough to support occlusions [12]. If required, many tracking systems have depth cameras for distance-based tracking [13], [14].

Aspects such as the tracking of geometries for railway safety by means of CNNs [15] and the tracking of body parts such as the eye [16], show the potential of CNNs in tracking tasks. In this work, the use of convolutional networks to identify and track parts of the human arm to replicate its movement in a robotic manipulator to perform specific cutting geometries oriented to industrial work for example steel sheets or similar is exposed. The tracking of the shoulder, elbow and hand to replicate a particular geometry with the robotic manipulator using convolutional networks by regions, to know its spatial location and match it with that of the robot, thus proposing the network architecture and the tracking method. Networks by regions such as the Faster region-based convolutional neural network (R-CNN) [17] are used for object classification and localization, with applications such as road safety [18], object collection by robots [19], among others [20]. This allows the use of these networks for tracking objects in complex environments where they need to be located.

The following is the development of an application focused on recognizing the position of a user's arm in real time, by means of an red, green and blue (RGB) supervisory camera and CNN-based pattern recognition methods. The objective is to generate a tracking system in order to move a virtual robotic manipulator responsible for mimicking the movements of the user, and trace the path of its end effector as if it were a cutting tool. This algorithm presents an innovative method of recognizing and tracking the movement of the human upper limb, by using Fast R-CNN to extract relevant sections of the arm such as the shoulder, elbow, wrist and fingers, and from their relative positions, control the rotation angles of each joint of the robot to achieve the desired displacement, in addition to estimating the missing points in the detection of the network to solve its low percentage of accuracy.

The article is divided into five sections: in the first section, a brief introduction to human movement imitation systems, the use of CNNs and their different variations and applications is given. In the second section, the operation of the algorithm from the first contact of the user with the interface to the completion of the program with the fully replicated stroke is explained. In the third section, the architecture of the Fast R-CNN trained for feature extraction is presented.

2. METHOD

Next, the methodology that allows obtaining the algorithm of imitation of movements of the human upper limb, whose purpose is to replicate the actual position of the arm during a video shot, and to generate strokes with the final effector, simulating a cutting tool, is presented. The developed algorithm is responsible for detecting and tracking the location of the relevant regions of the arm, such as the shoulder, elbow, wrist and fingertips. For this purpose, Fast R-CNN and RGB supervising camera are used, in order to extract a relationship of angles between each of the regions found and thus extract the degrees of rotation of each joint of a virtual robot to generate the imitation of the movement.

Simultaneously, the end effector of the robot generates a series of strokes during its displacement, which represent the cut made by a tool that would be located at its far end. The sequence of operation of the algorithm is shown in the diagram in Figure 1, where each step was demarcated with numbers from 0 to 4 and different functions were used for each of the tasks required by the program, which are explained in each of the following stages. The program interface is presented in Figure 2.

2.1. Step 0: Initialization of variables

The development required the use of an RGB camera to capture the working environment and the input image was resized from 480x640 pixels to 320x427 pixels, in order to speed up the arm position detection process by reducing the number of pixels to be evaluated. To generate the Fast R-CNN training database, the elements of the working environment were varied and the bare arm was captured from the shoulder to the fingertips, as the example shown in Figure 3, where it is possible to observe the ROIs marked on each section of the arm (shoulder, elbow, wrist, fingers). As shown in Figure 3, the arm must always enter from the right side of the image, and the hand must have its fingers fully extended in order to detect the fingers (Fingers).

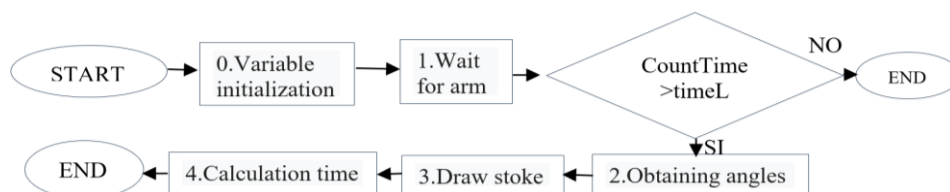


Figure 1. Algorithm flowchart

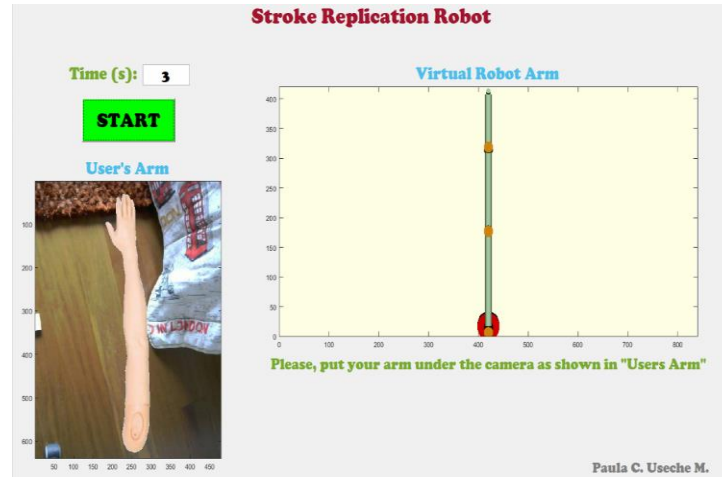


Figure 2. Real and virtual position of the arm within the interface (GUI)

