International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 16, No. 1 (Aug-24)

http://dx.doi.org/10.12785/ijcds/160178

Trajectory Tracking of a Robot Arm Using Image Sequences

Aicha Belalia[1], **Samira Chouraqui** [2] **and M'hammed Boussir** [3]

[1,2]*Computer Science Department, USTO, Oran, Algeria*
[3]*Department of Physics, USTO, Oran, Algeria*

Received 31 Mar. 2024, Revised 18 May 2024, Accepted 18 May 2024, Published 16 Aug. 2024

Abstract: In this research paper, in order to build trajectories, we used a robotic arm in computer vision and robotics. The combination of two methods, the first of which is used in this work, is a deep learning method based on convolutional neural networks (CNN); the second is Spline3, which is utilized to achieve accurate trajectory tracking. The CNN is employed to track a sequence of images acquired by the robotic arm while moving in a 2D plane. The CNN correctly locates and identifies the object by examining the visual data. Once the object is detected and located by the CNN, the outcome information is saved in a txt file. The next step is to generate a trajectory using the Spline3 method and the created txt file. This trajectory has the property of minimizing oscillations and irregularities, which ensures accurate path generation. Simulations are performed using a two-degree-of-freedom model of the SCARA arm in order to assess the efficacy of the suggested technique. These simulations demonstrate the relationship between accurate object localization by the CNN and trajectory tracking precision by the robotic arm. The metrics used for the evaluation of the proposed method include mean average precision (mAP), recall, precision, cosine similarity, mean squared error (MSE), and peak signal-to-noise ratio (PSNR). The metrics provide quantitative values of object detection accuracy by CNN and path generation similarity by Spline3.
The main aim of this study is to enable the use of this type of manipulator arm in the most complex areas, for example, to help surgeons carry out their surgical operations in an accurate and reliable manner.

Keywords: CNN, Object detection, SCARA, Spline3, Trajectory

Abbriviation Liste

DL: Deep Learning.
mAP: mean Average Precision.
TP: True Positive.
FP: False Positive.
TN: True Negative.
FN: False Negative.
MSE: Mean Squared Error
PSNR: Peak Signal-to-Noise Ratio.
AI: Artificial Intelligence.
R-CNN: Region-based Convolutional Neural Network
YOLO: You Only Look Once.
IoU: Intersection on Union distance.
DH: Denavit and Hetenberg.
2D.O.F : two degrees of freedom.
CNN: Convolutional Neural Network.
SCARA: Selective Compliance Assembly Robot Arm.
IGM: Inverse Geometric Model.
DGM: Direct Geometric Model.
OD: Object Detection.
RR:Revolute join, Revolute join.

1. Introduction

Robotics and trajectory generation are key areas of interest for researchers in the field of robotics. In robotic systems, generating optimal trajectories is crucial for tasks such as autonomous navigation and manipulation. Trajectory generation involves computing the path that a robot should follow to achieve its objectives efficiently and safely. Trajectory planning involves determining a sequence of successive joint angles over time to move a robot from an initial configuration to a final configuration, enabling the completion of a specific task [1]. In these situations, the robots are supposed to move quickly and smoothly, with little residual vibration and little position overshoot. On the other hand, trajectory design typically presents a conflict between reducing execution time and preserving motion smoothness. [2]. Trajectory generation is the process of computing a dynamically feasible state and control signal that optimizes mission objectives while adhering to specified constraints. This problem is often nonconvex, making it challenging to find effective and reliable solutions, particularly in autonomous vehicles [3].
A crucial component of trajectory planning involves object detection and tracking, enabling the robot to engage with its surroundings through the identification and manipulation of objects.
Object detection (OD) is used to locate objects in a specific area in a specific image, which is called object localization technology [4]. OD can produce valuable information for

E-mail address: aicha.belalia@univ-usto.dz, samirachouraqui@univ-usto.dz, boussirmhammed@gmail.com

the linguistic interpretation of pictures and videos. It can be applied to a variety of tasks, including autonomous driving, image classification, facial recognition, and human behavior analysis [5]. In recent years, deep learning DL technology, especially convolutional neural networks (CNN), has shown great potential for improving the accuracy and robustness of target detection algorithms.

Object detection and tracking involve analyzing images or video frames captured by cameras mounted on robotic manipulators. Traditional methods often rely on handcrafted features and classifiers, which require a significant amount of domain expertise and may not generalize well across different environments and object types. In contrast, deep learning approaches leverage large amounts of labeled data to automatically learn relevant features and classifiers.

In this work, the neural solution has been used to plan the trajectory of a SCARA manipulator arm. The robot arm is equipped with a camera to take a series of images; these images were processed by CNN in order to locate the object concerned with the tracking. The results of the localization were stored in a file. An algorithm for creating the trajectory of a SCARA manipulator arm used this file. In this step, the spline interpolation method was also applied in order to obtain a better result.

The object detection and tracking by CNN used in this work are justified by statistical measurements such as mAP, TP, FP,TN, FN,F1 SCORE, RECALL, and also the loss value (Loss), the average loss (Avg Loss), the mean squared error (MSE) and the peak signal-to-noise ratio (PSNR), which play a very important role in the evaluation of our work.

The remainder of the paper is organized as follows: the next subsections A and B review previous research related to the problem. Section 2 details the approach used, covering data pre-processing, the CNN algorithm for object detection, and performance metrics for evaluating the algorithm (subsection A). Trajectory generation for the SCARA arm manipulator is discussed in Section 2, Subsection B. Section 3 presents the results obtained and their evaluation. Finally, Section 4 summarizes the conclusions drawn from this study.

Numerous studies within the field of artificial intelligence (AI) have focused on trajectory planning and object detection. The subsequent section will delve into the related work, discussing several investigations conducted in these domains.

A. The Generation of Trajectories

Two main benefits of articulations for roboticists and other industrial developers are that they can be built more quickly than the previous generation, which reduces development time, and they move more like their real-world counterparts [6]. The optimal trajectory generation method proposed by [3] is based on the spline cubic interpolation approach. The best compatible approach to generating a smooth trajectory that passes through the points (x, y) is to use the cubic spline (spline 3).

Cubic spline (spline 3) is a commonly used interpolation method to generate smooth curves from a set of control points. This method adjusts a series of cubic polynomials between control points to create a smooth, continuous curve. The cubic spline is well suited to represent regular and continuous trajectories for the SCARA arm from discrete control points [7], [8]. In [9], to reduce vibration of the serial manipulator and increase robot efficiency, an efficient algorithm (IWOA-PSO) was established for optimal time-jerk path planning. In order to avoid settling into the local optimum solution, WOA was enhanced by utilizing adaptive weight and threshold to balance WOA exploration and exploitation. Next, to speed up the convergence of IWOA, the PSO method and IWOA techniques were combined. The results showed that the suggested IWOA-PSO successfully lessened the robot's jerk while increasing its productivity.

In their paper [10], a model-based methodology was provided for trajectory generation for a flexible-joint space manipulator grasping a massive object. Methodologies were provided to generate a feed-forward motor trajectory given a desired link-side trajectory and to generate a desired link-side trajectory while minimizing total trajectory time subject to dynamic constraints. Full-spatial simulation results show the efficacy of this trajectory generation method. Zhao and all developed a lightweight GDTP model to map the initial/final states and the control action sequence based on generative adversarial networks (GANs). This lifelong learning framework was proposed to achieve effective and high-accuracy direct trajectory planning (DTP) tasks. Several scenarios with different characteristics were used to test the suggested method. The experimental findings demonstrated that the method could quickly plan very feasible trajectories. They found that the method's tracking errors were 29.1% [11].

According to [12], they provided an innovative way to quickly generate collision-free ideal trajectories for multiple non-holonomic mobile robots in highly obstacle-filled environments. The process began with a discrete solution found by a graph-based multi-agent path planner and was refined into smooth trajectories through nonlinear optimization. In order to increase the algorithm's scalability, they divided the robot team into smaller groups and recommended a prioritized trajectory optimization strategy. Under some conditions, the decoupled optimization framework could result in unsolvable sub-problems. The superiority and efficacy of the suggested method were confirmed by hardware experiments and simulations.

In his research, [13] explored vibration control and trajectory tracking for rotary flexible joint manipulators with parametric uncertainties. The study began with a discussion on the dynamic modeling of a single-link flexible joint manipulator using the Euler-Lagrange equation. Following this, a continuous and bounded jerk trajectory was designed and implemented to achieve smooth motion between two points. The robot and obstacles were represented by polyhedral volumes within the workspace. The configuration space was constructed by calculating the joint angles in contact with these polyhedral volumes. The non-obstacle regions in the configuration space represented the permissible areas for robot movement.

