VOL. 16, NO. 16, AUGUST 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

3-D PATH ESTIMATION FOR A ROBOTIC ARM FOCUSED ON FOOD COLLECTION USING A CNN REGRESSION APPROACH

Javier O. Pinzón-Arenas[1], Robinson Jiménez-Moreno[1] and Giovanna Sansoni[2]
[1]Faculty of Engineering, Nueva Granada Military University, Bogotá D. C., Colombia
[2]Faculty of Engineering, Degli Studi di Brescia University, Brescia, Italy
E-Mail: u3900231@unimilitar.edu.co

ABSTRACT

To achieve autonomy of robotic agents, the estimation or planning of paths is one of the most relevant aspects, since this allows the robots to carry out the movements required to achieve a specific task within a work area. Various increasingly robust techniques have appeared and/or have been applied in this area, such as convolutional neural networks (CNN). This paper presents the implementation of a cascade CNN set, in such a way that, based on an input image that consists of a food dish, a 10-point path is obtained in a three-dimensional space, forming a path that enters to a given area and generates a movement of food collection. The first network, responsible for segmenting the image to obtain the area of the required food is called ResSeg. The second network is a ResNet-50 model modified to be applied in a regression framework, whose function is to estimate the path of the robot to perform the collection task, based on the segmentation obtained from the previous network. An accuracy of 90.33% is obtained in the path estimation, with a general deviation of 12.55 mm, taking into account a deviation threshold of ± 20 mm. Likewise, the network is tested in real-time in a virtual environment, applying the estimated path to a robotic manipulator of 4 degrees of freedom, being able to demonstrate the reliability and smoothness of the estimated path in the execution of the task.

Keywords: face recognition, convolutional network, CNN architectures, transfer learning.

1. INTRODUCTION

The control of robotic agents, whether static or mobile, has been one of the key points in their integration into highly complex tasks. The first control approach of the robots was given manually, that is, an operator is responsible for carrying out the movements of the robots, either through tools, joysticks and/or buttons. This is evidenced mainly in robots specialized in surgeries [1], or by means of signals or commands acquired by sensors, either by means of vision devices in order to recognize the operator movements [2] or electrical signals of the body [3]. However, there are tasks that require the robot to act autonomously, i.e. that an intermediary operator is not required to perform any action for it to act. For instance, in the case of facilitating high-risk jobs or even to improve the quality of life of people, and that this can be applied in any type of environment, including industry [4]. For this reason, it began to cover a type of control that allows the robotic agent to act on its own, e.g. for self-driving cars [5] or companion robots [6]. Within this field, there is a problem that has taken great relevance, called path planning, which consists in identifying a path that an agent must follow in order to reach a specific goal or point, especially for tasks where it is required to go from one place to another or perform manipulation of objects, as well as avoid obstacles that may be present in the environment, which makes it a crucial topic for the autonomy of robots.

For this reason, various methodologies have been developed to be able to give the system that controls the robot, the autonomy of making the decision of what path it should take to execute its task, regardless of the environment in which it is located. Within these, we have the classical approaches, which are mainly based on interconnection of lines, forming meshes that allow to know the possible paths that can be followed [7,8], taking into account that the environment is known as a whole and has a global perspective of it [9]. These approaches are mainly divided into 3 categories. The first one, called roadmaps, consists of joining, through lines, the different possible paths existing through nodes based on the characteristics of the environment, joining the starting point with the endpoint, using a kind of graph that describes the route [7]. The second one is cell decomposition, which is based on dividing the environment into small regions by means of a mesh, reducing the size of the mesh if an obstacle is found within the region, until it forms free spaces, creating the way to follow [7]. The last one is the artificial potential field, which, unlike the previous two, does not require a global perspective of the environment, but is done locally, generating fields or lines of repulsion from the agent, in such a way that it avoids collisions on its way to its target [9, 10].

Other more robust methods are found in heuristic and artificial intelligence approaches. One of the best known in the heuristics is the A-star algorithm [9, 11], whose objective is to find the shortest path from an initial point to an endpoint, by means of nodes that have weighted values that are updated while moving along the path, going for the smallest values. As for the artificial intelligence techniques used to plan paths, there are genetic algorithms, behavior-based techniques such as ant colony and swarms. For example, the genetic algorithm, in general, is based on the functioning of genes through the crossing, mutation, and selection of individuals in a population, so that the best individuals (best paths) are selected until a final path is obtained. On the other hand, there are the methods based on artificial neural networks [8, 9], which in general terms are intelligent systems

composed of neurons or interconnected nodes that allow learning characteristics of the environment, they are used as mappers of an environment, in other words, they can identify relations of the work area and describe possible restrictions existing in it [8,9]. An example of this can be seen in [11], where a neural network is proposed to plan the path of a humanoid robot using a global camera, so that it reaches a certain point, avoiding colliding against obstacles arranged in the workspace. However, due to the random nature of the environment, the size and amount of possible architectures of neural networks make neuron weights difficult to be set, making them a complex method of being applied to path planning.

Although path planning techniques were mainly focused on terrestrial mobile robots due to their ease of application since they commonly move in a flat environment, they have been used in other types of robotic agents, e.g. autonomous air vehicles [12]. This can be seen in [13], where a combination of artificial potential fields and optimal control is used to generate the path that a quadrotor must follow during its flight. Another example is presented in [14], whose planning model is implemented through the union between genetic and ant colony algorithms, taking into account information from the environment acquired by sensors located in the air agent.