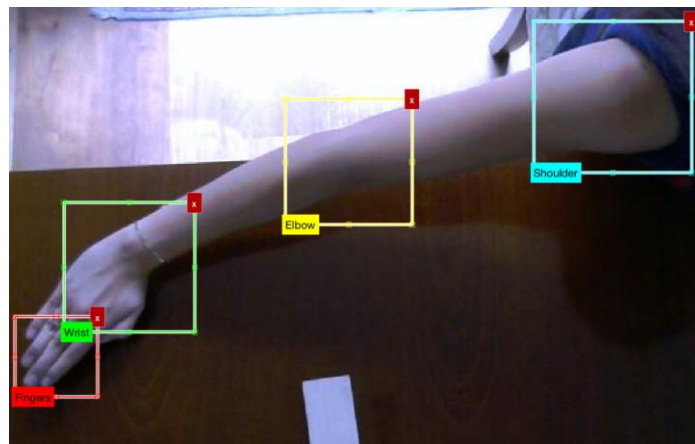


Figure 3. Arm capture with labels shoulder (blue), elbow (yellow), wrist (green), and fingers (red)

The first step of the algorithm consists of initializing the variables to be used throughout the program, which are listed in Table 1, where their function, name and initialization value are mentioned, with timeL being the completion time entered by the user in the graphical interface. Both the length of the links (L) and the angles of the robot (Q) were taken in order from the point anchored to the ground (equivalent to shoulder in Figure 3), to the end effector (equivalent to fingers in Figure 3). The values of L were taken as an average of the lengths of an adult arm as measured experimentally with different test subjects. In addition to the variables mentioned in Table 1, the camera, the virtual environment, and the Fast R-CNN trained for the algorithm (called detector) are initialized.

Table 1. Varibale initialization

Variable	Funtion	Value
L	Length of the links of the virtual manipulator [L(1) L(2) L(3)] (meters)	[0.36 0.3 0.2]
Q	Angles of the joints of the virtual manipulator [Q(1) Q(2) Q(3)] (radians)	[0 0 0]
CountTime	Timer to determine the end of the program (seconds)	0
timeL	Time limit for program completion	User-defined
ite	Iterator for plotting the trace generated by the robot	1
numN	Number of intermediate points between an end effector point and the next one	4
P	Array containing the position of the end effector at each detection [P(ite,1) P(ite,2) P(ite,3)]	--
boxes	Detection frames on the input image [x y width height] (pixels)	--
[x _c y _c]	Geometric center of the boxes	--
Dim	Input image dimensions [row column]. (pixels)	[320 427]

2.2. Step 1: Wait for the arm

During the process of detection and tracking of the upper limb, the sections found and/or estimated are plotted in "User's Arm", as shown in Figure 4 (where each color corresponds to the labels in Figure 3), in order to give feedback to the user about the recognition process, and the current location of the arm. Subsequently, the angles for the movement of the virtual robot are calculated using the "ArmAngles", which is explained below. The direct kinematics of the manipulator is used to obtain the position of the end effector according to the angles found, storing in the matrix $P(ite,:)$ that point. Finally, ite is increased by one (in each iteration) and Step 3 is executed.

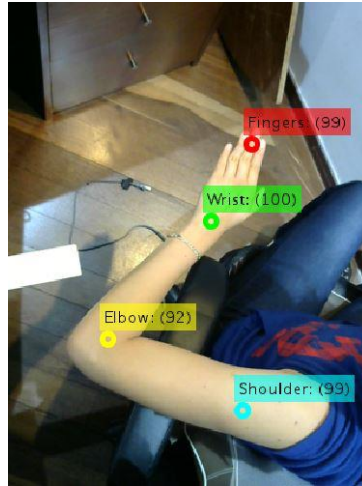


Figure 4. Detection of arm sections

2.3. Step 2: Obtaining angles

In the second step, the detector is used to obtain the arm sections in each iteration of the "while" of Figure 1 and the functions "uniqBoxes" and "AngulosBrazzo" are applied to select the detection boxes with the highest percentage of confidence in case of detecting more than one element per category, and to convert the inclination angles of the fingers, wrist, elbow and shoulder points, generated on the image, into rotation angles for the joints of the manipulator. The detector receives as input the image captured by the camera and generates as output the classification of the sections (labels), the percentage of confidence with which each element was classified (scores) and the detection boxes generated on the image, where the arm sections (boxes) were found. The "uniqBoxes" function evaluates the number of detection boxes generated per category, and if more than one is found, it compares the scores of these boxes and selects the one with the highest percentage.