B. *Object Detection by CNN*

In deep learning techniques, algorithms analyze data to detect relevant features and then combine them for rapid learning. To recognize objects and people based on images obtained from videos, various deep learning techniques are used [7]. Among deep learning methods, CNN is most suitable for image-based search, and it achieves the best recognition accuracy in various applications. Therefore, we consider CNN in this study.
CNN-based OD algorithms include:

1) *R-CNN*

IT is an object detection architecture. R-CNN first extracts regions of interest in the image and then uses these regions as input data for the CNN. This region division makes it possible to detect multiple objects from different categories in the same image.This solution, proposed by [14], improves the accuracy of the recognition model.

2) *Faster-CNN*

The most popular object detection method is faster R-CNN. It is part of R-CNN. Fast R-CNN improves upon the R-CNN family. [4] developed R-CNN, resulting in faster R-CNN. A convolutional neural network (CNN) receives the input image to obtain a property map of the objects present in the image. Backbone is a Region Proposal Network (RPN), which then uses this feature map and anchors to generate region proposals. These regions are then filtered using NMS (non-maximal deletion). By combining the character maps and bounding boxes of connected objects that the CNN extracted, new feature maps are created by pooling regions of interest (ROIs). After that, a fully connected layer processes the grouped areas to forecast output classes and object region coordinates. The header network is the name of this component of the Faster R-CNN design.

3) *YOLO*

One of the DNN-assembled innovations based on excellent speed and accuracy execution is YOLO. YOLOv3 completes related execution much more quickly than other identifiers [15]. YOLO, or ”You Only Look Once,” is a deep neural network that processes an image just once; this is in contrast to ”region proposal methods,” which are primarily employed by R-CNN-based models. There are nine iterations of YOLO (YOLOV1,... YOLOV9) [16].

2. Research Methodology

Our methodology is divided into two main parts: the first is the object detection of the image sequences acquired by the arm using the deep learning method CNN. The second part is the trajectory generation using the output of the CNN in the first part and the spline method.
Our method can therefore be represented by the following steps:

1) Acquisition of images by the SACRA robotic arm.
2) Processing these images involves resizing them and then running the object detection algorithm using a convolutional neural network (CNN) for all images.
3) Extracting coordinates (x, y) and saving them in a (.txt) file.
4) Calculating maps, F1 score, and recall for each image.
5) Testing the map values for each image; if it is less than 50%, then reject; otherwise, using these coordinates (x, y).
6) Apply the DH parameters and call the spline 3 method.
7) Call the generate-Trajectory function.
8) Finally, calculate metrics (MSE, PSNR, cosine similarity).

The following flowchart (Figure 1) displays all steps.

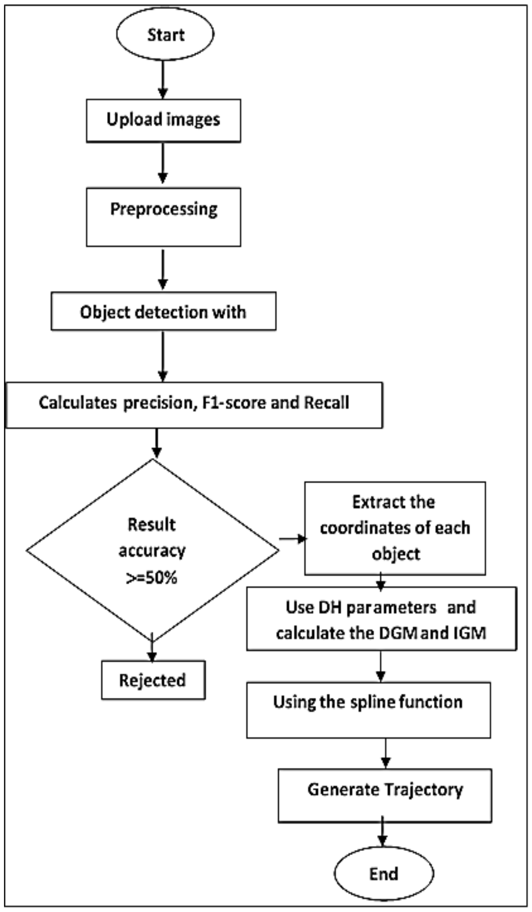
Start
Upload images
Preprocessing
Object detection with
Calculates precision, F1-score and Recall
Result accuracy >=50%
Extract the coordinates of each object
Use DH parameters and calculate the DGM and IGM
Rejected
Using the spline function
Generate Trajectory
End

Figure 1. Flowchart of the used method

A. *Object Detection Using CNN*

There are 172 images in our database used to track the first object and 180 images for the second object.
All the images were collected manually and do not belong to any specific database. Out of these, 152 images were used for training the first object detection model and 160 for the

second. The pre-trained Darknet-53 network was employed to process the 152/160 images initially. Darknet-53 first resizes each image to 416x416 pixels, in color (3 channels), to extract features. This method utilizes a technique known as anchors or bounding boxes, which frame objects within a rectangular space.
Each image will be divided first by 32, then; by 16, and finally by 8, resulting in images that have been divided into a grid of cells (13x13, 52x52, 26x26). The object's center is located in the cell that is in charge of object detection. We obtain for each image (13x13 + 52x52 + 26x26) * 3 = 10647 bounding boxes. The algorithm selected one box from this group to identify our object.
Figure 2 presents the CNN architecture used in our work, illustrating the steps involved. The size of the input and output images can vary between each step.

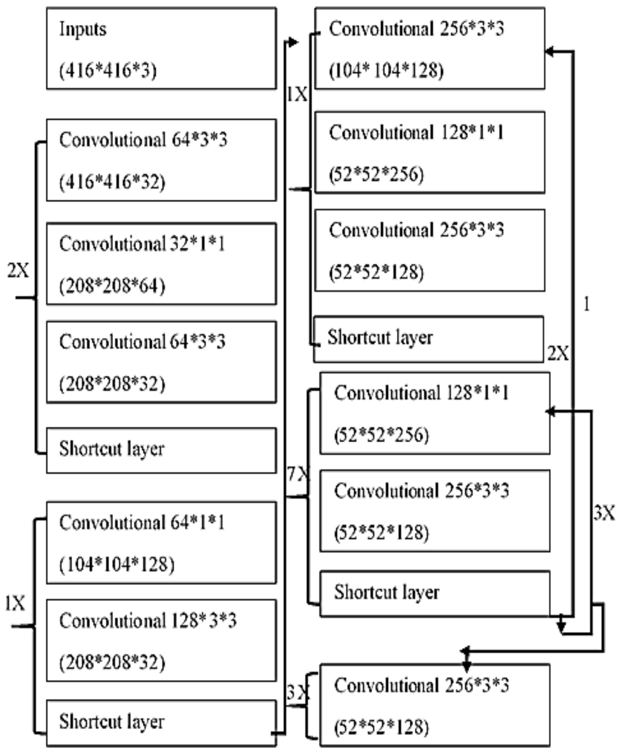

Figure 2. CNN architecture

Each image is processed by a set of layers: convolution, pooling, ReLU correction, and fully connected.

1) Convolution

Its objective is to identify whether a specific set of characteristics (or features) is present in the input images. To do this, convolutional filtering is employed. The basic concept is to "drag" a window that represents the filter over the image, and then calculate the convolutional layer by taking the convolutional product of the filter and each segment of the scanned image.

2) The ReLU Correction Layer

The real nonlinear function given by ReLU(x)=max(0,x) is represented by the symbol ReLU (Rectified Linear Units). Therefore, any negative values received as inputs are replaced with zeros by the ReLU correction layer. It serves as a mechanism for activation [17].

3) The Pooling Layer

It takes in several feature maps as input and performs the pooling procedure, which reduces the size of the image files without sacrificing quality. The image is divided into square cells, with the maximum value preserved within each cell. In practice, small square cells are often used to minimize information loss. The most common choices are 2××2 pixels that are near each other but do not overlap, or 3××3 pixels that are near each other; but are spaced apart by a step of 2 pixels (and so overlap).
The amount of feature maps in the output and the input is the same, however they are significantly smaller.

4) The fully-connected Layer

Is the final layer of a neural network, converting a vector into a new vector using linear combinations and activation functions. It classifies images using a vector of size N, with each element representing the probability of the input image belonging to a class.