Similarly, path planning is applied to static agents, in other words, robots that do not move through an environment, but a part of their body is responsible for performing the task, having as reference the base of their body, more specifically, robotic manipulators. In these agents, the planning is done with respect to its end effector, so that it reaches its desired objective or position. What makes the task more complex is the use of path planning techniques for a three-dimensional environment. This can be seen in the work presented in [15], where a path planning algorithm called Flood Fill is modified, which is normally used in 2-D to solve mazes, in order to be applied in a 3-D environment. On the other hand, in [16], an ant colony algorithm is used to establish the path of a 6 degree of freedom (DoF) manipulator, so that it does not collide with obstacles while performing a collection task. These methods have also been used in assembly and disassembly of products in the manufacturing industry, where it is necessary to find paths that allow to locate or remove parts of a structure [17]. The applications are even in tasks of high level of difficulty, such as welding [18], using methods based on artificial intelligence in conjunction with control models, since it not only requires estimating the path but also taking into account the dynamics of the robot [19].

However, the requirements and the level of complexity to generate paths have been increasing, since the tasks are becoming more difficult. For this reason, new more robust techniques have begun to be used to plan the path of robotic agents. One of these is called convolutional neural network (CNN) [20], whose main feature is that, from an input image, patterns can be extracted from it and learn them to generate a response. Unlike a conventional neural network, CNN does not require layers of a great number of neurons, thanks to its structure composed of convolution filters, which are responsible for learning the characteristics of objects or environments. Its first integration in the operation of robots is given to guide the user in the control of the robot manually, either by means of the location of the end effector of a manipulator [21] or to indicate what action to take by means of hand gestures [22].

Within the autonomous control of robotic agents, CNNs began to be used as support for tracking the object or person to follow, in order to generate the path of mobile agents using a local vision system [23]. An example can be seen in [24], where a CNN is implemented based on a regression output, which is part of a path planning system for a robotic chair. In that development, CNN is responsible for granting the position and orientation of the mobile agent, based on an entry image of the current position of the chair. However, due to its great capability to extract features, it was integrated into a model called value interaction network [25, 26], obtaining a high degree of accuracy in the estimation of paths, close to 99%, to reach an objective point. Finally, CNN architectures specialized in path planning began to be developed. For instance, there is the work presented in [27], which describes the implementation of a CNN architecture that directs a mobile robot through a path until it reaches a specific point, based on an input satellite image. However, this network does not give a complete path, but indicates the robot a point-to-point movement, i.e. the satellite image of the robot's initial position is entered into the network, giving as output a direction of movement to arrive at a point along the way; then, a new image with the updated position is entered, to generate a new direction of the path, repeating the cycle until reaching the target point. Another methodology that has begun to be exploited is the combination of CNN with reinforcement learning, where, with respect to an image and information of the robot's position by means of sensors, the torques of the arm motors are calculated to generate the trajectory of the robot [28,29]. However, that application focuses more on trajectory planning that is the generation of position, speed, and acceleration, but not on the estimation of the path the robot should follow [7].

From the analysis of the state of the art presented, it is possible to infer that CNNs have the possibility of being applied in the estimation of complete paths with a high degree of reliability, especially for static robotic agents, which has not been explored in the literature in relation to 3-D spaces. Similarly, current techniques focus on reaching a specific endpoint, without the capability to generate an additional path within an area to generate a task, without adding intermediate target points. This leads to the implementation of a technique that allows estimating these types of paths, where the objective is an area, and within this, a specific movement should be performed. Under these arguments, it is proposed to implement a CNN that allows the path that a manipulator should follow in a 3D environment, which must enter the target area and do a kind of journey within it. The application approach of this work is the collection of a type of food located on a dish, using an assistive robotic

arm with 4 degrees of freedom. In the same way, it is proposed to use a regression output, resulting in the path that the manipulator must follow, consisting of 10 points located on the X, Y and Z axes, that is, 30 output values.
The paper is divided into 4 sections, where the first one is the present introduction. In section 2, the database used is described, together with the proposed architecture to be used, its training and respective validation, and the virtual test environment developed. In section 3, the experimental results of the trained network applied within the virtual environment are shown. Finally, section 4 gives the conclusions reached.

2. METHODS AND MATERIALS

The objective of this development is to estimate a 3-D path that a 4-DoF manipulator robot must follow to collect food, in such a way that, from a photo of the food served on a dish, the points through which the end effector will pass are obtained. For the implementation of the work, firstly, the database to be used is proposed. To obtain the path, the use of a cascade configuration of 2 convolutional neural networks is proposed. The first network is responsible for segmenting the existing food on the dish, which locates the required food through a mask image (or binary image). The second network receives the mask obtained and with this, it estimates the path that the manipulator must follow, generating 10 points in a 3-D space through which the end effector passes. Following this, a virtual environment is created that contains the robot to be used, an image of the food and a camera in charge of capturing the photo of the food, in order to verify the paths estimated by the network. Each step is described below.

2.1 Dataset

In order to train a convolutional neural network in a regression framework, a database that is divided into two sections is used. In the first section, the images to be entered into the neural network are established. The images that make up an initial database are of food dishes called "executives," which contain 5 types of food. The photos were taken at 550 millimeters of distance between the camera and the table, with a resolution of 480x360 pixels and RGB format. Taking into account that there are different types of food on the dishes, it is decided to focus the collection on one of these, which is rice. However, it is necessary to know whether or not there is rice since the dishes can be recently served or the food has already begun to be eaten, as well as determining the location of the food to be collected, so it is required to use a detection method that allows establishing these two parameters. For this reason, a neural network specialized in segmenting the food is used, called ResSeg [30], which allows the rice to be segmented from the rest of the food as well as from the plate, which is white and may generate confusions. Using the ResSeg-CNN Regression cascading network method reduces the computational cost, since the image is significantly large for a CNN, and at the same time it reduces the possibility of failure to recognize where the food is really located, in order to generate the path.