A human centered motion detection (HCMD) matrix is calculated to store the coordinates of the geometric center of the detection boxes of each category, where each row of the matrix contains the positions $[xc \ yc]$ of the shoulder, elbow, wrist, and fingers sections, in that order, each section being a different row, numbered from 1 to 4. The coordinates $[xc \ yc]$ are obtained as shown in (1) and (2), yc being the vertical component and xc the horizontal one. Each row of boxes is composed of the elements previously indicated in Table 1, and as in HCMD, its rows correspond to each of the boxes detected in the classification categories.

$$x_c = boxes(:,1) - \frac{boxes(:,3)}{2} \quad (1)$$

$$y_c = boxes(:,2) + \frac{boxes(:,4)}{2} \quad (2)$$

The function "EstimatedAngles" seeks to estimate the angle of rotation of the point whose coordinates of the neighboring sections are known, e.g., the case where the shoulder and wrist were detected, but the elbow was not (PC1, PC2 and P_e for Figure 5, respectively). For which, it is required to know the coordinates of the neighboring points and the anterior position of the missing point (PCa1 or PCa2 in Figure 5). With this information, a triangle relation is made as shown in Figure 5, where the dimensions of each robot link in pixels (L_m calculated in (3) assuming that the arm spans the entire length $Dim(2)$ of the image) are known and, therefore, the internal angles of the triangle can be known, using the cosine theorem.

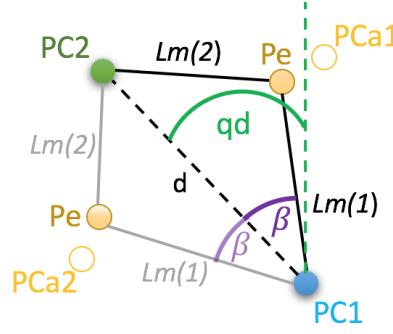


Figure 5. Ratio of triangles applying the cosine theorem

$$\begin{bmatrix} Lm(1) \\ Lm(2) \\ Lm(3) \end{bmatrix} = \begin{bmatrix} L(1) \\ L(2) \\ L(3) \end{bmatrix} * \frac{Dim(2)}{L(1)+L(2)+L(3)} \quad (3)$$

Where qd is the angle between the vertical and the line generated between the points $PC1=[PC1_x \ PC1_y]$ and $PC2=[PC2_x \ PC2_y]$, called d , which can be known by applying the distance equation between two points (4). Pe is the point to be estimated, $PCa1$ or $PCa2$ is the previous position of the point to be estimated, which is known, Lm are the lengths of the robot links in pixels ($Lm(1)$ and $Lm(2)$ for the case in which the elbow is estimated, $Lm(2)$ and $Lm(3)$ in case of estimating the wrist) and β is the angle between d and $Lm(1)$, which will allow estimating the first angle of the q vector.

$$d = \sqrt{(PC1_x - PC2_x)^2 + (PC1_y - PC2_y)^2} \quad (4)$$

Calculate β with the cosine theorem as shown in (5), and estimate the angle between Pe and $PC1$ ($qe1$ and $qe2$) as shown in (6) and (7), where $qe1$ or $qe2$ is selected as the estimated angle, according to its difference with respect to the previous angle for that link, choosing the one that is closer.

$$\beta = \cos^{-1} \left(\frac{Lm(1)^2 + d^2 - Lm(2)^2}{2 * Lm(1) * d} \right) \quad (5)$$

$$qe1 = qd - \beta \quad (6)$$

$$qe2 = qd + \beta \quad (7)$$

After estimating the first angle (qe), we proceed to obtain the estimated coordinates of the missing point (Pe) using (8), where h is 1 for the case in which the elbow is estimated, and 2 for estimating the wrist. In (8) what is done is to take the triangle formed by $Lm(h)$, the vertical and a horizontal passing through Pe and the vertical, and calculate its two legs, in order to add them to the current position of point $PC1$ and thus obtain Pe . Once the estimated coordinates of the missing point are known, Pe is stored in $HCMD(h+1,:)$, converting the unknown point into an estimated point.

$$P_e = \begin{bmatrix} y_e \\ x_e \end{bmatrix} = \begin{bmatrix} PC1_y + Lm(h) * \cos(qe) \\ PC1_x - Lm(h) * \sin(qe) \end{bmatrix} \quad (8)$$

In order to subtract the angles of arm inclination, is calculated to $q(1)$ (angle between $L(1)$ and the vertical) to obtain the first estimated angle. After $q(1)$, the estimated position of elbow is calculated using (9), where a rotation of elbow, $q(1)$ degrees with respect to shoulder is made, taking as initial position of elbow a point just above shoulder, assuming that the distance both is $Lm(1)$, as shown in (10), being $RotZ$ the rotation matrix with respect to Z in (11), in $HCMD$ matrix.

$$HCMD(2,:) = RotZ(q(1)) * (HCMDe(2,:) - HCMD(1,:)) + HCMD(1,:) \quad (9)$$

$$HCMDe(2,:) = [HCMD(1,1), HCMD(1,2) + Lm(1)] \quad (10)$$

$$RotZ(q(1)) = \begin{bmatrix} \cos q(1) & -\sin q(1) \\ \sin q(1) & \cos q(1) \end{bmatrix} \quad (11)$$