5) Feature Extraction

We introduced the K-means algorithm to calculate the distance between the delimitation boxes estimated by the algorithm and the predefined bounding box, but we replaced the Euclidean distance it uses with the intersection over union distance (IoU)

$$dist(box, centroid) = 1 - IoU(box, centroid). \qquad (1)$$

The closer the IoU is to one, the better the results. The extraction of features from images will be saved in a file as weights. Its weights will later be used for object detection in other images.
The following list of formulas is displayed during the object detection phase.
Confusion Matrix: Four elements are required in order to create a confusion matrix.
TP: Based on the ground truth, the model accurately anticipated and matched the label.
TN: The model is not a component of the ground truth and does not forecast naming.
FP: The model predicts a label, but it does not correspond to the ground truth (Type I mistake).
FN: The label is a component of the ground truth, but the model does not forecast it. (Error type II).
Precision: A model's capacity to recognize only the pertinent data points.

$$Precesion = \frac{TP}{TP + FP} \qquad (2)$$

Recall: A model's capacity to locate every pertinent case in a given data set.

$$Recall = TP/(TP + FN) \tag{3}$$

mAP : It is calculated by finding the Average Precision (AP) for each class and then the average over several classes.

$$mAp = \frac{1}{n}\sum_{i=1}^{N} AP_i \tag{4}$$

where n is the number of classes and AP_i is the average precision of classe i
F1-score: The F1-score is defined as the harmonic average of precision and recall, which results in the following equation:

$$F1 - score = \frac{2 \times (precision \times recall)}{(precision + recall)} \tag{5}$$

The subsequent section will make use of the findings obtained in this step.

B. *Trajectory Generation Steps*

To generate the trajectory of the SCARA arm manipulator, several steps involving the arm joints must be followed.

1) *The SCARA 2 d.o.f Robot*

A SCARA robot (Figure 3) is defined in ISO standard 8373:1994, No. 3.15.6. SCARA robots use two parallel revolute joints to provide compliance in the horizontal plane against vertical loads [18]. The first prototype of the SCARA robot was built in 1978. The second one was built in 1980 [19].
In this work, we used a SCARA manipulator arm with a 2.d.o.f.; the type of both joints is a rotary joint (revolute joint) (RR), equipped with a camera, that works in a plane of 2 directions. More information on how we modeled the SCARA arm is below.

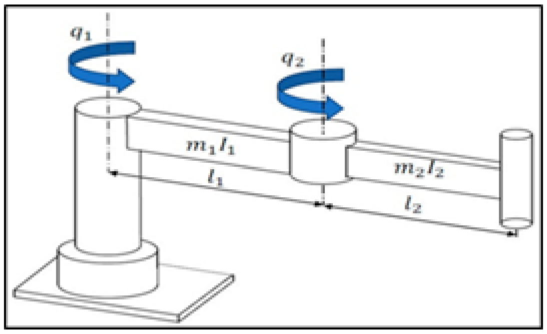

Figure 3. SCARA Arm Manipulator [20]

2) *Denavit and Hetenberg Parameter*

For the construction of the marks, it is necessary to follow the following procedure:
Name the robot fields from i=0 to $i = n$, starting at the robot base with i=0.
Name the joints from $i = 1$ to n. (1 for the first degree of freedom and n for the last).
For $i = 0$ up to $i = n - 1$, fix the Z_i axis on the articulation $i + 1$.
The origin of the coordinate system R_0 is any point on the Z_0 axis, so that X_0 and Y_0 form a direct orthonormal coordinate system.
For $i = 1.2, ..., n - 1$, the origin of the coordinate system is fixed at the intersection of the Z_i axis with the perpendicular line common to $Z_{(i-1)}$ and Z_i. If the two axes intersect, the origin is the point of intersection.
If, on the other hand, the axes are parallel, the origin is the origin of the coordinate system of the articulation $i + 1$.
The X_i axis is the perpendicular line common to $Z_{(i-1)}$ and Z_i, and the Y_i axis is chosen so that the coordinate system is directly orthonormal [21]-[22]. The following figures (Figure 4 and Figure 5) depict the structure of the SCARA arm and the SCARA arm system. The arm's structure was modeled in a 2D plane, utilizing only two axes: the x-axis (abscissa) and the y-axis (ordinate).

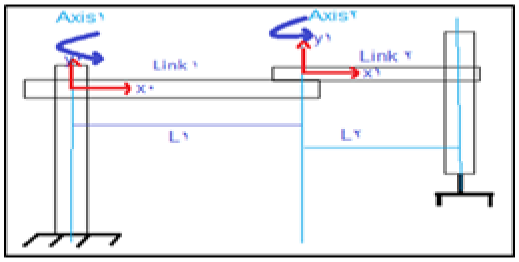

Figure 4. SCARA arm structure

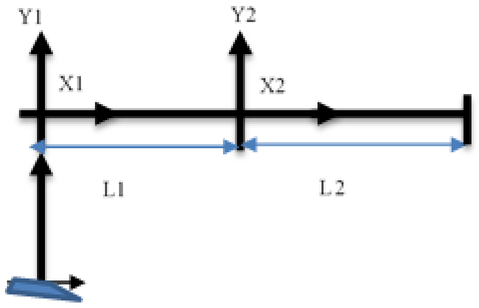

Figure 5. SCARA arm coordinates system

The DH method principle dictates that the SCARA robot's structural parameters were categorized into four groups. The torsional angle α_i angle of articulation (theta$_i$)$jointlengtha_i$, and linkage length d_i make up these groups. The following are the definitions for these parameters:

1) d_i is the translation in the positive direction Z_i between the $X_{(i-1)}$ and X_i axes.
2) The translation between the axes of $Z_{(i-1)}$ and Z_i is $a_{(i-1)}$, and it is positive along the $x_{(i-1)}$ direction.
3) The rotation of approximately $x_{(i-1)}$ counterclockwise between the axes of $z_{(i-1)}$ and z_i is denoted by $\alpha_{(i-1)}$.
4) The rotation around z_i counterclockwise between the axes of x_i and $x_{(i-1)}$ is denoted by θ_i.

3) *Khalil's Model*

In 1986, Khalil and Kleinfinger made improvements to the DH model due to ambiguities regarding robots with closed or tree structures. They also proposed a method for robots with simple open architectures. For robots with an open structure, the following convention is proposed: The coordinate system R_j is fixed with the link j. The axis of the articulation j is the axis z_j and the axis x_j is aligned with the perpendicular common to the current axes of the articulation j and following j + 1 [23]. With this convention, the equation of the transformation matrix from the coordinate system j-1 to the coordinate system j is:

$$T^j_{(j-1)} = Rmat(x_{(j-1)}, \alpha_j) \times Tras(x_{(j-1)}, \alpha_j) \times Rmat(z_j, \theta_j) \times Tras(z_j, \theta_j) \tag{6}$$

Where: The angle formed by $x_{(j-1)}$ and $x_j$about the z_j axis is denoted by θ_i.
The distance along z_j between x_j and $x_{(j-1)}$ is denoted by r_j.
The length of $x_{(j-1})$ is denoted by d_j, while the angle formed by $z_{(j-1})$ and z_j around $x_{(j-1})$ is represented by α_j.
In relation to the coordinate system $j-1$, the coordinate system transformation matrix j is as follows: [24].

$$T^j_{(i-1)} = \begin{bmatrix} C\theta_j & -S\theta_j & 0 & d_j \\ C\alpha_j S\theta_j & C\alpha_j S\theta_j & -S\alpha_j & r_j S\alpha_j \\ S\alpha_j S\theta_j & S\alpha_j S\theta_j & C\alpha_j & r_j C\alpha_j \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4) *The Two dof SCARA Arm Geometric Modelled on a 2D Plane*

The configuration of the robot is determined by the articular variables θ_1 and θ_2 (Figure 6). The DH parameters of the SCARA robot (2 d.o.f) are presented in Table 1: (With: $l_1 = 0.2$ m and $l_2 = 0.3$ m).