With the results obtained from the segmentation, the mask or area where the rice is located is extracted. This resulting mask is the one that enters the neural network to estimate the path of the robot. Using the mask helps to know more precisely where the rice is located to avoid possible confusion with other elements within the environment. In total, 236 masks are obtained, belonging to each image of the original database. An example of the database is shown in Figure-1, where the original photo, the segmentation performed and the final mask obtained are presented

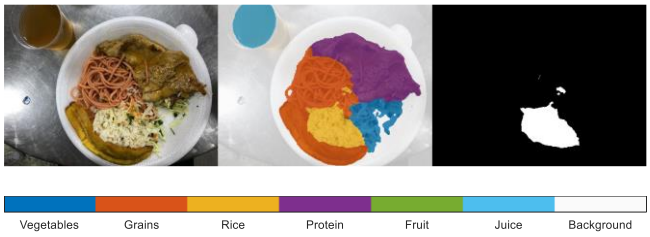

Figure-1. Food segmentation and obtained mask of the rice.

The second section of the database refers to the desired output or path. For this, 10 points are placed manually, which make up the path to follow. The path contains an initial point, located in the lower central part of the image (where the initial position of the robot's end effector is proposed), reaching to the area where the rice is located, to finally perform the collection to an endpoint, as shown in Figure-2. In the same way, the proposed path can be seen in a three-dimensional way, taking into account that the initial point of the final effector, in local coordinates of the robot, is defined as (0, 106, 246) mm, taking into account the dimensions of the robot, which are presented later. Each point on the path consists of an array of 3 data belonging to the values on the X, Y and Z axes in millimeters, i.e. as desired output, there are a total of 30 values. In the case that rice is not found in the image or the amount is very small, the robot will remain in its initial position, that is, all 10 points are located at the starting point. The process of manual positioning of the path is done in all the images of the database.

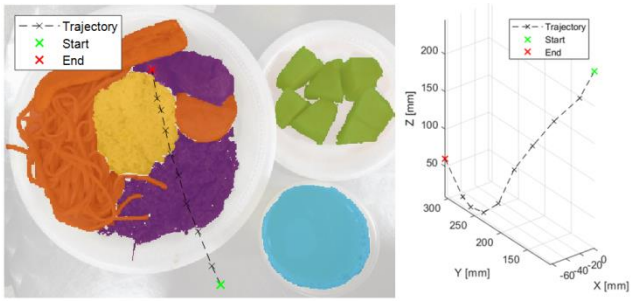

Figure-2. New path start point and the 3-D view.

With the input and output of the neural network set in the database, it is divided into two sets. The first set, belonging to the training dataset, contains approximately

85% of the data, which are 200 images and paths. The second set consists of approximately 15% of the data (36 images and paths), which is established for network tests.

2.2 CNN Training

In a previous work [31], three CNN with different architectures were evaluated to verify their performance generating paths with a 2-D approach. The architectures evaluated were a so-called Shallow CNN, the AlexNet model, and the ResNet-50 model. From them, the best performance was obtained by ResNet-50 [32], with a 62.5% accuracy, so it is the one selected to estimate the path in 3-D.

To couple the ResNet-50 to a regression framework and to a larger input to the one set originally for this network, modifications are made to its first and last layers. For this, the parameters of the first convolution layer are replaced, so that it fits the required size (480x360 pixels). The last three layers, belonging to the fully connected, classification and SoftMax layer, are removed since these are designed for object recognition, being replaced by a fully connected output layer with 30 neurons, i.e. the information of the coordinates in the X, Y and Z axes of the path points, and a regression layer. The modified and new layers are initialized with random weights by means of the He method [33].

The regression is calculated using the mean square error or MSE (1), where N is the number of responses, y_d is the desired output, y is the estimate obtained by the network in the observation i. The loss function is calculated with half of the MSE (2).

$$MSE = \sum_{i=1}^{N} \frac{(y_{d_i} - y_i)^2}{N} \quad (1)$$

$$loss = \frac{1}{2}\sum_{i=1}^{N} \frac{(y_{d_i} - y_i)^2}{N} \quad (2)$$

It should be noted that the complexity in the 2-D approach was lower, in other words, there were only 20 output values (the 10 points on the X and Y axes) and these were integers (given in pixels). For this reason, the amount of path data and complexity of these increased, that is, apart from the values defined in the X and Y axes for the 10 points of the path, the values referring to the depth or Z-axis were added, obtaining a total of 30 values. In addition, the values are no longer in pixels (integer values) but in millimeters (real values). Due to this, an iterative procedure is performed to verify the learning rate so that the network does not fall into the exploding gradient problem and to observe its early behavior.

Finally, the final values of the training parameters are set, with a learning rate of 10-4 with a reduction factor of 0.5 every 40 epochs, a batch size of 10 and 350 training epochs. With this, the behavior shown in Figure-3 is obtained, where the behavior was similar to that performed with 2-D information. In training, there is an RMSE of 26.54 mm, while in the validation, it was 67.1 mm, in its final epoch.

2.3 Validation and Results

With the trained network, more detailed results are obtained with the validation set, by observing how the estimated path varies in each of the coordinate axes.

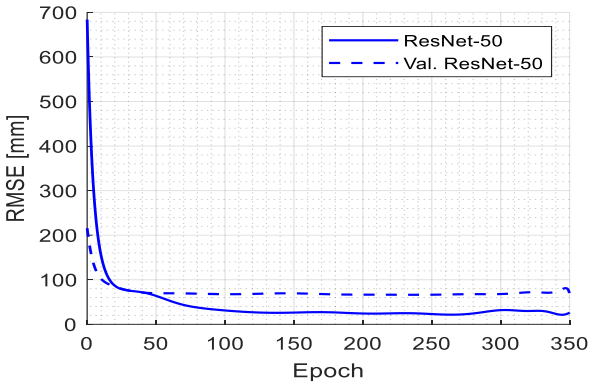

Figure-3. Training and validation behavior of the ResNet-50 with a 3-D approach.