After obtaining and estimating the largest number of arm sections, the position points HCMD(i,:) and HCMD(i+1,:), being i the iterator of a for from one to three, are used to calculate the tilt angle of each pair of points using the tangent as shown in (12), where all angles stored in q were taken with respect to the vertical. Once the angles are obtained (either calculated or estimated), they are adjusted to be taken with respect to the robot's coordinate axis, where q(1) is taken with respect to the vertical, q(2) with respect to q(1), and q(3) with respect to q(2), using (13) through (15), forming the vector Q with the angles needed to mimic the motion of the human arm in the virtual environment. Finally, a noise filter was added to avoid abrupt changes in the position of the virtual arm between each iteration, denying its movement in cases where the variation of the angles between the previous and the current iteration is greater than 15°, whose value was experimentally selected to ensure smooth movements and avoid erroneous detections altering the polygonal trace.

$$q(i) = \tan\left(\frac{HCMD(i+1,2)-HCMD(i,2)}{HCMD(i+1,1)-HCMD(i,1)}\right) \quad (12)$$

$$Q(1) = q(1) \quad (13)$$

$$Q(2) = q(2) - Q(1) \quad (14)$$

$$Q(3) = q(3) - Q(1) - Q(2) \quad (15)$$

2.4. Step 3: Draw stroke

To draw the cut line made by the virtual arm with the end effector, straight lines were drawn between each pair of points of the P matrix and the start and end points of the trajectory were changed to green and red, as shows Figure 6(a). The first joint, anchored to the ground, represents the shoulder, the second the elbow and the third the wrist, with the final effector being the user's fingertips. The drawing is performed during each iteration, allowing the user to observe the cutting process as he/she moves the arm.

To prevent the number of points drawn in "Virtual robot arm" from altering the execution time of the algorithm, the cutting image was refreshed every 100 points, and the complete stroke was drawn at the end of the program. The Q angles are sent directly to the robot's joints to move it around the environment, no matter which arm of the user is being recognized, and if any of the arm sections are not detected, the missing angle is replaced by the angle of the previous iteration.

A Fast R-CNN type network was used for the detection and classification of the arm sections: shoulder, elbow, wrist, and fingers, which was trained with a total of 4900 training images, for 250 epochs, a MiniBatchSize of 49, an input image of 320x427 pixels and detection frames of dimensions 40x40 pixels. An initial acquisition of 700 training images, and 100 test images were made, where the user's gender, arm position, height relative to the camera, and working environment were varied, and then the images were augmented with the Data Augmentation program from [21], altering the light intensity, colors, and adding noise to the image.

The architecture for the network CNN [22] is presented in section 3 (Network 1), where small and square filters were used to extract the details of each section of the arm, and to differentiate each category by aspects such as the fingers of the hand, the change of area at the wrist between the hand and the arm, the folds of the elbow, and the contact of the shoulder with the clothing. The Fast R-CNN used (Arm_FastRCNN) achieved 60.2% accuracy for a total of 700 test images. Figure 6(b) shows an example of detection and classification of the human upper limb where it is possible to observe the detection of each of the sections of the arm, and the manipulator's imitation of the recognized position, in addition to the cut trace generated from the previous position to the current one.

2.5. Step 4: Calculation of time

In each iteration of the "while" in Figure 1, it is evaluated whether the arm is recognized or not, using a conditional that counts the number of sections detected by the detector, where a count of the time that the application has been executed is started, in case no section is found. If the arm is not detected within the working area, the time that has elapsed from the capture of the environment to the detection of the upper limb is stored in the variable CountTime and is added to the following iterations until a section is detected, in which case, CountTime is equaled to zero and steps 2 and 3 are executed. This conditional is only executed after the first detection of the arm (Step 1).

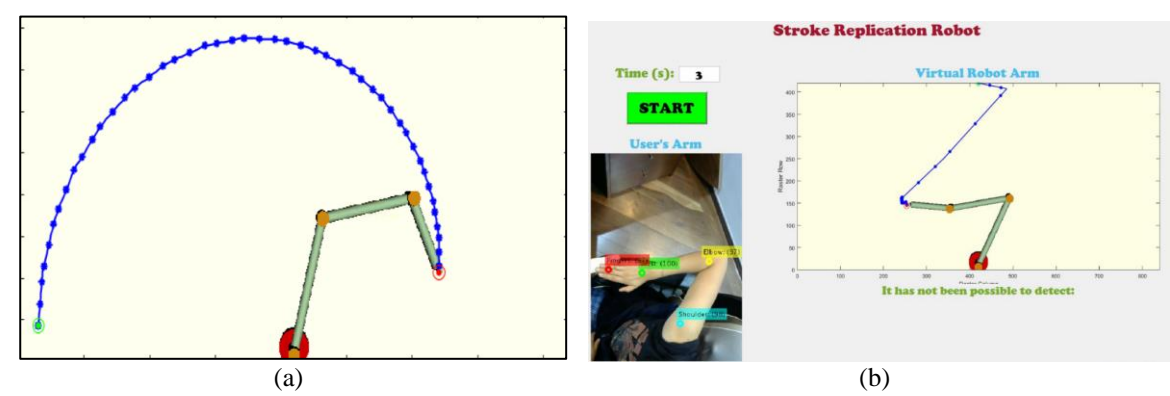


Figure 6. Stroke test (a) stroke example and (b) arm tracking for stroke

3. RESULTS AND DISCUSSION

The algorithm was tested with the realization of 12 polygons as shown in Figure 7, where the accuracy of the stroke was evaluated based on the closure of the drawing and the number of lines that remained outside the polygon, as shown in Table 2, where "Closure" represents the percentage of closure of the polygon, being 0% completely open (a straight line) and 100% completely closed. "Noise" represents the percentage of lines that remained outside the polygon and do not belong to the figure, obtained with respect to the dimensions of the polygon (corresponding to 100%) and without considering the start and end lines of the stroke. "Accurate closure" represents the distance between the end point and the initial point of the figure, 100% being completely together, and 0% being the drawing of a completely straight line between both points. The "Average Closure" was obtained as the average between the results of "Closure" and "Accurate Closure", and the "Stroke Accuracy" was calculated by subtracting the average noise percentage from the "Average Closure".