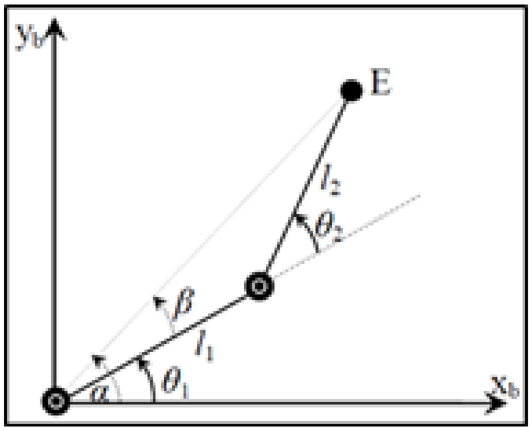

Figure 6. The SCARA arm configuration [7]

TABLE I. SCARA prm parametrs

Label	a_i	α_i	d_i	θ_i
1	L_1	0	0	θ_1
2	L_2	0	0	θ_2

To calculate the direct geometric model (DGM) of a SCARA arm in a 2D plane, we followed these steps.
In our scenario, we were focusing on a 2D plane due to the nature of convolutional neural networks (CNNs), which are primarily designed for 2D vision (images) and do not directly process depth information (z) from a single image. In this case, the translation matrix is given by:

$$T^2_0 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_2 & L_1 + L_2 * \cos\theta_2 \\ \sin\theta_1 & \cos\theta_1 & L_2 * \sin\theta_2 \\ 0 & 0 & 1 \end{bmatrix}$$

5) *Inverse Geometric Model*

The problem is to calculate the joint coordinate's θ of the robot from the operational coordinates X. The inverse geometric model is the inverse problem that makes it possible to know the articular variation according to the situation of the terminal organ, which can be represented by the relation: [25]-[22].

$$\theta = g(X) \tag{7}$$

Where X is the vector of the operational coordinates expressed in the reference frame R_0, and θ is the articular variance.
The matrix T, 0 and n represents the position and orientation, expressed in the reference frame R_0, of the robot's terminal member.
Three methods of calculating IGM can be used [25]

1) Paul's technique [26], which is appropriate for the majority of industrial robots and handles each unique instance separately.
2) Pieper's solution [27] solves the problem for six-degree-of-freedom robots with three prismatic joints or three roasted joints of consecutive axes.

3) An example of the modification is provided by [28], which modifies the Raghavan and Roth algorithm for solving inverse kinematics. The inverse kinematics of a KUKA KR 5 robot were illustrated. A better approach to solving the IK for robot manipulators with six degrees of freedom and offset wrists is suggested. This approach is predicated on the Newton iteration methodology. This algorithm requires only 4% of the comparison algorithm's mean calculation time for a single IK solution [29].

Figures 7 and 8 illustrate how the arm can move in both left and right directions.

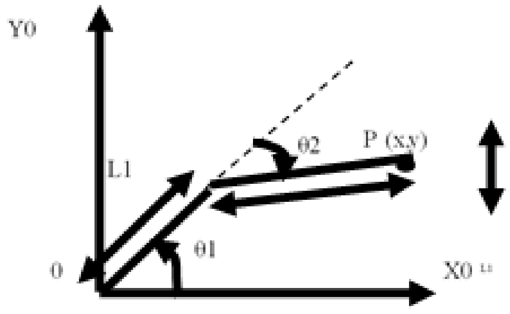

Figure 7. Arm parameters in a 2D plane, step 1

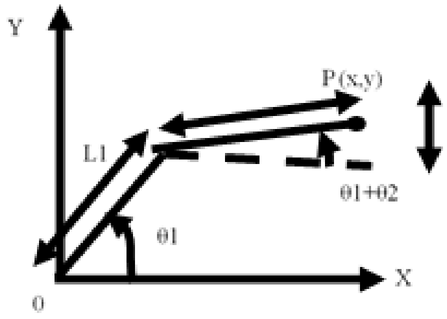

Figure 8. Arm parameters in a 2D plane, step 2

We have the following direct geometric model:

$$x = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) \tag{8}$$

$$y = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \tag{9}$$

Let us complete the scheme of the direct geometric model by using the generalized Pythagorean theorem (Figure 9). We have the following relationships:

$$L2 = x^2 + y^2 \tag{10}$$

$$L2 = l_1 2 + l_2 2 - 2l_1 l2 \cos(\alpha) \tag{11}$$

$$\theta = \pi + \theta_2 \tag{12}$$

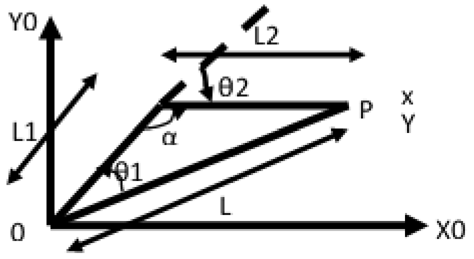

Figure 9. Pythagorean Theorem

Generalized Pythagorean theorem:

$$x^2 + y^2 = l_1 2 + l_2 2 - 2l_1 l_2 \cos(\theta_2) \tag{13}$$

we have :

$$\cos(\pi + \theta) = -\cos(\alpha) \tag{14}$$

When θ_2 is positive (resp., negative), the robot has a low elbow posture (resp., elbow high) (figure 10).

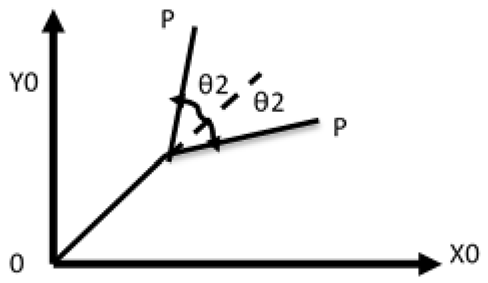

Figure 10. The calculates of $\theta_1 and \theta_2$

Let us, via the spline method, calculate that, as a result, the MGI is:

$$\theta_1 = arctg(\frac{y(L_1 + L_2 \cos(\theta_2)) - xL_2 \sin(\theta_2)}{x(L_1 + L_2 \cos(\theta_2))}) \tag{15}$$

$$\theta_2 = arcos(\frac{x^2 + y^2 + (L_1)^2 + (l_2)^2}{2L_1 L_2}) \tag{16}$$

6) The Procedure Followed and Limitations for Trajectory Generation

The coordinates for object detection obtained from the CNN were stored in (.txt) files, which were then utilized as input for the arm manipulator system to study the objects. Consequently, considering the specified constraints, we formulated the trajectory for a SCARA manipulator arm.

The constraints considered in this study are:

1) The robot's kinematic constraints, such as acceleration, speed, and maximum angular velocity, are not

considered.

2) Ignore collisions that occur on the work plane.
3) The plane in this work is divided into zones, each containing a single point with coordinates (x, y) representing the object's center of gravity detected in the image at time t_i.
4) The images were sorted based on their acquisition time, with images $01.png$ taken at time $(t_1)and02.png$ taken at time t_2, and so on.
5) There are six seconds worth of acquittals, each lasting 300 milliseconds.
6) We have obtained 20 images, or discarded those with a detection accuracy below 50%, resulting in only 3 images in our case for the first object , and there are no discarded images for the second object.
7) The matrix containing time (t_i) and coordinates (x_i, y_i) at time (t_i) is used as input for the previously created model, following articulation rules and respecting the Direct Geometric Model (MGD) and Inverse Geometric Model (MGI).
8) Joint 1 (for the abscissa): It controls the movement of the arm along the x-axis.
9) Joint 2 (for ordinates): It controls the movement of the arm along the y-axis.
10) The coordinates of the points we want to reach are given by (x, y).

For each joint, there is a function.
Equation for articulation 1:

$$\theta_1 : \theta_1(t) = f_1(t) \tag{17}$$

Equation for articulation 2:

$$\theta_1 : \theta_1(t) = f_1(t) \tag{18}$$

Consequently, we completed the following tasks in total:

1) The first step is to create a matrix to store the (x, y) coordinates of the regions.
2) Establish the duration needed to arrive at each point, which corresponds to the quantity of images used for the test.
3) The trajectory was created using the cubic-spline interpolation method.
4) This cubic spline allows us to smoothly connect the key points and generate a continuous trajectory. Additionally, by adjusting the parameters of the cubic spline, we can control the smoothness and curvature of the trajectory to match specific requirements or constraints.
5) Using the spline function, we obtain the cubic polynomials for each joint, θ_1 and θ_2. For this process, we have converted the coordinates x, y utilized for this process to meters.
6) By inputting the vector x_i, vector y_i, length L_1, length L_2, and time t into the function, we can obtain the corresponding angles θ_1 and θ_2 at that specific time. The function will return two vectors of the same length, where one vector contains the values of θ_1 and the second vector contains the values of θ_2
7) The generate-trajectory function takes the two vectors of angles θ_1 and θ_2 and combines them to create a single trajectory for the arm. This final trajectory represents the movement of the arm over time, allowing us to visualize its path and analyze its motion. Additionally, this function can be customized to include any necessary calculations or adjustments to optimize the arm's performance.

3. Results And Discussion

As previously discussed, our work is divided into two main phases. The first phase utilizes Convolutional Neural Networks (CNNs) to recognize objects in image sequences. In the second phase, we use the results from the object recognition to generate a trajectory that incorporates spline interpolation. The subsequent section will present the results of each phase, organized according to the two steps.

A. Object Detection Output

After applying the CNNs with 2000 iterations on 152 base images for the first object (and 160 for the second), the results of extracting feature maps are detailed below.

In this section, we were focused on explaining the results obtained for the first object, as the results and precision were identical for both objects. Object detection is depicted by a rectangle enclosing the object's centroid coordinates.