For this, weighted accuracies of the 3 axes at each of the points are calculated, taking into account that, in order to classify a correct point, a threshold of ±20 mm of separation with respect to the proposed path is obtained. Thus, the results shown in Table-1 are obtained. It also shows the weighted deviations in the X-axis, Y-axis and Z-axis, as well as the general RMSE, which are calculated by means of (3), where their variables are the same as those used to calculate the MSE, but adding the variable j, which represents the number of the current point of the path, and the constant M, that is the number of points to evaluate, where it takes the value of 10 points when each axis is analyzed individually, and 30 to obtain the RMSE of the network.

It is observed that the more the path moves through the points on the X and Y axes, its accuracy decreases, since the estimated path may reach other points of location of the rice and not specifically that of the proposed path. In contrast to accuracy, the weighted RMSE was below the established threshold. On the other hand, the points on the Z-axis obtained 100% accuracy and obtained the lowest RMSE, of 1.26 mm.

$$RMSE = \sqrt{\sum_{j=1}^{M}\sum_{i=1}^{N} \frac{(y_{d_{ij}} - y_{ij})^2}{N\,M}} \quad (3)$$

Table-1. Accuracy of each point and average network RMSE, and overall network performance.

Point	X-axis	Y-axis	Z-axis
1	100%	100%	100%
2	100%	100%	100%
3	97.22 %	97.22 %	100%
4	91.67%	94.44 %	100%
5	88.89%	83.33%	100%
6	77.78%	80.56%	100%
7	77.78%	77.78%	100%
8	77.78%	77.78%	100%
9	77.78%	77.78%	100%
10	77.78%	72.22%	100%
Average Acc.	86.67%	86.11%	100%
RMSE [mm]	13.94	16.63	1.26
Network RMSE [mm]		12.55	
Network Accuracy		90.93%	

This is possible since, in general terms, the descent of the oath is very similar between all the proposed paths, with slight variations in the location of the points. In general, the network obtained a total RMSE below the threshold, of approximately 12 mm, and high accuracy, with a value of 90.93%, in comparison with the 2-D approach presented in [31].

It is also important to verify the deviation of each of the points in the validation set, therefore, a box plot of the 10 points on each axis is made for all the images, to visualize this distribution. With respect to the X-axis (Figure-4), the median of all points is less than 3 mm, even the range (interception between quartile 2 and 3, which make up the box) and more than half of quartiles 1 and 4 are within the threshold of ± 20 mm. However, in points 9 and 10, the minimum and maximum limits, and in points 6 and 8 the maximum limit, are outside the threshold. On the other hand, point 7 is the one with the most outliers in total, with 10, and outside the threshold, with 8 outliers. For this reason, the accuracy of these 5 points was the same. It is also possible to see 2 patterns very far from point 0, possibly due to errors in the path, having deviations up to approximately 60 mm, and a separation of the final path point of 46.66 mm, which is the upper outlier. This estimated path can be observed in Figure-7c.

Regarding the distribution of errors on the Y-axis (Figure-5), the median in most points obtained a value of less than 1 mm, only point 10 obtained a value of 3.29 mm. The range and most limits were kept within the variation threshold, with the exception of the maximum limits from point 7 to 10. On the other hand, although in this axis the distribution of 25% of the estimated points that are between quartile 2 and the median was very close to point 0, the path with greater dispersion is found this axis, whose farthest point is more than 90 mm away from the proposed path, whose estimated path can be seen in Figure-7d. Similarly, the second most dispersed values, which are also found at the bottom, are those belonging to Figure-7c.

In the distribution on the Z-axis (Figure-6), the outliers were kept at reduced values, even less than 10 mm and with medians smaller than 1 mm. It is notable to highlight the distribution at points 6 to 10, where the maximum value of the outlier was 2.12 mm at point 10, which is important in terms of the height of the end effector, since, having a small standard distribution, it prevents the final effector from colliding against the surface of the plate. For this, it was contemplated to keep the effector at a minimum distance of 3.5 mm from the surface in the proposed paths, in order to mitigate possible collisions due to the variations that the estimation of these points could have.

In the same way, it is relevant to observe how the estimated paths are compared to that proposed (ground-truth) in 3-D, in order to verify and better understand the distributions obtained. Among the estimations obtained, 4 types of paths were classified.

-The first type refers to the one that closely follows the proposed path, even ending at a point close to it. This type of estimated path is shown in Figure-7a.

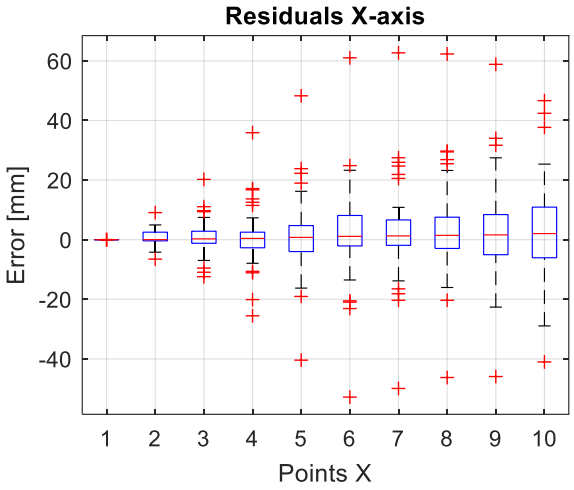

Figure-4. Box plot for X-axis points.