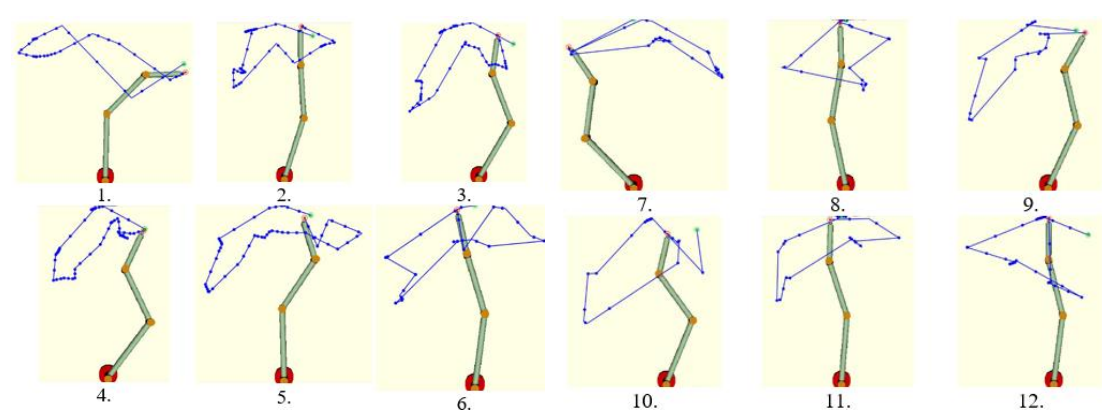


Figure 7. Polygons made with the program

Table 2. Precision of the program for the realization of multipolygonal traces

Polygon	Closing (%)	Noise (%)	Accurate closing (%)
1	100	0	95
2	96	10	90
3	100	15	90
4	100	10	100
5	96	0	98
6	100	15	90
7	100	10	50
8	100	5	98
9	95	5	95
10	100	0	80
11	100	15	98
12	100	30	80
Average	98.92	9.58	88.67
Average closing			93.79
Stroke precision			84.21

According to the results in Table 2, in 94% of the cases it will be possible to make a 99% figure closure, with a closing accuracy of 89%. 99% of the cases it will be possible to close the figure, with a closing accuracy of 89%. Additionally, in 84% of the cases it will be possible to draw a polygon without noise. noise, and in case of having noise, it will be less than 10%. On the other hand, the percentage of accuracy of imitation of upper limb movements was calculated, based on the percentage the time during which the position of the virtual arm resembles that of the upper limb. the virtual arm position resembles the real one by more than 90%. For this purpose, five polygons were 5 polygons were performed, two with the right arm, two with the left arm and one with both arms. the right arm, two with the left arm and one with both arms, obtaining the results shown in Table 3, where all time values are in seconds. time values are in seconds. The "Imitation percentage" was obtained by dividing the "Successful imitation time" by the "Total imitation time" and multiplying it by 100%, and the "Average" is the average percentage for the 5 cases presented. The "Total Imitation Time" was started counting from the first detection of the 4 arm sections, until the beginning of the count for the end of the program.

Table 3. Program accuracy for the imitation of human upper limb movements

Test arm	Total imitation time	Successful imitation time	Percentage of imitation (%)	Average
Right	31	27	87.10	90.39
Right 2	20	20	100.00	
Left	30	28	93.33	
Left 2	24	21	87.50	
Both	25	21	84.00	

In all tests, the virtual arm followed the real one with very small differences in position, determining as a not accurate imitation the positions in which 2 or more sections of the arm were not detected and estimated for more than half a second, which were the only situations in which a difference in the imitation was appreciated. As can be seen in Table 3, the program achieved 90.39% accuracy in the imitation of the movements of the human upper limb during the realization of a multipolygonal figure, thanks to the estimation of undetected points, which made up for the low percentage of accuracy of the Fast R-CNN.

Next, a comparison between three different R-CNN architectures trained for the recognition of arm parts is presented, whose results obtained by confusion matrices and recall Vs precision plots were used to select the final network architecture. All networks were trained for 250 epochs with a MiniBatchSize of 49, which was selected as higher values increased training time per epoch and reduced accuracy. Figures 8(a) to 8(c) shows the confusion matrices [23]–[25] (where 1 is shoulder, 2 is elbow, 3 is wrist, and 4 is fingers) for the networks trained with the architectures in Table 4. Where the first network is an R-CNN as shown in Figure 8(a) Arm_RCNN and the second ones are Fast R-CNN as shown in Figure 8(b), which presented a lower detection time than the R-CNN with better detection results (Thrid network Figure 8(c)), reason why it was decided to use a Fast R-CNN for the application.