The detection results, represented as scores or weights, were stored in backup files. For example:
(Next (mAP) calculation at 2000 iterations)
Last accuracy ($mAP@0.50$) = 99.98 %, best = 99.98% .
Furthermore, the individual values obtained from this detection are provided as follows:
detection's count = 223, unique truth count = 196
Mean average precision ($mAP@0.50$) = 0.999767, or 99.98 %.
Therefore, among 223 detection, there are 196 true detection, which gives us a percentage of the average accuracy equal to 99.98%. Table II illustrates the matrix of positive and negative detection. The curve of evolution of mean average precision as a function of loss is given in the Figure 11.

TABLE II. CONFUSION MATRIX.

	Positive	Negative
Positive	$TP(196)$	$FP(4)$
Negative	$TN(0)$	$FN(0)$

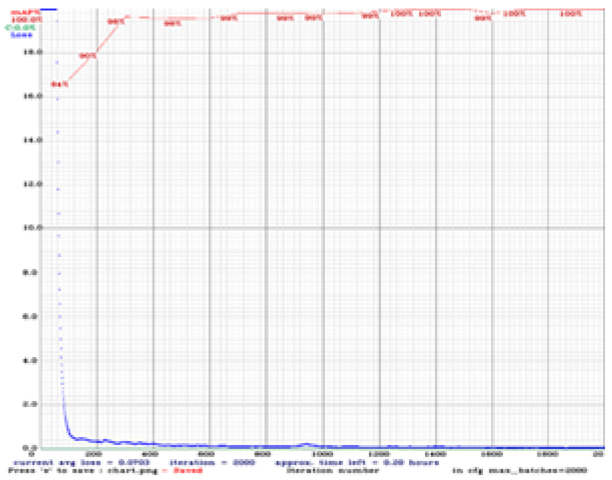

Figure 11. Average precision versus loss curve

In our experiment, the accuracy was measured at 0.98 with a recall of 1.00.

An inverse relationship was observed between the mean Average Precision (mAP) value (shown in red) and the loss (depicted in blue). As the loss value decreases, the precision value increases until it reaches its peak at iteration 800, achieving an accuracy of 98% with a loss of 0.1.

Following this, several tests were conducted using various images, including those from our database or external sources. Below is an example illustrating the results of a test performed on an external image: *../content/drive/MyDrive/yolo*20222/ *imagesballonsanstrajet*/05.*png Predictedin*715.666000 *milli − seconds*.
ballon: 98%
The image (05.png), depicted in Figure 12, exhibits a high detection percentage (98%) for a single object of type "ball" (the first tracked object).

The detection accuracy for this instance can be visually represented directly on the tested image or displayed in text form, as illustrated in the following figure

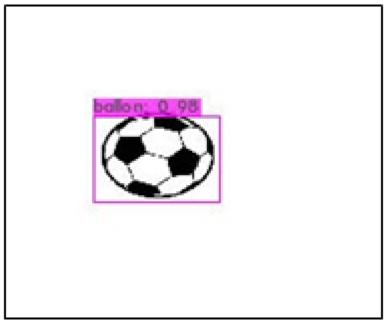

Figure 12. Object detection with CNN

Figure 13 displays another detection scenario where three ball-type objects are identified. Each object is enclosed within a rectangle, and the percentage of detection confidence is indicated on each rectangle.

The object on the left shows a detection confidence of 99%. The object in the middle and the one on the right both exhibit a detection confidence of 100%.

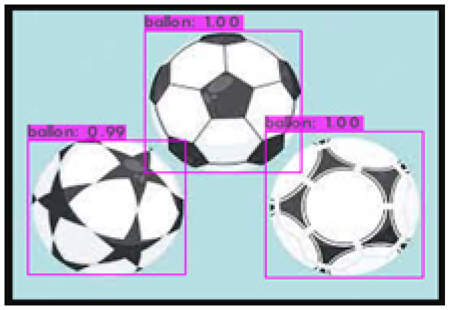

Figure 13. Detection of three objects with CNN

The subsequent figures display the results obtained by the CNNs in the form of statistical metrics.

Figure 14 illustrates the true positive and false positive detection zones. Figure 15 presents precision, recall, and F1-score metrics.

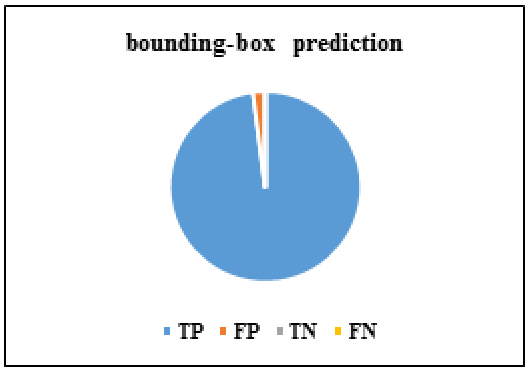

Figure 14. Bounding-box prediction

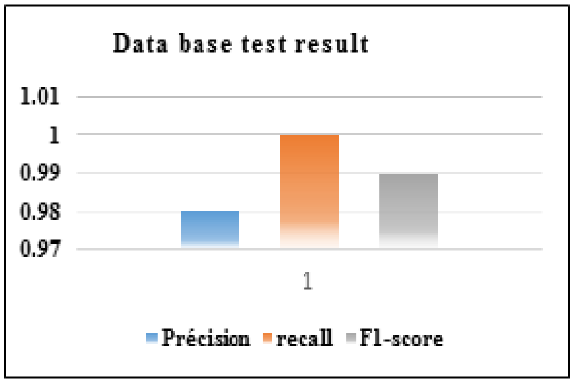

Figure 15. Data base test result

Figure 16 showcases the mean Average Precision (Map@0.50) along with the overall average performance metrics.

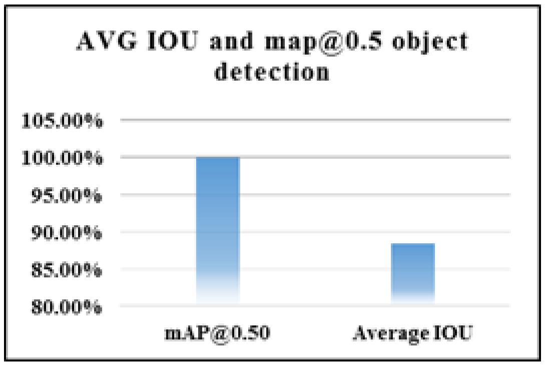

Figure 16. AVG IOU and Map@0.5 object detection

The detection and localization of the ”ball” class in the 20 images named 01.png to 20.png have resulted in saved text files (.txt). Each file contains the following information for each detected ball: image name and prediction time.
Detection confidence percentage for the ball class.
Cartesian coordinates of the centroid of the detected object, along with the width and height of the bounding box surrounding the object.
The format of the information within each file is structured as follows:
Ball: nn% (confidence percentage)
Where:
nn% represents the percentage confidence level for the ball detection (higher values close to 100 indicate more accurate detection). This detailed information provides precise localization and characterization of the detected ball objects within each image.

B. Trajectory Generation Results

The following section explains and displays the various trajectories created in our work using different comparison tools, along with our approach. The following figures (Figure 17 and Figure 18) illustrate the process of generating the trajectory in two stages using the created model.