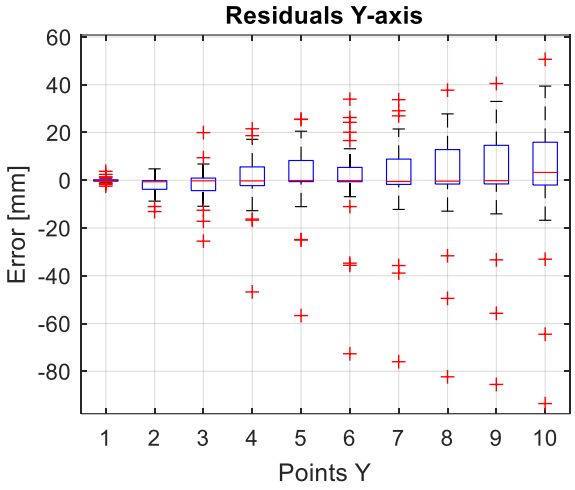

Figure-5. Box plot for Y-axis points.

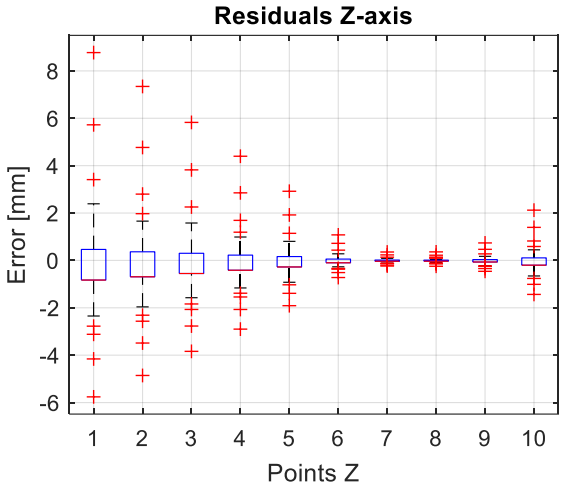

Figure-6. Box plot for Z-axis points.

- The second is that estimated path that performs the task correctly, although it does not follow the planned path manually done, i.e. it arrives at the rice and collects it, taking into account the notion of movement that from point 6 to 10 are for collection, as can be seen in Figure-7b.

- The third type of estimation is described as one that, because two or more possible collection areas were found, reaches one of those areas (not initially contemplated) and executes the collection. This type can be seen in Figure-7c, where there are 2 large groups of rice, and that unlike the ground-truth, the neural network estimates the collection path to the second group.

- The third refers to a wrong estimate, where the collection is done in a place where there is no food. This is presented in Figure-7d.

2.4 Virtual Environment

To observe the operation of the network in conjunction with the manipulator, a virtual environment developed with the V-REP software is implemented. This contains a 4-DoF robot, which uses as an end effector a tool for fastening cutlery, and in this, a spoon, as shown in Figure-8a, where each link is denoted with a color.

Likewise, the furthest point of the final effector is calculated with respect to the tip of the spoon. The robot measures 129.6 mm in link 1 (from base to joint 2), 111.5 mm in link 2 (from joint 2 to joint 3), 111 mm in link 3 (from joint 3 to joint 4), and 195.3 mm (from joint 4 to the robot reference point). Based on this, its initial position to execute the estimated path can be seen in Figure-8. Additionally, a camera (object 1) is used to acquire the image of the food located on the table (object 2) and send it to the algorithm in order to be entered into the neural network. The camera is located at the height at which the photos were taken, 55 mm away from the table. The implemented environment is presented in Figure-8b.

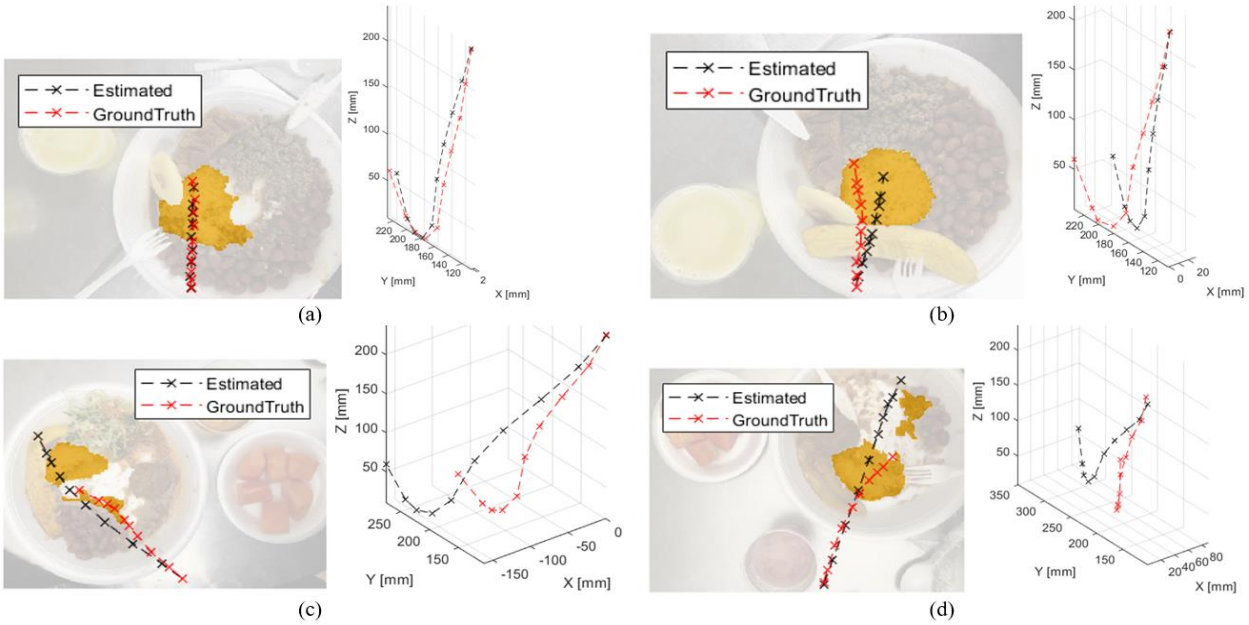

Figure-7. Example of four identified types of estimated paths.