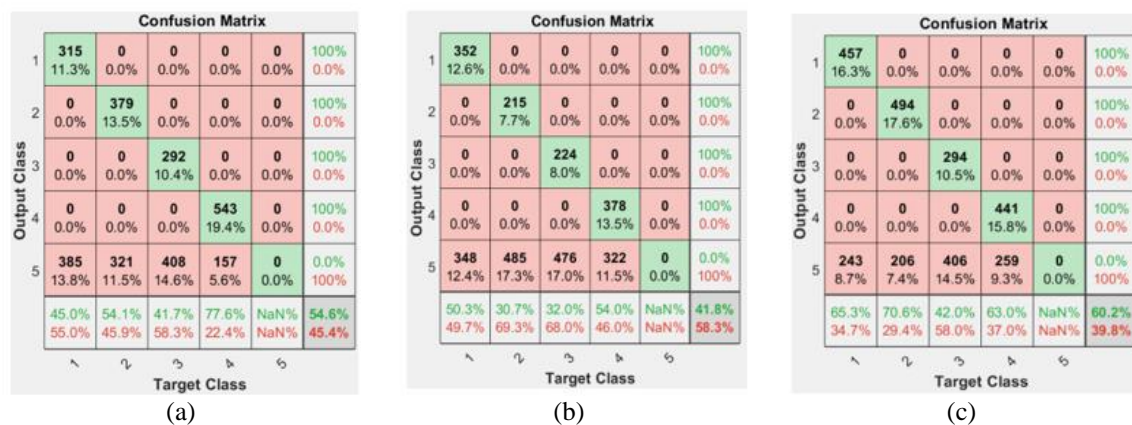


Figure 8. Confusion matrix (a) red arm_RCNN, (b) red arm_FastRCNN, and (c) red arm_FastRCNN

Table 4 present the architectures of the trained networks [26], where we tried to vary the size of the filters, the order of the layers and the depth of the networks. For all cases, the dimensions of the detection frame were 40x40 pixels, since this is the minimum size of the labels generated on the training images. Where C corresponds to a convolution layer, R to the rectified linear units, B to a batch normalization, MAXP to one of Maxpooling, FC to a fully connected, DROP to a dropout, and SOFT to a softmax.

Table 4. CNN architectures

NETWORK 1				NETWORK 2				NETWORK 3			
LAYERS	Filter Size	Stride	Number Filters	LAYERS	Filter Size	Stride	Number Filters	LAYERS	Filter Size	Stride	Number Filters
C+B+R	6x6	1	16	C+B+R	6x6	1	16	C+B+R	3x3	1	16
C+B+R	5x5	1	64	C+B+R	5x5	1	64	MAXP	2x2	2	--
C+B+R	4x4	1	128	MAXP	3x3	2	--	C+B+R	3x3	1	256
MAXP	3x3	2	--	C+B+R	4x4	1	128	MAXP	3x3	2	--
C+B+R	3x3	1	256	C+B+R	3x3	1	256	C+B+R	3x3	1	512
C+B+R	3x3	1	512								
FC1+RELU+DROP											
FC2+RELU+DROP											
FC3+SOFT											

Observing that none of the networks exceeded 61%, the network with the highest percentage was taken and trained for an additional 200 epochs, however, its accuracy dropped to 59.5%, with no noticeable improvement in frame detection percentage. We tried to train that same network with a larger MiniBatchSize, and with a smaller one, obtaining percentages of 43% and 43.1%, respectively (Arm_FastRCNN_3 being the network with 43% accuracy). To verify that the percentage reduction was not due to over-training, the new trained networks were tested with the training database, obtaining about 44% accuracy, which allows us to deduce that the network did not memorize.

Due to these results, it was decided to evaluate the activations [27] of the trained networks to observe the quality of the information extracted by each of them and from there determine the structure required for the final network. The activations showed that these architectures did not manage to focus on learning the arm, but captured other types of features such as the wood, the chair and the body, while the Arm_FastRCNN, focused on the arm and part of the background. For any of the trained networks, the most difficult section of the arm to recognize is the elbow, due to the multiple positions it can adopt, from fully stretched to tucked towards the chest, in addition to the upward or downward tilt (with respect to the ground), which can occur during the movement of the arm. Similarly, the displacement of Wrist and its rotation throughout the image also led to multiple representations, some of them similar to other parts of the arm such as Elbow.

Due to these detection difficulties, the decision was made to make the process of "Point Estimation" as a Tracking system for the algorithm, in order to keep track of as many parts of the arm as possible for a robot teleoperation closer to the real one, over the lack of information generated by the network. Assuming that the point estimation conditionals make up for the low detection of elbow, the confusion matrix of Arm_FastRCNN was calculated for the remaining 3 sections of the arm, where the accuracy rose to 66.3%.