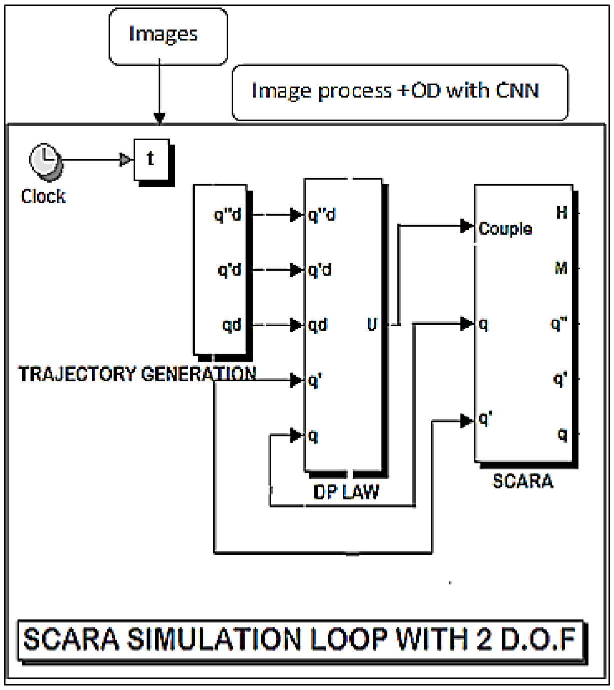

Figure 17. Extract coordinates with CNN

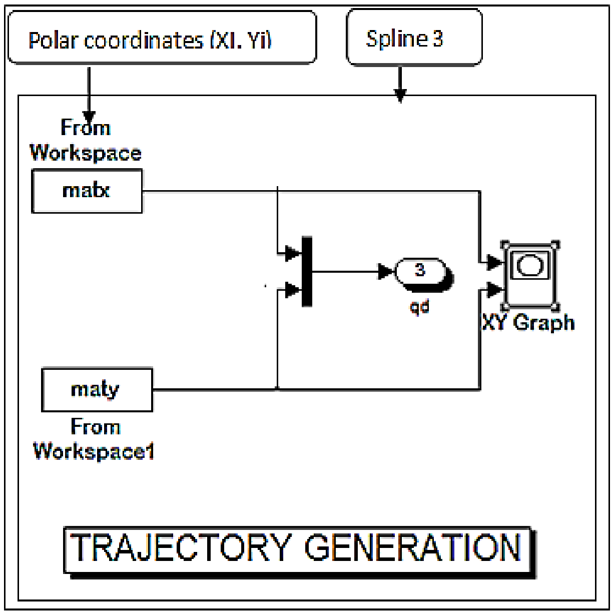

Figure 18. Trajectory generation using SCARA arm

The two generated trajectories tracked in this work are presented consecutively in Figures 19 and 20.
The two figures present the two trajectories generated by the SCARA 2.$d.o.f$ manipulator after entering the coordinates (xi, yi) (found previously) as well as the spline3 method. So, Figure 19 represents the simple trajectory, and Figure 20 presents the complex trajectory.

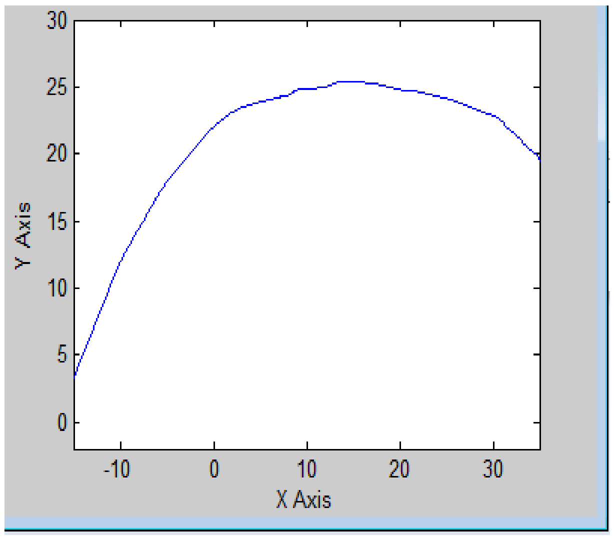

Figure 19. The first path created by the SCARA arm

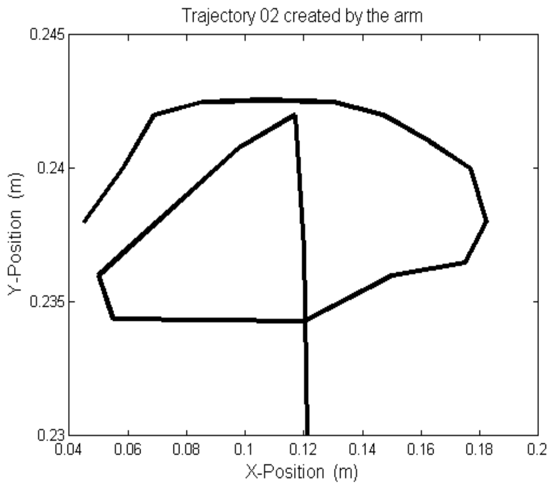

Figure 20. The second path created by the SCARA arm

Figures 21 and 22 display the real trajectories that the monitored object made while it was moving.

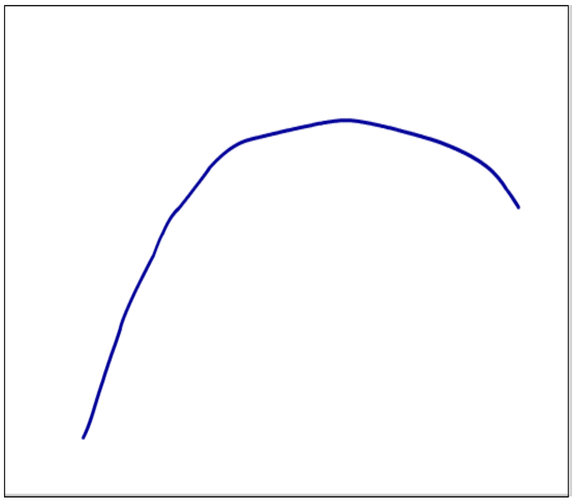
Figure 21. The first real path created by the first object

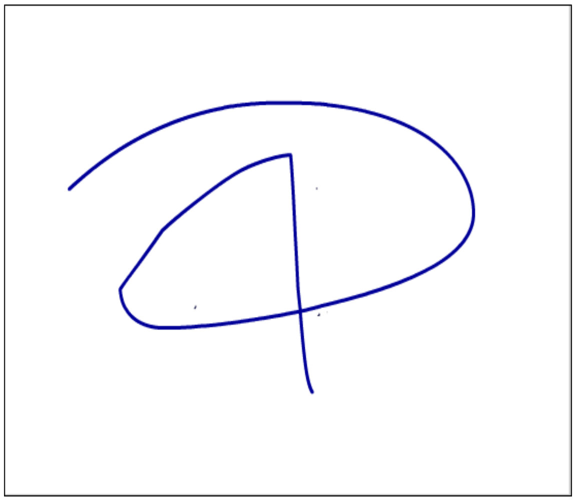
Figure 22. The second real path created by the second object

C. Evaluation

For the initial trajectory analysis, we aimed to construct the balloon's trajectory as depicted in Figure 21. We utilized two distinct tools to generate the trajectory from the same set of points mentioned above, enhancing the evaluation of our strategy. Here are the outcomes:
The first application is a robust tool designed for complex data manipulation and advanced analysis, offering detailed control over graphical customization. Figure 23 displays the trajectory obtained using this tool.
The second application is a quick and straightforward method that utilizes the provided coordinates in columns to generate a static graph. Refer to Figure 24 for the trajectory obtained using this approach.

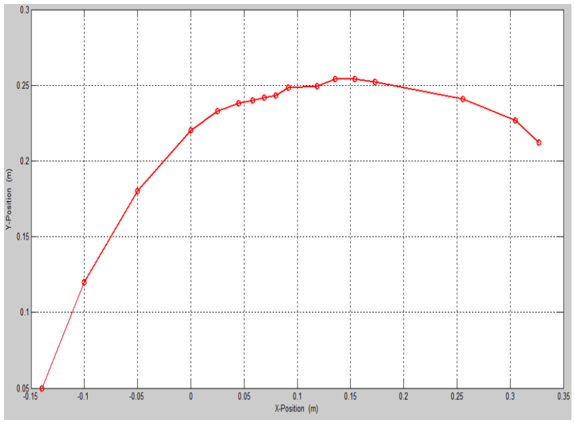

Figure 23. Path created by application 1

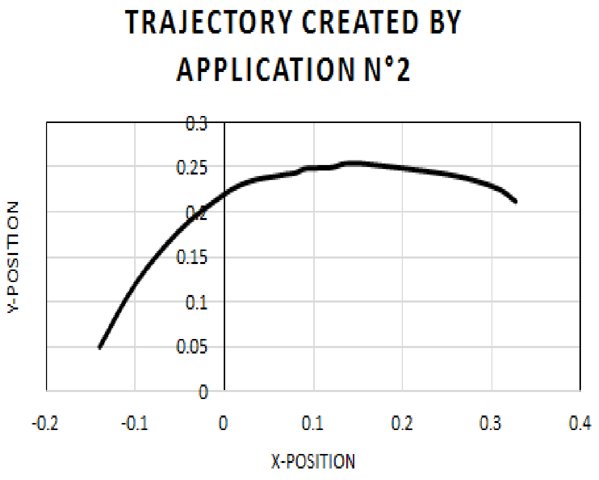

Figure 24. Path created by application 2

To further evaluate this work, we conducted comparisons with other research and calculated various metrics to assess the similarity between the trajectory generated by the SCARA arm and the actual trajectory.
Our evaluation involved the following steps and considerations:
Comparison with Existing Research: We reviewed related literature and compared our approach with similar studies in the fields of trajectory generation and robotics.