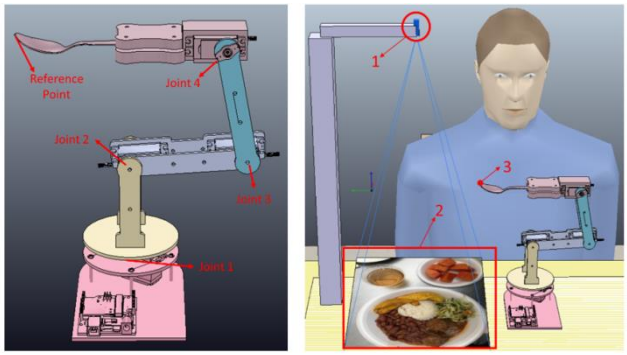

Figure-8. a) 4-DoF Robot used. b) Overall view of the environment, where (1) is the camera, (2) is the food image, and (3) is the reference point of the end effector.

3. RESULTS AND DISCUSSIONS

With the trained network and the built environment, operation tests of the network are performed within a real-time application, verifying the estimated paths with respect to the movements that the manipulator must execute. As can be seen in Figure-9, the path made by the robot in the 3-D space is drawn with a green line. In that example, it starts with the robot completely straight, going to the initial point of collection or positioning. Once there, the manipulator starts to follow the path estimated by CNN.

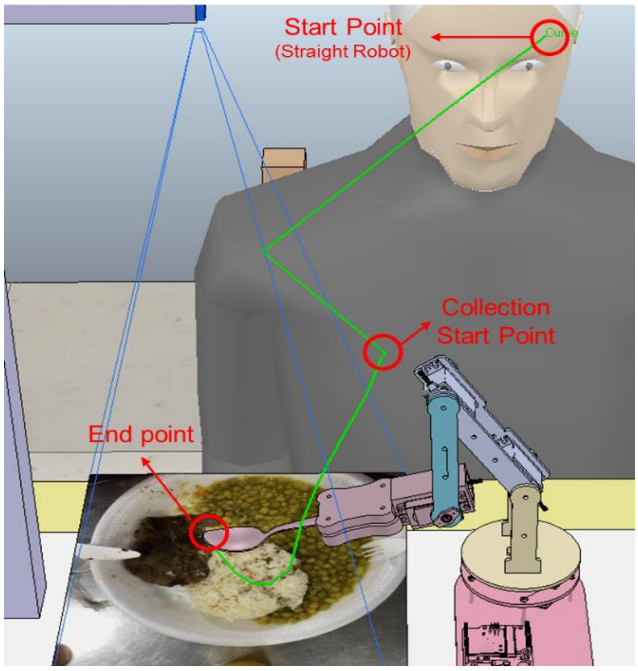

Figure-9. Global view of the estimated path performed by the robotic manipulator.

The process of the food collection algorithm is visualized in Figure-10, where first, the image of the dish is taken, and then sent to the CNN responsible for carrying out the semantic segmentation of it. Once the segmented image is obtained, the mask belonging to the rice is separated, which is converted to an image in grayscale format and introduced into the CNN regression, within which the existence or not of rice is verified. In the case that does not exist, the CNN will give as output, an arrangement of points located at or near the starting point, telling the robot to remain still. In the opposite case, CNN will give the estimated path in the direction of rice. The points given by the network are sent to an inverse kinematics algorithm of the manipulator robot to obtain the angles of each joint in order to perform the movement. Finally, the manipulator, with the angles calculated for each path, performs the collection task.

This algorithm is applied in the virtual environment implemented for different food dishes, in order to observe the path and the movement made. In Figure-11, an example of the path followed by the robot for a dish containing rice is displayed, in addition to the segmentation obtained and the graph of the path, which is identical to the one that the robot performed. Additionally, it can be observed, in the box of the camera titled Vision_RGB, the positioning of the final effector on the location of the rice. Another example can be seen in Figure-12, but in this case, the robot keeps fixed on the starting point, because there is no relevant amount of rice on the dish. This is because CNN identified this case and only sent the points with values close to the starting point.

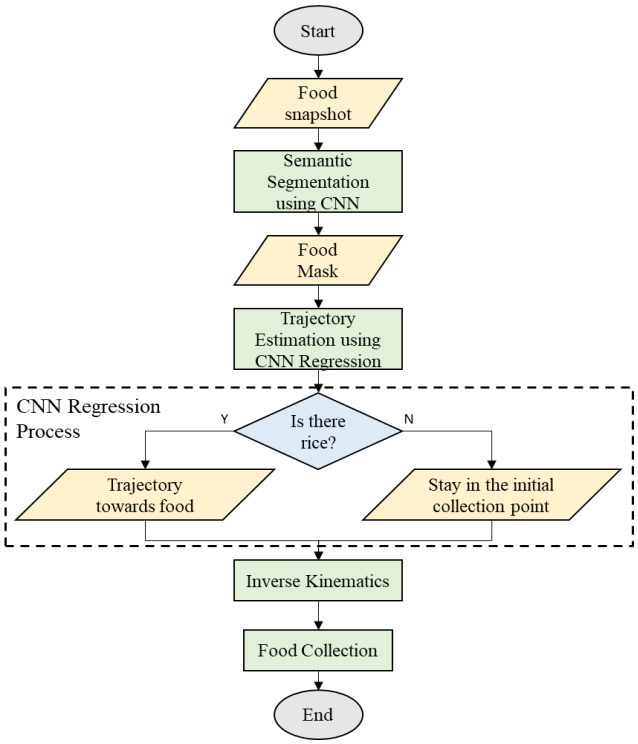

Figure-10. Flowchart of the algorithm implemented for food collection.