4. CONCLUSION

The tracking system (point estimation) implemented in the program, allowed reaching a percentage of recognition and replication of movements of the human upper limb of up to 90% accuracy, leading the program to achieve more than 98% of closure in the multipolygonal figures, with a percentage of noise of less than 10%, despite using a Fast R-CNN with only 60% accuracy. Concluding that the action of network recognition can be strengthened within the application. The wide variation of positions that the elbow can adopt throughout the application, and the high degree of similarity between the different sections of the arm and its multiple positions (both for the right and left arm), generated difficulties in the training of the network, due to the demanding need not to confuse the sections with each other, nor with the background, and to achieve an accuracy in the detection frames that would allow the position of the virtual arm to be similar to the real one. Due to these complications, it was preferred to select a network with an accuracy of 60% instead of another with 72%, because of the accuracy of the Arm_FastRCNN in the generation of the detection frames for each category, where a smaller number of frames were recognized, but closer to what was expected. The use of larger input images than those used in the training database for a Fast R-CNN, may increase its accuracy in the classification of the network categories, however, it does not mean that it increases in the same way the accuracy of the detection frames, nor the overall performance of the program performed for the replication of

traces and imitation of human upper limb, so it is essential, for this type of applications, to evaluate both the accuracy of the network and the accuracy in the detection of each of its categories.




ACKNOWLEDGEMENTS

The authors are grateful to Universidad Militar Nueva Granada and Universidad de los Llanos.




REFERENCES

- [1] Q. Wu, "Research on human body detection and tracking algorithm based on kinect," in *2021 International Conference on Electronic Information Technology and Smart Agriculture, ICEITSA 2021*, 2021, pp. 7–10, doi: 10.1109/ICEITSA54226.2021.00011.
- [2] A. A. Kupin, S. Banerjee, N. Banerjee, S. H. Roy, J. C. Kline, and B. Shiwani, "System architecture for VR yoga therapy platform with 6-DoF whole-body avatar tracking," in *2024 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality, AIxVR 2024*, 2024, pp. 360–366, doi: 10.1109/AIxVR59861.2024.00062.
- [3] N. A. S. N. Nandasena, W. A. A. Vimukthi, H. M. K. K. M. B. Herath, R. Wijesinghe, and S. L. P. Yasakethu, "Real-time upper body motion tracking using computer vision for improved human-robot interaction and teleoperation," *Moratuwa Engineering Research Conference, MERCon*, pp. 201–206, 2023, doi: 10.1109/MERCon60487.2023.10355479.
- [4] S. M. L. Gioi, G. Loianno, and F. Cordella, "Robust upper limb kinematic reconstruction using a RGB-D camera," *IEEE Robotics and Automation Letters*, vol. 9, no. 4, pp. 3831–3837, 2024, doi: 10.1109/LRA.2024.3373236.
- [5] H. Xu, J. Fan, H. Ma, and Q. Wang, "Semiparametric musculoskeletal model for reinforcement learning-based trajectory tracking," *IEEE Transactions on Instrumentation and Measurement*, vol. 73, pp. 1–16, 2024, doi: 10.1109/TIM.2024.3370760.
- [6] Z. Feng and P. Wang, "A model adaptive updating kernel correlation filter tracker with deep CNN features," *Engineering Applications of Artificial Intelligence*, vol. 123, 2023, doi: 10.1016/j.engappai.2023.106250.
- [7] L. A. Hussain, S. Singh, R. Mizouni, H. Otrók, and E. Damiani, "A predictive target tracking framework for IoT using CNN–LSTM," *Internet of Things*, vol. 22, 2023, doi: 10.1016/j.iot.2023.100744.
- [8] Q. Zhu, X. Huang, and Q. Guan, "TabCtNet: Target-aware bilateral CNN-transformer network for single object tracking in satellite videos," *International Journal of Applied Earth Observation and Geoinformation*, vol. 128, 2024, doi: 10.1016/j.jag.2024.103723.
- [9] M. Babae, Z. Li, and G. Rigoll, "A dual CNN–RNN for multiple people tracking," *Neurocomputing*, vol. 368, pp. 69–83, 2019, doi: 10.1016/j.neucom.2019.08.008.
- [10] G. Li *et al.*, "One-shot multi-object tracking using CNN-based networks with spatial-channel attention mechanism," *Optics and Laser Technology*, vol. 153, 2022, doi: 10.1016/j.optlastec.2022.108267.
- [11] Y. Wang, X. Wei, X. Tang, H. Shen, and L. Ding, "CNN tracking based on data augmentation," *Knowledge-Based Systems*, vol. 194, 2020, doi: 10.1016/j.knsys.2020.105594.
- [12] M. F. Aslan, A. Durdu, K. Sabanci, and M. A. Mutluer, "CNN and HOG based comparison study for complete occlusion handling in human tracking," *Measurement: Journal of the International Measurement Confederation*, vol. 158, 2020, doi: 10.1016/j.measurement.2020.107704.
- [13] Y. Wang, X. Wei, H. Shen, L. Ding, and J. Wan, "Robust fusion for RGB-D tracking using CNN features," *Applied Soft Computing Journal*, vol. 92, 2020, doi: 10.1016/j.asoc.2020.106302.
- [14] Y. Wang, X. Wei, L. Luo, W. Wen, and Y. Wang, "Robust RGB-D tracking via compact CNN features," *Engineering Applications of Artificial Intelligence*, vol. 96, 2020, doi: 10.1016/j.engappai.2020.103974.
- [15] X. Wang, Y. Bai, and X. Liu, "Prediction of railroad track geometry change using a hybrid CNN-LSTM spatial-temporal model," *Advanced Engineering Informatics*, vol. 58, 2023, doi: 10.1016/j.aei.2023.102235.
- [16] N. Gunawardena, J. A. Ginige, B. Javadi, and G. Lui, "Performance analysis of CNN models for mobile device eye tracking with edge computing," *Procedia Computer Science*, vol. 207, pp. 2291–2300, 2022, doi: 10.1016/j.procs.2022.09.288.
- [17] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, ICCV 2015, pp. 1440–1448, doi: 10.1109/ICCV.2015.169.
- [18] F. Cui, M. Ning, J. Shen, and X. Shu, "Automatic recognition and tracking of highway layer-interface using Faster R-CNN," *Journal of Applied Geophysics*, vol. 196, Jan. 2022, doi: 10.1016/j.jappgeo.2021.104477.
- [19] J. O. P. Arenas, M. R. Jiménez, and P. C. U. Murillo, "Faster R-CNN for object location in a virtual environment for sorting task," *International Journal of Online Engineering*, vol. 14, no. 7, pp. 4–14, 2018, doi: 10.3991/ijoe.v14i07.8465.
- [20] L. H. Li and R. Tanone, "Ensemble learning based on CNN and transformer models for leaf diseases classification," in *Proceedings of the 2024 18th International Conference on Ubiquitous Information Management and Communication, IMCOM 2024*, 2024, pp. 1–6, doi: 10.1109/IMCOM60618.2024.10418393.
- [21] M. P. C. Useche, J. P. Arenas, and 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, doi: 10.3923/jeasci.2017.5323.5331.
- [22] P. A. Javier Orlando, J. M. Robinson, and J. E. M. Baquero, "Comparison of convolutional neural network models for user's facial recognition," *International Journal of Electrical and Computer Engineering*, vol. 14, no. 1, pp. 192–198, 2024, doi: 10.11591/ijece.v14i1.pp192-198.
- [23] H. Yun, "Prediction model of algal blooms using logistic regression and confusion matrix," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 3, pp. 2407–2413, 2021, doi: 10.11591/ijece.v11i3.pp2407-2413.
- [24] L. E. Pomme, R. Bourqui, R. Giot, and D. Auber, "Relative confusion matrix: efficient comparison of decision models," in *Proceedings of the International Conference on Information Visualisation*, 2022, vol. 2022, pp. 98–103, doi: 10.1109/IV56949.2022.00025.
- [25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017, doi: 10.1145/3065386.
- [26] T. D. Trinh, P. C. L. T. Vu, and P. T. Bao, "Comparing mask R-CNN backbone architectures for human detection using thermal imaging," *International Journal of Electrical and Computer Engineering*, vol. 14, no. 4, pp. 3962–3970, 2024, doi: 10.11591/ijece.v14i4.pp3962-3970.
- [27] P. K. Balla, A. Kumar, and R. Pandey, "A 4-channelled hazy image input generation and deep learning-based single image dehazing," *Journal of Visual Communication and Image Representation*, vol. 100, 2024, doi: 10.1016/j.jvcir.2024.104099.