Metric Calculation: We computed metrics to quantify the similarity and accuracy of the generated trajectory compared to the actual trajectory. These metrics may include measures of trajectory error, such as mean squared error (MSE), root mean square error (RMSE), or other distance metrics.

D. Comparison

There are limits to the computational efficiency and viability of planned trajectories with current trajectory planning techniques. For example, in [11], a framework for continual learning is introduced, aimed at executing high-precision and efficient direct path planning (DTP) tasks. Leveraging Generative Adversarial Networks (GAN), the study develops a lightweight GDTP model to establish mappings between initial and final states as well as the sequence of control actions. Through experimental comparisons with the widely used Cubic Curve RRT* (CCRRT*) algorithm, the results indicate that the proposed method enables the generation of highly attainable trajectories within a short timeframe. Notably, the tracking errors of their method, in terms of both position and heading angle, are found to be 29.1% and 44.1% lower than those of CCRRT*. In the work of [30], in order to solve the problem of forecasting future vessel trajectories with a forecast horizon of several hours, they investigated deep learning strategies using historical AIS observations. Utilizing recurrent encoder-decoder neural networks (RNNs) trained on historical trajectory data, they present novel sequence-to-sequence vessel trajectory prediction models that forecast future trajectory samples based on past observations. The proposed architecture uses both long- and short-term memory RNNs for sequence modeling, combining multiple intermediate aggregation layers to capture spatiotemporal dependencies in sequential data, encoding observed data and producing future predictions. The outcomes of the experiments demonstrate the superior performance of sequence-to-sequence neural network-based deep learning techniques for trajectory prediction when compared to baseline strategies that rely on linear techniques.
When comparing our method to the two alternative approaches, we observed that our method achieved a precision of 0.98 and a similarity score of 0.993691 between the generated trajectory and the actual trajectory. These results demonstrate the effectiveness and reliability of our approach to trajectory generation and analysis.

1) Difference Between Two images

The difference between the generated trajectory and the actual trajectory for the two objects is shown consecutively in Figures 25 and 26.
The two figures display the similarity between the two real trajectories and those created by the arm. We notice that there is no sufficiently significant difference between the two trajectories, which proves the effectiveness and precision of the method applied.

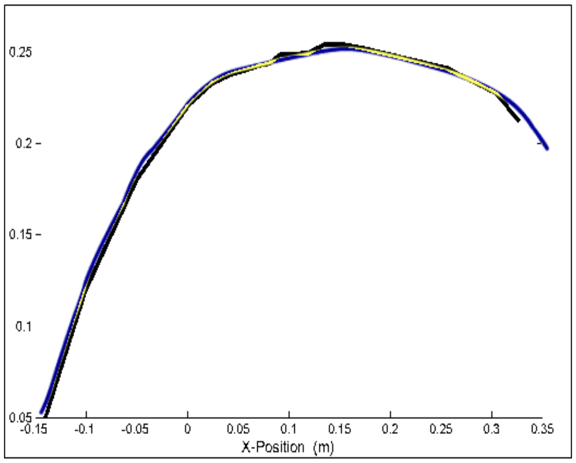

Figure 25. Difference between the real and the generated trajectory path1

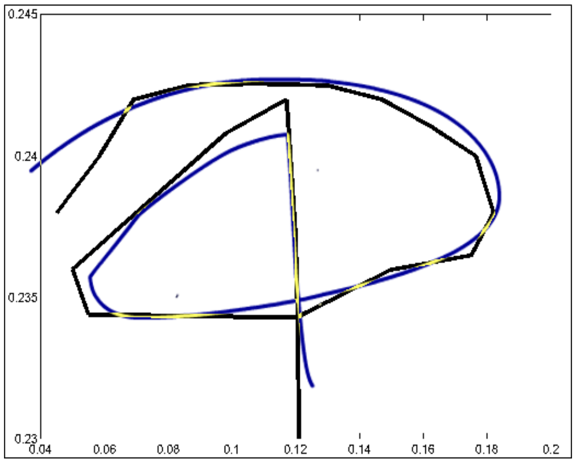

Figure 26. Difference between the real and the generated trajectory path2

2) *The Difference Between the First Real Trajectory and The Trajectory Created by The Arm*

We calculated the difference between the pixels according to each pixel (Figure 27).
In this figure, we observe two prominent peaks in the graph, one at pixel level 150 and another at pixel level 1100. This suggests notable distinctions among the pixels at these positions, potentially reflecting variations in brightness, color, texture, or other attributes. Conversely, the remaining pixels between 0 and 1200 exhibit values uniformly ranging from 0 to 0.5, indicating a high degree of similarity. The minimal difference between these pixels suggests that they appear nearly identical.
The graphic effectively illustrates the pixel-by-pixel fluctuations, reflecting the degree of object detection precision attained. We may further improve the trajectory to guarantee smoother and more accurate motions by adding the manipulator arm's kinematic constraints.

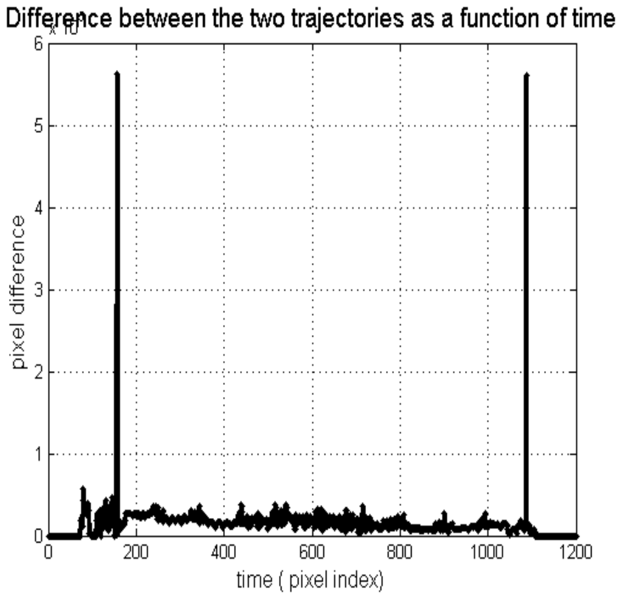

Figure 27. Difference between the real trajectory and the trajectory created by the arm

3) *Similarity Metrics*

The calculation of similarity between two images using metrics provides a quantitative measure of how closely the detected objects match the ground truth, allowing for better detection optimization. In this section, we used MSE, $PSNR$, and $cosinesimilarity$ to evaluate the performance of our detection algorithm. In order to calculate these metrics, we followed these steps: First, we resized the two images containing the real trajectory and the trajectory created by the arm ($image1$) and ($image2$) at the same height and same width.
Convert images to matrices of doubles.
Convert images to gray-scale.
Finally, we apply the following formulas:

1) The (Mean Squared Error) MSE
   we named $squared-difference$ as DD, so we have

$$DD = (double(image1) - double(image2)) \times 2 \quad (19)$$

$$MSE = mean(squared_difference(:)) \quad (20)$$

   Its result is 509.943256. This rating suggests that, while the images are still quite comparable, there are some obvious variances between them. Minor changes in the illumination, noise, or other elements may be the cause of the differences.
2) The Peak Signal-To-Noise Ratio Or PSNR

$$max_intensity = 255 \quad (21)$$

$$psnr = 10 * log10((max_intensity^2)/mse). \quad (22)$$

   Its result is 21.055585dB, which is moreover relatively high, indicating that the quality of image compression or reconstruction is excellent. A high PSNR is frequently linked to the same pictures

3) Cosine Similarity

$$vector1 = double(image1r(:)). \quad (23)$$

$$vector2 = double(image2r(:)). \quad (24)$$

$$cosine - similarity = \left(\frac{dot(vector1, vector2)}{(norm(vector1) \times norm(vector2)} \right) \quad (25)$$

The result of the cosine similarity is 0.993691. The two images are extremely similar to one another based on this value's proximity to one.
The figure below (Figure 28) depicts the degree of similarity between the two graphs in relation to these three metrics.