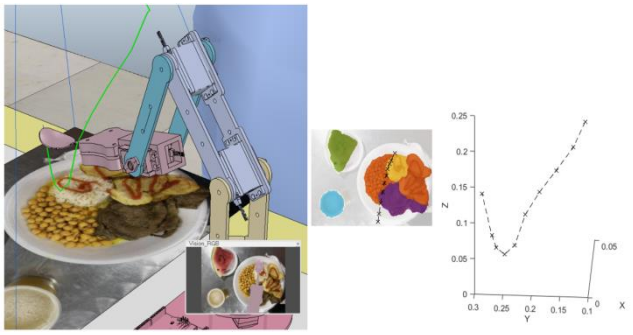

Figure-11. Example of the robot performing the estimated path.

Figure-12. Example of a static path due to the lack of a great amount of rice.

To observe in more detail the final collection path, the last 4 points of it are observed. The example in Figure-13 shows the path followed by the robot in the virtual environment and its respective delineation. In this, the points of analysis that are demarcated from 1 to 5 are indicated, being 1 when it begins to enter the food and 5 when it finally leaves. Figure-14 shows the movements of the end effector at these points, where in Figure 14a, the effector enters at an angle inclined at point 1 in order to be able to enter the food. Then, the angle is reduced by going from point 1 to 2 to collect part of the food in that sector, as shown in its final position at point 2 in Figure-14b. Following this, a sweep is made from position 2 to position 3 to have a greater amount of food in the spoon (Figure-14c), to finally ascent diagonally from 3 to 4 to accommodate what was collected and leave the food area reaching point 5 (Figure-14d). As it is possible to observe, from point 3, the ideal is not to keep the end effector straight, since due to the shape of the spoon, the food would have a tendency to move backward and fall. For this reason, it is set at an angle of 15º to keep the concavity of the spoon straight and that it can better contain the food.

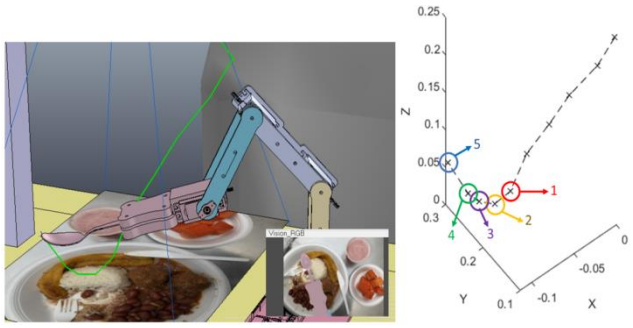

Figure-13. Path performed in the virtual environment and plotted, giving the analysis points.

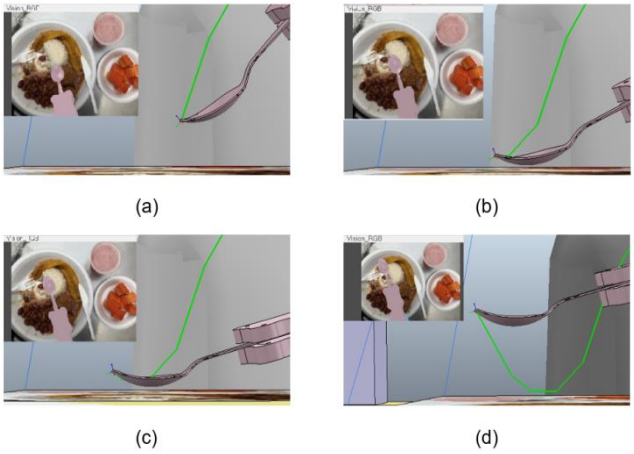

Figure-14. Process of collection using the estimated path.

4. CONCLUSIONS

This work demonstrated the capability of convolutional neural networks to solve regression problems based on the estimation of 3-D paths applied to robotic arms. It was found that, although the input was a grayscale mask-like image, the complexity of generating paths or estimating a considerable amount of data, in this case 30 values, may generate infinite correct solutions. This was demonstrated in the examples given in the experimental results section, where the network estimated paths different from those proposed that gave a logical solution, since it performed the task correctly, specifically in the examples in Figures-11b and-11c.

The network proved to generate successful paths that, although they could go beyond the error threshold, correctly performed the estimate towards an appropriate collection area, even identifying when there was no rice on the dish, managing to estimate the points near the starting point, preventing the robot from having an unnecessary displacement, achieving a degree of accuracy of 90.93% in path estimation.

On the other hand, the box plot allowed us to observe that very few trajectories were estimated with high deviation ranges. However, it was evident that some deviations were presented because the network focused its estimation on other possible collection areas other than the areas defined for the proposed path.

In the same way, the smoothness of the estimated paths was demonstrated when they were applied in a

simulated environment, where the manipulator was able to perform them without problems or collisions against the table, even keeping a certain distance over the possible height of the food dish.

This work can be applied to robotic assistants focused on feeding people, in such a way that, based on the image of the food plate and prior knowing the initial position of the robot, a path for collecting the food that the user wants is estimated. As future work, it is expected to strengthen the path estimation system, adding three-dimensional information of the environment, so that the network estimates a path focused on the section where there is more volume of food and, in the same way, verify the depth with which the final effector enters the volume of food.

ACKNOWLEDGMENTS

The authors are grateful to the Universidad Militar Nueva Granada, which, through its Vice Rectory for researchs, finances the present project with code IMP-ING-2935 (2019-2020) and titled "Prototipo robótico flexible para asistencia alimentaria", from which the present work is derived.

REFERENCES

[1] Gillen Sonja, *et al*. 2014. Solo-surgical laparoscopic cholecystectomy with a joystick-guided camera device: a case–control study. Surgical endoscopy. 28(1): 164-170.

[2] Tsarouchi Panagiota, *et al*. 2016. High level robot programming using body and hand gestures. Procedia CIRP. 55: 1-5.