BIOGRAPHIES OF AUTHORS

Paula Useche Murillo    is a Mechatronics Engineer graduated with honors in 2017 from Universidad Militar Nueva Granada in Bogotá, Colombia, M.Sc. in Mechatronics Engineering and works in this project as a research assistant in the Mechatronics Engineering program. She published two papers in the International Journal of Applied Engineering Research (IJAER) in 2016 about the implementation of Myo Armband in applications for manipulation of a robot arm through myoelectric signals. Her research focuses on the use of convolutional neural networks for object recognition, and image processing for grasp detection and trajectory planning. She can be contacted at email: est.paula.useche@unimilitar.edu.co.



Robinson Jiménez Moreno    is an Electronic Engineer graduated from Universidad Distrital Francisco José de Caldas in 2002. He received a M.Sc. in Engineering from Universidad Nacional de Colombia in 2012 and Ph.D. in Engineering at Universidad Distrital Francisco José de Caldas in 2018. His current working as Assistant Professor of Universidad Militar Nueva Granada and research focuses on the use of convolutional neural networks for object recognition and image processing for robotic applications such as human-machine interaction. He can be contacted at email: robinson.jimenez@unimilitar.edu.co.



Javier Eduardo Martínez Baquero    is an Electronic Engineer graduated from Universidad de los Llanos in 2002. Posgraduated in Electronic Instrumentation from Universidad Santo Tomas in 2004, posgraduated in Instrumentation and Industrial Control at Universidad de los Llanos in 2020 and M.Sc. in Educative Technology and Innovative Media for Education at Universidad Autonoma de Bucaramanga in 2013. His current working as Associated Professor of Universidad de los Llanos and research focuses on Instrumentation, Automation, Control and Renewable Energies. He can be contacted at email: jmartinez@unillanos.edu.co.