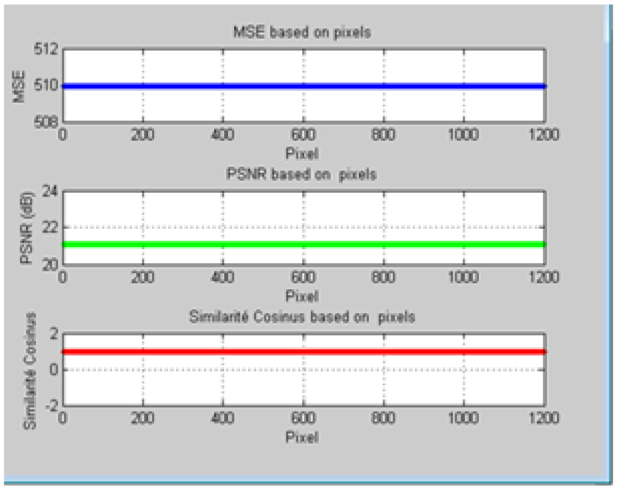

Figure 28. Image metrics comparison based on pixels

4. Conclusion

In this research paper, we applied a method to simulate and generate trajectories for a SCARA robotic arm with 2 degrees of freedom (2DOF). The approach involved utilizing Convolutional Neural Networks (CNNs) to detect tracked objects in a sequence of arm-captured images. The CNN outputs were then used as inputs for the SCARA robotic arm (2DOF), incorporating specific constraints and leveraging a modified Denavit-Hartenberg model by Khalil tailored to the SCARA arm's kinematic structure, enabling precise and efficient trajectory planning.

Using the previously developed arm model, trajectories were generated, and the Spline3 method was applied, achieving an impressive accuracy of 98.99%. Evaluation criteria encompassed Maps, F1-score, recall, precision of detection, Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and cosine similarity of trajectories.

When comparing our method to alternative approaches, our method exhibited a precision of 0.98 and a notable similarity score of 0.993691 between the generated trajectory and the actual trajectory. These results underscore the effectiveness and reliability of our approach to trajectory generation and analysis, highlighting its capability to produce accurate trajectories with minimal deviation from the desired path.

The demonstrated high degree of trajectory similarity showcases the method's potential to generate precise trajectories by accurately positioning objects using CNN-based object detection (OD). This research offers significant potential benefits across diverse fields, particularly in medicine, where robots can enhance surgical procedures by accurately tracking essential surgical objects.

It's important to note that this study did not address certain constraints such as speed, acceleration, or collisions, which are essential considerations for future research and practical applications in robotic trajectory planning

References

[1] K. Ning, T. Kulvicius, M. Tamosiunaite, and F. Wörgötter, "A novel trajectory generation method for robot control," *Journal of Intelligent & Robotic Systems*, vol. 68, pp. 165–184, 2012.

[2] Y. Fang, J. Qi, J. Hu, W. Wang, and Y. Peng, "An approach for jerk-continuous trajectory generation of robotic manipulators with kinematical constraints," *Mechanism and Machine Theory*, vol. 153, p. 103957, 2020.

[3] D. Malyuta, T. P. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, and B. Acikmese, "Convex optimization for trajectory generation," 2021.

[4] M. Sushma Sri, B. Rajendra Naik, K. Jayasankar, B. Ravi, and P. Praveen Kumar, "On the use of region convolutional neural network for object detection," in *Data Engineering and Communication Technology: Proceedings of ICDECT 2020*. Springer, 2021, pp. 315–324.

[5] C. Vimala, T. Thenmozhi, M. Jagadeesh Kumar, and C. Subramani, "Convolutional neural network-based automatic object detection on aerial images," in *Sixth International Conference on Intelligent Computing and Applications: Proceedings of ICICA 2020*. Springer, 2021, pp. 363–370.

[6] Y. Anthony and G. Cameron, "Use articulation bodies to easily prototype industrial designs with realistic motion and behavior," 2020.

[7] Q. Agrapart and A. Batailly, "Cubic and bicubic spline interpolation in python," Ph.D. dissertation, École Polytechnique de Montréal, 2020.

[8] A. Abadi, "Contribution à la génération de trajectoires optimales pour les systèmes differentiellement plats application au cas d'un quadri-rotor," Ph.D. dissertation, Université d'Orléans; Ecole Nationale d'Ingénieurs de Sousse (Tunisie), 2020.

[9] J. Zhao, X. Zhu, and T. Song, "Serial manipulator time-jerk optimal trajectory planning based on hybrid iwoa-pso algorithm," *IEEE Access*, vol. 10, pp. 6592–6604, 2022.

[10] D. S. Carabis and J. T. Wen, “Trajectory generation for flexible-joint space manipulators,” *Frontiers in Robotics and AI*, vol. 9, p. 687595, 2022.

[11] C. Zhao, Y. Zhu, Y. Du, F. Liao, and C.-Y. Chan, “A novel direct trajectory planning approach based on generative adversarial networks and rapidly-exploring random tree,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17 910–17 921, 2022.

[12] J. Li, M. Ran, and L. Xie, “Efficient trajectory planning for multiple non-holonomic mobile robots via prioritized trajectory optimization,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 405–412, 2021.

[13] H. Bilal, B. Yin, and A. Kumar, “jerk-bounded trajectory planning for rotary flexible joint manipulator: an experimental approach.” *Application of soft computing*, vol. 27, p. 4029–4039, 2023.

[14] V. Schülé and D. Genoud, “Détection automatique d’objets agricoles avec du machine learning,” *Master’s thesis*, 2018.

[15] T. NITHISSH and P. RAMPRAKASH, “Real time multiple object detection with yolo,” 2021.

[16] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, “Yolov9: Learning what you want to learn using programmable gradient information,” *arXiv preprint arXiv:2402.13616*, 2024.

[17] K. Smeda, “Understand the architecture of cnn,” *Towards Data Science*, 2019.

[18] K. Mathia, *Robotics for electronics manufacturing: principles and applications in cleanroom automation*. Cambridge university press, 2010.

[19] H. Makino, “Development of the scara,” *Journal of Robotics and Mechatronics*, vol. 26, no. 1, pp. 5–8, 2014.

[20] L. A. Soriano, J. d. J. Rubio, E. Orozco, D. A. Cordova, G. Ochoa, R. Balcazar, D. R. Cruz, J. A. Meda-Campaña, A. Zacarias, and G. J. Gutierrez, “Optimization of sliding mode control to save energy in a scara robot,” *Mathematics*, vol. 9, no. 24, p. 3160, 2021.

[21] P. Corke, “Denavit-hartenberg notation for common robots,” *PeterCorke: Brisbane, Australia*, 2014.

[22] B. Bayle, J.-Y. Fourquet, F. Lamiraux, and M. Renaud, “Kinematic control of wheeled mobile manipulators,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2. IEEE, 2002, pp. 1572–1577.

[23] W. Khalil and J. Kleinfinger, “A new geometric notation for open and closed-loop robots,” in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3. IEEE, 1986, pp. 1174–1179.

[24] C. Feng, G. Gao, and Y. Cao, “Kinematic modeling and verification for a scara robot,” in *2016 3rd International Conference on Materials Engineering, Manufacturing Technology and Control*. Atlantis Press, 2016, pp. 918–921.

[25] W. Khalil and E. Dombre, “Modelisation, identification and control of robots,” *Hermes Penton Science, London*, vol. 67, 2002.

[26] R. P. Paul, *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. Richard Paul, 1981.

[27] D. L. Pieper, *The kinematics of manipulators under computer control*. Stanford University, 1969.

[28] R. Kumar, A. Dan, and K. Rama Krishna, “A note on implementation of raghavan–roth solution for wrist-partitioned robots,” in *Machines, Mechanism and Robotics: Proceedings of iNaCoMM 2019*. Springer, 2022, pp. 687–696.

[29] X. Zhou, Y. Xian, Y. Chen, T. Chen, L. Yang, S. Chen, and J. Huang, “An improved inverse kinematics solution for 6-dof robot manipulators with offset wrists,” *Robotica*, vol. 40, no. 7, pp. 2275–2294, 2022.

[30] S. Capobianco, L. M. Millefiori, N. Forti, P. Braca, and P. Willett, “Deep learning methods for vessel trajectory prediction based on recurrent neural networks,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 6, pp. 4329–4346, 2021.

Aicha Belalia , Bac in 2003. She obtained her engineering diploma in computer engineer (computer networks).from USTO Oran University.in 2008 She obtained a magister diploma in Computer science (simulation and modeling of systems) from USTO Oran University in 2014. PhD candidate studying in computer science. She is currently an assistant professor of computer science at a university and a supervisor of L3 students.

Samira Chouraqui Pr. Samira Chouraqui is a full professor at the computer science department of USTO ORAN University, Algeria. She obtained her engineering diploma in Electrical engineering from USTO Oran University. The MSc degree in satellite communication from Surrey University in UK. The Ph.D. degree in computer science from the USTO ORAN University. Current research interests are in the area of computer vision, artificial intelligence and UAV. E-mail: samirachouraqui@univ-usto.dz

M’hammed Boussir Bac, 1991. 1998 USTO state computer engineer (computer networks). 2012 USTO a magister diploma in physics (matter and radiation). PhD candidate studying radiation and matter physics.he Is employed with the vocational training institute as a teacher. Overseeing trainee students in IT technology. Algorithm researcher in materials physics and radiation