[3] Roy Rinku; Mahadevappa M.; Kumar C. S. 2016. Trajectory path planning of EEG controlled robotic arm using GA. Procedia Computer Science. 84: 147-151.

[4] S.M. LaValle. 2006. Planning Algorithms, Cambridge University Press, Cambridge, UK.

[5] Berntorp Karl. 2017. Path planning and integrated collision avoidance for autonomous vehicles. En 2017 American Control Conference (ACC). IEEE. pp. 4023-4028.

[6] Islam Md Jahidul; Hong, Jungseok; Sattar, Junaed. Person following by autonomous robots: A categorical overview. arXiv preprint arXiv:1803.08202, 2018.

[7] Gasparetto Alessandro, *et al*. 2015. Path planning and trajectory planning algorithms: A general overview. En Motion and operation planning of robotic systems. Springer, Cham. pp. 3-27.

[8] Patle B. K., *et al*. 2019. A review: On path planning strategies for navigation of mobile robot. Defence Technology.

[9] Zhang Han-ye; LIN Wei-ming; Chen Ai-xia. 2018. Path Planning for the Mobile Robot: A Review. Symmetry. 10(10): 450.

[10] JI Jie, *et al*. 2016. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. IEEE Transactions on Vehicular Technology. 66(2): 952-964.

[11] Sahoo Biswajit; Parhi Dayal R.; Priyadarshi B. 2018. K. Analysis of path planning of humanoid robots using neural network methods and study of possible use of other AI techniques. Emerging trends in Engineering, Science and Manufacturing,(ETESM-2018), IGIT, Sarang, India.

[12] Ben Amarat, Samia; Zong Peng. 2019. 3D path planning, routing algorithms and routing protocols for unmanned air vehicles: a review. Aircraft Engineering and Aerospace Technology.

[13] Chen Yong-bo, *et al*. 2016. UAV path planning using artificial potential field method updated by optimal control theory. International Journal of Systems Science. 47(6): 1407-1420.

[14] Yang Qin; Yoo Sang-Jo. 2018. Optimal UAV path planning: Sensing data acquisition over IoT sensor networks using multi-objective bio-inspired algorithms. IEEE Access. 6: 13671-13684.

[15] Benavides Julián Esteban Herrera, *et al.* 2018. Flood Fill Algorithm Dividing Matrices for Robotic Path Planning. International Journal of Applied Engineering Research. 13(11): 8862-8870.

[16] Dai Yong; Zhao Ming. 2016. Manipulator path-planning avoiding obstacle based on screw theory and ant colony algorithm. Journal of Computational and Theoretical Nanoscience. 13(1): 922-927.

[17] Ghandi Somayé; Masehian Ellips. 2015. Review and taxonomies of assembly and disassembly path planning problems and approaches. Computer-Aided Design. 67: 58-86.

[18] Zhang Qiang; Zhao Ming-Yong. 2016. Minimum time path planning of robotic manipulator in drilling/spot welding tasks. Journal of Computational Design and Engineering. 3(2): 132-139.

[19] Ogbemhe John; Mpofu Khumbulani. 2015. Towards achieving a fully intelligent robotic arc welding: a review. Industrial Robot: An International Journal. 42(5): 475-484.

[20] Lecun Yann; Bengio Yoshua; Hinton Geoffrey. 2015. Deep learning. Nature. 521(7553): 436.

[21] Sarikaya Duygu; Corso Jason J.; Guru, Khurshid A. 2017. Detection and localization of robotic tools in robot-assisted surgery videos using deep neural networks for region proposal and detection. IEEE transactions on medical imaging. 36(7): 1542-1549.

[22] Nagi, Jawad, *et al*. 2011. Max-pooling convolutional neural networks for vision-based hand gesture recognition. En 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA). IEEE. pp. 342-347.

[23] Chen Bao Xin; Sahdev Raghavender; Tsotsos. 2017. John K. Integrating stereo vision with a CNN tracker for a person-following robot. En International Conference on Computer Vision Systems. Springer, Cham. pp. 300-313.

[24] Atsuzawa Keisuke, *et al.* 2019. Robot Navigation in Outdoor Environments using Odometry and Convolutional Neural Network. In IEEJ International Workshop on Sensing, Actuation, Motion Control, and Optimization (SAMCON).

[25] Tamar Aviv, Yi Wu, Garrett Thomas. 2016. Sergey Levine, and Pieter Abbeel. Value iteration networks. En Advances in Neural Information Processing Systems. pp. 2154-2162.

[26] Lee Lisa, *et al.* 2018. Gated path planning networks. arXiv preprint arXiv:1806.06408.

[27] Zhang Jiang; Xia Yuanqing; Shen Ganghui. 2019. A Novel Learning-based Global Path Planning Algorithm for Planetary Rovers. Neurocomputing.

[28] Levine Sergey, *et al*. 2016. End-to-end training of deep visuomotor policies. The Journal of Machine Learning Research. 17(1): 1334-1373.

[29] Chebotar Yevgen, *et al*. 2017. Path integral guided policy search. En 2017 IEEE international conference on robotics and automation (ICRA). IEEE. pp. 3381-3388.

[30] Pinzón-arenas J. O., Jiménez-Moreno R., Pachón-suescún C. G. ResSeg: Residual Encoder-Decoder Convolutional Neural Network for Food Segmentation. International Journal of Electrical and Computer Engineering, accepted for publication.

[31] Pinzón-arenas J. O., Jiménez-Moreno R., Rubiano A. 2019. Comparative approach of CNN regression architectures for robotic manipulator 2D trajectory estimation. In Multimedia Conference 2019.

[32] HE Kaiming, *et al.* 2016. Deep residual learning for image recognition. En Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770-778.

[33] HE Kaiming, *et al*. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. En Proceedings of the IEEE international conference on computer vision. pp. 1026-1034.